Robust Modelling and Efficient Recognition of Quasi-Flat Objects - Application to Ancient Coins

Florian Lardeux

LA ROCHELLE UNIVERSITÉ

ÉCOLE DOCTORALE EUCLIDE

LABORATOIRE L3i

THÈSE présentée par :

Florian Lardeux

soutenue le : **4 janvier 2022**

pour obtenir le grade de : **Docteur de La Rochelle Université**

Discipline : **Informatique et Applications**

Modélisation robuste et reconnaissance efficace d'objets quasi-plans — application aux monnaies anciennes

---

Rapporteurs	Martin KAMPEL	Professeur des universités	Université de Vienne
	Nicole VINCENT	Professeure des universités	Université Paris Descartes
Examinateurs	Bertrand KERAUTRET	Professeur des universités	Université Lumière Lyon 2
	Pascal DESBARATS	Professeur des universités	Université de Bordeaux
Direction	Sylvain MARCHAND	Professeur des universités	La Rochelle Université

Thèse réalisée au Laboratoire L3i
Faculté des Sciences et Technologies
Avenue Michel Crépeau
17042 La Rochelle cedex 01

Tel: +33 5 46 45 82 62
Web: http://l3i.univ-larochelle.fr

Sous la direction de

Sylvain Marchand	Professeur des universités	La Rochelle Université
Petra Gomez-Krämer	Maître de conférences	La Rochelle Université

Robust Modelling and Efficient Recognition of Quasi-Flat Objects – Application to Ancient Coins

Abstract

Quasi-flat objects are obtained from a matrix which defines specific features observable in their engraving. Some examples of these are dry stamps on documents; amphora stamps, which are obtained with a wooden matrix printed on clay; or ancient coins, obtained with a metal matrix (the die) which leaves a mark on a softer metal piece (the flan). The case of ancient coins will be taken as an example throughout this work. Quasi-flat objects are subsequently understood as very flat shapes onto which a characteristic relief is inscribed. Therefore, the recognition of the objects and their matrix can be performed using their topography, which allows for the choice of a 2.5D geometry paradigm for their description. The categorization of quasi-flat objects is not straightforward and requires expertise. Nevertheless, even for experts the task of classifying can quickly become strenuous, especially when it comes to comparing numerous items with each other. Numerically recognizing such objects is not an easy feat either as many barriers come into play. The relief of quasi-flat objects is prone to non-rigid deformations which implies substantial intra-class variations along with low inter-class variations. The illumination conditions are a key component for the analysis of the object's relief, which limits the relevancy of mere images to study them. Furthermore, these items may have undergone various deteriorations, leading to the occlusion of some parts of their relief. In the field of computer vision, techniques have been developed in order to deal with the analysis, the recognition and the classification of such objects. The majority of them relies on the use of images which, we argue, is patent but impractical in real conditions. In this dissertation, we tackle the issue of recognizing quasi-flat objects, which encompasses the choice of a model and techniques to perform matching or classification.

Remerciements

Je voudrais tout d'abord remercier mes encadrants, Sylvain et Petra, qui ont su m'accompagner, m'aiguiller et m'encourager durant toutes ces années, que ce soit lors des périodes de grande difficulté ou dans les moments plaisants du monde de la recherche scientifique.

Ensuite, il est important pour moi de remercier ma mère, une alliée précieuse qui m'a épaulé, qui m'a aidé à continuer à bien des reprises, et qui m'a soutenu pendant toutes les épreuves que j'ai traversées.

Enfin, j'aimerais remercier tous mes amis sans qui la vie de thésard aurait été plus terne. Les rires, les larmes, les conversations et l'entraide partagés ont été des éléments importants constitutifs de mon voyage.

Contents

List of Figures

List of Tables

List of Symbols

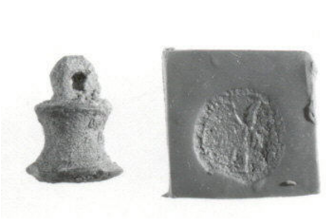
U	2D lattice
I	Image
x	Abscissa in the image plane
y	Ordinate in the image plane
z	Height (axis perpendicular to the image plane)
$\mathcal{F}$	Fourier transform
ω	Point in the Fourier space
$\imath$	Imaginary number
$\vec{n}$	Normal vector
α	Normal azimuth
σ	Normal slant
$\vec{l}$	Light vector
φ	Light azimuth
τ	Light colatitude
n_l	Number of lights
$\mathcal{E}$	Energy map
$\mathcal{A}$	Orientation map
$\mathcal{M}$	Coin mask
S	Sum of images
$\mathcal{S}$	Set of segments
ℓ	Length of a segment or a contour
s	Curvilinear abscissa
$\mathcal{G}$	Gaussian convolution kernel
$\mathcal{B}$	Blob map after local thresholding
f	Contour signature
$\mathcal{D}f$	Signature descriptor
$\mathcal{C}$	Shape complexity
ψ	Contour turning angle
H	Statistical entropy
D	Graph density
Λ	Contours database
n_Λ	Number of contours

h	Hash function/table
$\mathcal{H}$	System (set) of hash tables
n_u	Number of tables in the system
n_v	Number of nested arrays per table
u	Index of table
v	Index of nested array
q	Quantization parameter(s)
ε	Tolerance of geometric alignment
m	Maximum number of retrieved items per table
η	Score from retrieval
γ	Collisions
N_B	Number of bins in the histogram of alignment
j	Contour index
$\mathcal{T}$	Image structure tensor
J	List of coin indices
n_c	Number of coins in the database
X	List of interest points
n_X	Number of keypoints in a coin
R	Detector repeatability
$\mathcal{P}$	Patch region
c	Side length of a patch
$\theta_{\mathcal{P}}$	Patch angle
β	Minimum fraction of the patch inside the coin mask around a keypoint
d	Feature descriptor (HOG or HOME)
K	Number of final retrieved candidates
A	Clustering adjacency matrix

Chapter 1

Introduction

Quasi-flat objects are flat shapes with specific engravings on one or both sides. These characteristic features are obtained via the fabrication of these objects: they are struck with a matrix, which imprints a distinctive relief on it. Quasi-flat objects are best described by this relief, therefore any relevant information to identify or classify them has to be looked for in it, which restrains the object to a height map or depth map, thus enabling us to work with a 2.5D geometry. There is a variety of such objects: dry stamps on documents; amphora stamps, which are made with a wooden matrix printed on clay; medieval tiles; or ancient coins. Visual examples are presented in Figure 1.1. Throughout this dissertation, we will focus our research on the analysis of ancient coins, although almost every result can be inferred for other types of quasi-flat objects. Ancient coins are more available in general and many specimens of various types exist. Their conception is specific to quasi-flat object making and is detailed below.



(a) Seal and stamp seal

(b) Amphora stamp

(c) Ancient roman coin

Figure 1.1: Examples of quasi-flat objects.

1.1 The example of ancient coins

Coins were, most of the time, individually hammered between dies. Dies were carved or engraved by hand, which was the work of an artist engraver called a *celator*. See in Figure 1.3[1] an example of a reverse die. Very few dies have survived since most of them were melted down in order to prevent counterfeiting. The first step is to prepare a coin blank or flan from metals extracted from mines. Those flans could be struck directly while cold if the metal was sufficiently soft, such as pure gold or silver. Alternatively, they were heated before being struck. Usually, the obverse die was attached to an anvil while reverse die was welded to an iron shank held by the mint worker. He then proceeded to strike with a hammer. Figure 1.2[2] presents an illustration of the process. Sometimes, several striking were necessary and double-striking patterns emerged as a result of slight shifts of the dies between blows.

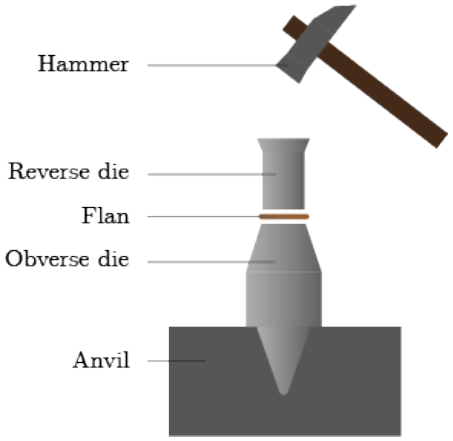


Figure 1.2: Making an ancient coin. The flan is placed between two dies and the whole rests on an anvil. The mint worker then strikes the upper die with a hammer.

The obverse die was usually deeper than the reverse and typically features the head of the current ruler or a divinity, a design called "type". The dies naturally wore out with time and could last from some months to several years depending on the frequency of their usage, which represent up to several dozen hundred strikes. Some dies were kept even if they were cracked and could also repaired if broken. The legitimacy of coins, which gave their monetary value, came from the particular designs of the "type" – which as it as been mentioned often represents a human or divine figure – and the "legend", which is composed of engraved letters.

The strikes were likely to be off-center if the flan was not held steady. Consequently, the engraving could appeared cropped on the flan, as presented in Figure 1.4a.[3] More-

[1] Source: https://www.ngccoin.com/.

[2] Sources: (a) https://www.metmuseum.org/art/collection/search/326634; (b) Badoud 2017 [10]; (c) https://www.forumancientcoins.com/catalog/roman-and-greek-coins.asp?vpar=580.

[3] Source: https://www.ngccoin.com/news/article/4357/ancient-coins-/.

Figure 1.3: Example of a reverse die (Demetrius I of Bactria, 200-185 BCE).

over, oftentimes the dies were not perfectly parallel to each other. This resulted in an unevenness of the strike, which thus applied an uneven pressure on the die, causing the final coin to appear blurred. Finally, depending on the strength of the blow, the imprint could be more or less deep, resulting in various qualities of details: the stronger the strike, the more details can be outlined, as can be seen in Figure 1.4b.

(a) If the flan is not correctly centered, the resulting coin appears cropped.

(b) Impact of strength and unevenness resulting in more or less details and blurs.

Figure 1.4: Various strike perturbations.

1.2 Issues, barriers and problematic

As we have stated by evoking the case of ancient coins, there are several aspects of the fabrication which make each object really unique, although certain classes can be determined. Recognizing a quasi-flat object, which, as we have noted, are mostly found the field of archaeology, is a difficult task, often requiring an expertise so that each object can be identify as belonging to a certain type, being created during a specific era, etc. Years of expertise in the field cannot sometimes aid one in performing efficient recognition, especially if the number of specimens is large and the task is demanding.

Pairwise matching of coins, for instance, is extremely strenuous and time-consuming.

In the case of ancient coins, it has already been mentioned that one die can be used many times before being replaced by another one. This new one is also handcrafted; it is impossible for it to be an exact replica of its predecessor. Therefore, coins struck by the former die are different than future coins struck by the latter, new die, even though they have the same symbols and monetary value. In this dissertation, the work is centered around quasi-flat object recognition, especially when it comes to objects struck with the same matrix. There are various pitfalls on the way, some of which are illustrated in Figure 1.5:

- **Illumination conditions**. These objects do not feature specific textures – in the sense that there are no drawings or inks – so what is perceived is determined by the object and the light direction. It is, in a way, akin to illuminate a human face from different angle: it changes completely the perception of the image, which, for a machine, complicates the recognition of the target.

- **Occlusions**. This may originate from various reasons: the object is struck off-center (Figure 1.4a), parts of the object are missing or oxidation. Occlusions subsequently involve missing parts of original data, which naturally alter the efficiency of the recognition tools; global recognition, for example, is most likely to fail in this type of scenario.

- **Altered quality**. Because of the way the object is struck – for instance, hammering a flan with two unparalleled dies (Figure 1.4b) – and because of erosion and tear, the object may have lost its initial details, depending on the way it has survived several millennia. This can also entail a form of occlusion because, in a way, it may implies that data are missing. The information may as well be preserved, but disrupted because of some erosion effects or localized fractures. Moreover, it also adds new undesirable information, which can also be taken into account, but should not alter the recognition.

- **Non-rigid deformations**. Changing the matrix inevitably changes the exactitude of the engraving. Therefore, there can be differences between coins of the same type. For instance, when a coin die is replaced, but keep its original symbols, legend and overall monetary value, the new die is undoubtedly different – remember that the die is handcrafted – and therefore slight non-rigid deformations are visible from one die to another, and consequently from a coin from the first die and a coin from the new one.

For any classification task, given the types of the objects, the way they are fabricated and the various distortions they can suffer, one can easily predict high intra-class variations and very low inter-class variations. In other words, it is sometimes hard to actually cluster objects from the same type because of a significant amount of differences between the objects inside the class; while appropriately distinguishing two very similar objects not belonging to the same class can also be challenging.

(a) Impact of light conditions

(b) Impact of erosion

(c) Impact of changing matrix

Figure 1.5: Examples of perturbations which change or alter the object's quality.

The automation of recognition jobs is sought-after because it is desirable to avoid having to rely on the scrutiny of experts in the field to, at least, be able to narrow the scope of research to know more about an object, its relationship with some other objects, the type which it belongs to, the matrix it was struck with or even identify the object. The goal, then, is to build a solution, ideally non-invasive, which could help either researchers save time and effort or even non-experts retrieving efficiently information about a recently found object. With all the various blocks in mind, it is obvious that the deed is not an easy challenge. Solving this problem inevitably encompasses many sub-tasks, such as determining a way to extract data from the object, organizing and analyzing these data, adequately storing the data, constructing tools for matching parts of the data and retrieving the correct final information about the object.

Because they are more available than any other quasi-flat objects, ancient coins will be used as an example for every experiment. Yet, most mathematical models, definitions and algorithms remain true and transferable to other quasi-flat objects.

1.3 State of the art

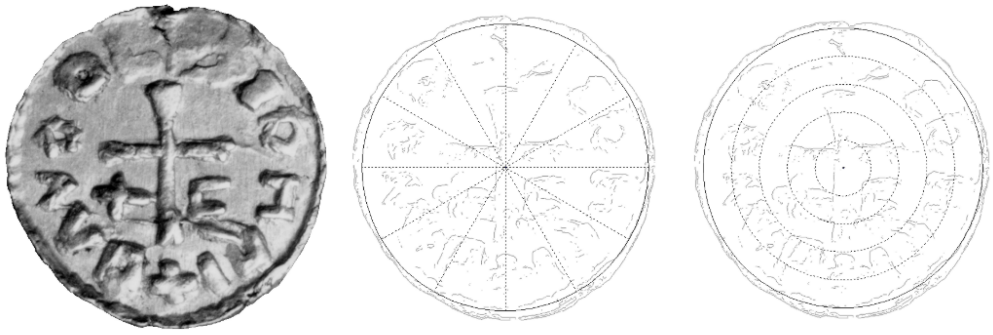
There are many ways to analyze, recognize or classify coins, as for any objects on which measurements can be performed. Early techniques focused on directly measurable information such as mechanical properties: weight, diameter, thickness [32]; electric properties: permeability, conductivity [32], oscillating electromagnetic field characteristics [98], or piezoelectric properties [93].

Computer vision can provide a large panel of tools for coin analysis using 2D or 3D. Single images can be used as a whole with the intent of using a global recognition method. Basic signal processing techniques such as cross-correlation or frequency analysis can be employed. The Fourier-Mellin transform has been used to perform registration of ancient coins by estimating scaling, rotation and translation between two images of the same object [72, 71]. In [71] in particular, Marchand proposes an image-to-model iterative algorithm for registration, which successively estimates and updates the viewing and lighting conditions; the process starts with random lighting parameters. To do so, the author considers the Fourier transform of the image model $\hat{I}$ as the product of a Lambertian reflectance [60] and retrieve the object's normals $\vec{n} = (n_x, n_y, n_z)$ via Equation (1.1), where $\mathcal{F}^{-1}$ is the inverse Fourier transform, $\hat{I}^*$ is the conjugate of the Fourier image I, N is the Fourier transform of $\vec{n}$ and $c_{x,y,z}$ are coefficient found by aligning one image with the other.

$$c_{x,y,z} = \mathcal{F}^{-1}\left(\frac{\hat{I}^* \cdot N_{x,y,z}}{|\hat{I}|^2}\right) \tag{1.1}$$

Experiments proved to be successful in comparison to using mere images, especially when captured with different light conditions. Indeed, correlation is not robust enough to deal with small variations, and thus cannot reliably account for differences in illumination of single images. For various reasons besides light conditions, it is largely preferable to locally describe the objects. For instance occluding manifestations such as wear, tears or

oxidation as well as non-rigid deformations present within the relief of the object make it unlikely for a global description to adequately enable two objects to be compared. Since quasi-flat objects recognition undergoes first and foremost illumination issues, many techniques deemed the utilization of edges and contours to be relevant. Nölle et al. [78] was the first to introduce this idea for the recognition of modern coins. Of course, modern coins feature a sharper relief which is easily captured by a Canny edge detector. Edges can be tricky to analyze because they feature topological discontinuities which are influenced by to a variety of factors including light conditions and occlusions. The authors used the edgels independently and sampled their occurrence in quantized polar coordinates. The amount of missing and superfluous edgels give them a score which enable a direct comparison between two coins. Also on modern coins, Huber-Mörk et al. [46] performed a classification using an eigenspace approach on edge images of the coins. Various other works [69, 68, 108] used similar approach for both modern and medieval coins, using for instance angular and radial histograms of the edgels (see Figure 1.6). The edge detection was performed using Sobel kernels and thresholding and the edgels were pooled into distance and angle histograms, relying once again on a polar representation of space. One can make two remarks on their results. Firstly, the distance histogram is a far better sampling of the 2D space than the sole angle histogram – 68% vs 17% accuracy [69], up to 78% using a combined angle-distance histogram [68] – mostly because the former is rotation-invariant. Secondly, the classification accuracy drops immensely when



(a) Original coin image. (b) Edge angle histogram. (c) Edge distance histogram.

Figure 1.6: Edge histograms designed by Maaten et al. [69] (Figure 1). Lighting conditions have a direct impact on the edge distribution and because of other perturbations such as non-rigid deformations, classification accuracy on medieval coins is very low.

applied to medieval coins. This suggests that the edge distribution is largely disturbed by occlusion and non-rigid deformations. Moreover, ancient coins are not perfectly round, meaning their "center" does not really inherently exist, thereby leading to unavoidable misalignments of the polar coordinates between two coins. Besides internal edges, the contour of a coin is a relevant information for the analysis of ancient coins, since no two such coins feature the same general shape. Zaharieva et al. [107] exploited this idea to identify ancient coin images taken under different conditions. They used shape contexts

for the contour representation, which features a local, per-edgel distribution of edgels. By combining it with a texture descriptor, they achieve very good results for various camera and lighting conditions. However, note that they used a dataset containing up to three images per coin; classification is biased by this example availability. In general, contour analysis is a fairly relevant technique for identification, but does not make sense for classification, since only the metal flan is responsible for the coin shape.

The second category of techniques for coin description is the set of algorithms and mathematical tools to describe textures provided by signal processing and later by image processing. Texture description details the distribution of image intensities at a given location. It seems relevant for coin analysis since a coin surface, as seen in an image, is only a collection of distinctive grey-value variations. Reisert [83] employed a strategy quite close to edge detection and analysis, except they did not really compute edges. They begin by deriving the image gradient, from which they infer the gradient direction in order to align coins. This idea has only been applied to modern coin, and no remarks have been made on ancient coins. The authors mentioned that using only orientations makes the technique illumination-invariant and eliminate any threshold parameters, which would depend on light conditions – the Canny filter is an example of this. Maaten et al. [69] tried Gabor as well as Daubechies wavelets. These features alone give medium results on modern coins (respectively 55% and 46% accuracy) and very poor results on ancient coins. Of course, the standard SIFT descriptor has been used for coin classification. Zaharieva et al. [107] tried it on various dataset, including modern and ancient coins, with convincing results. Kampel et al. [50] did a survey of various descriptors, including SIFT [67], GLOH [74], SURF [12] and shape contexts [15], along with several interest point detectors. Similarly to the previous work [107], the dataset included several images per coin, including both sides. The classification process included a verification step to check the coherence of both sides. The results depend largely on both the choices of the detector and the descriptor. Overall, SIFT and SURF seem to be the most robust to the choice of the detector. In another paper [110], the authors implemented a solution using SIFT-flow, which consists in minimizing an energy function over a dense grid of SIFT descriptors. The dataset was a set of 24 classes of 3 images of ancient coins. Results show that SIFT alone performs poorly for ancient coin classification (29.2%), whereas SIFT-flow ensure a more reliable score (68.8%). Aslan et al. [7, 8] also computed a dense SIFT over the coin. Their strategy is however original in that they use graph transduction games inspired from from game theory to let the images "choose" their class label according to their descriptors and the choice of other coins. They reached up to 87.2% accuracy on the Roman Republic coin dataset. Zambanini et al. [111] developed a new descriptor which tackled specifically the issue of illumination. The goal was to find an efficient description of the coin which is robust to light conditions. The authors employed Gabor wavelets combined into a single histogram descriptor, the LIDRIC. Considering an image $I_{\theta_i,\omega_j} = |G_{\theta_i,\omega_j} * I|$ obtained by convolving the image with a Gabor kernel G_{θ_i,ω_j} of orientation θ_i and scale ω_j, the descriptor is essentially, for each location (x, y), the resulting four-dimensional

and normalized vector:

$$\tilde{F}(x,y,\theta_i,\omega_j) = \frac{I_{\theta_i,\omega_j}(x,y)}{\sqrt{\sum_i \sum_j I_{\theta_i,\omega_j}(x,y)^2}} \tag{1.2}$$

Equation (1.2) is then used to build histograms by pooling the values across various cells over the image. Experiments to test the invariance to light conditions were not performed on coins; it can be argued that the datasets used to assess the robustness of LIDRIC are not sufficiently close to the specific geometry of coins. In another study [113] however, experiments were made on Roman coins: they achieve between 68.6% and 95.8% accuracy by combining LIDRIC and SIFT-flow. Finally, an approach using binary patterns has been tested [65]. Namely, they use Local Binary Patterns (LBP) for local texture description and Local Pattern of Co-occurrence Matrix (LPCM) for rotation and illumination invariance. However, the technique is only applied to modern coins, on which they are able to superimpose a polar coordinate system. It is also possible to combine different ways of describing the coin in order to increase to accuracy of recognition. In two articles [48, 47], both contour information – described with DCSM, a descriptor based on the coin radius variations – and texture description (SIFT) were analyzed and fused for coin identification. They achieve 98.5% accuracy with SIFT and DCSM on ancient coins, interestingly increasing only by 1 point the accuracy of DCSM alone. This is not surprising, given the discriminative power of the shape of ancient coins. A different approach to information fusion was also tested [112, 55] in which local image description (SIFT-flow) and textual information (extracting letters from the legend of coins) are merged. While the top accuracy achieve by the fusion reaches 91.4%, this represent only a 4.5 points increase from the accuracy of using only the image content. One can also note that legends are not always present on ancient coins, so this might not generalize well to coin other than roman republican coins, which they used.

A common general idea within the coin analysis community is to keep in mind that the coin structure is critical for its recognition. The experiments of Anwar et al. [2] cue on the importance of spatial information. They took a SIFT-based approach with a dense sampling and adopted a bag of words strategy. The bag of words technique loses immediately the spatial information since it samples only the occurrence of a given descriptor value. The authors tried several tiling scheme to compensate this loss: circular, rectangular and radial-polar (see Figure 1.7). The latter gave the best results, up to 93% with a small vocabulary size, while no spatial tiling yielded up to 87% with a very large vocabulary size. Arandjelovic et al. [5] obtained an even lesser result with a bag of words without tiling (2.4%). In their work, they also argued that "what allows a human expert to distinguish between [coins] is, in large part, the geometric relationship between their otherwise common features". To follow this perspective, they developed the locally-biased directional histogram, which allows to base the description of the image from the point of view of each interest point. The classification accuracy reaches 57.4% on the Roman Imperial coin dataset, which is a richer dataset (65 classes) than the one used by [4] (21 classes). Anwar et al. [3] rely on the bag of words model with circular tiling described in some previous works [2, 4], to which they concatenated what

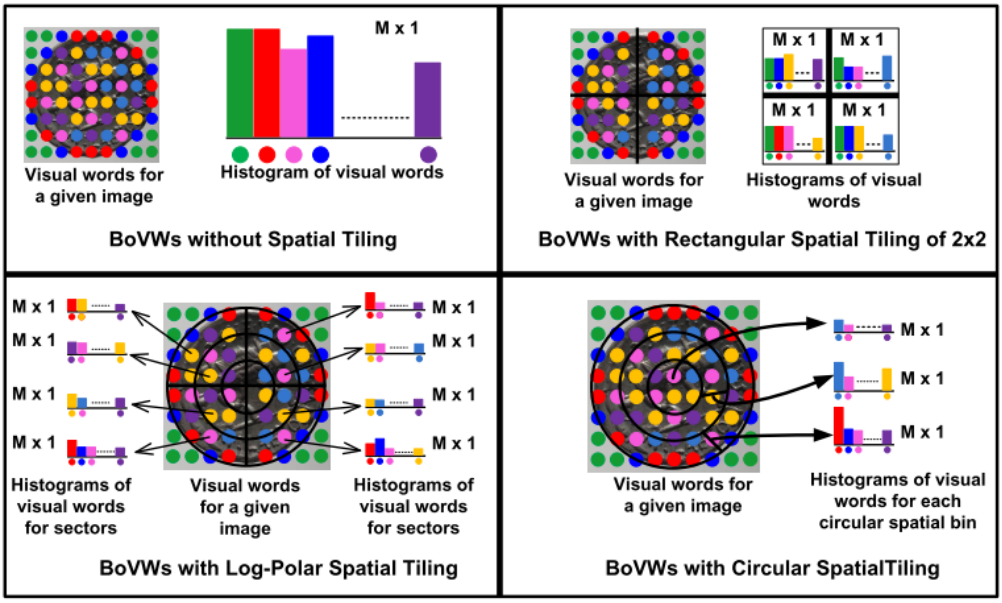


Figure 1.7: Various tiling structures combined with the bag-of-words model on ancient coins [2] (Figure 3). Restricting the pooling to certain areas improves significantly the recognition accuracy while losing structural information is detrimental.

they called pairwise identical words angles histograms (PIWAH) in order to add structural information. Zambanini et al. [113] tackled the structural issue by considering the geometrical consistency in a set of matched features between two coin images. The idea is to compute a dissimilarity based on the number of matches, the displacement between points and the variation in length and orientation of pairs of matching points. This geometrical verification led to improved results in comparison with using only the standard descriptors comparison. Let us finally cite the work of Kim et al. [56, 57] in which the authors intended to use the coin composition knowledge as a mean to improve recognition. This was either done by finding the emperor's face and aligning coins accordingly or by using a deformable part model to align the image by locating the emperor's forehead, nose, chin and ear.

In addition to hand crafted features, some works applied the neural network framework to coin classification. Applications were first focused on modern coin recognition, such as Yens [37] or Rupees [18, 75]. With the arrival of convolutional neural networks, deep learning approaches were adopted. Aslan et al. [8] showed that using transfer learning via a CNN pre-trained from ImageNet leads to very poor results when applied on ancient coins. The AlexNet neural structure was used [22] to recognize modern coins and built a client/server architecture to enable the utilization of a smartphone for recognition. Because of their highly distinguishable relief features and their large availability, modern coins poses few problems to CNN-based techniques, since training is possible over a large number of reliable and clean examples. Even though this is not the case for

ancient coins, some progress has been made on the matter. Kim et al. [58] employed a hierarchical knowledge scheme to make use of the relationship between the roman imperial coins, the depicted emperor and the symbols on the reverse side. The main idea is to detect landmarks in the coin, which embody areas of significant changes in the final classification accuracy. In order words, this is very much akin to a human recognition approach, where the eye recognize specific parts of the engraving as being relevant for recognition. In the same fashion, Schlag et al. [85] intended to identify the emperor shown on the obverse side of the coin. They also performed an analysis to find the parts of greatest discriminative power, which revealed to be located on the face, hairstyle, headwear and clothing. Cooper et al. [27, 28] followed a multi-modal perspective by making use of both the image of the coin and its textual description. The learning is thus weakly supervised, meaning labels are not assigned pixel-wise but image-wise, with a filtering of the description only focusing on 5 symbols. There are around 65000 images, which is comfortable for such a semantic learning with AlexNet. Overall the symbol-based approach is compelling, which seem to go along with human recognition (see Figure 1.8). Also, some authors [1] constructed a new deep learning architecture named CoinNet and center yet again on semantic understanding, associating reverse side symbols to their signification.

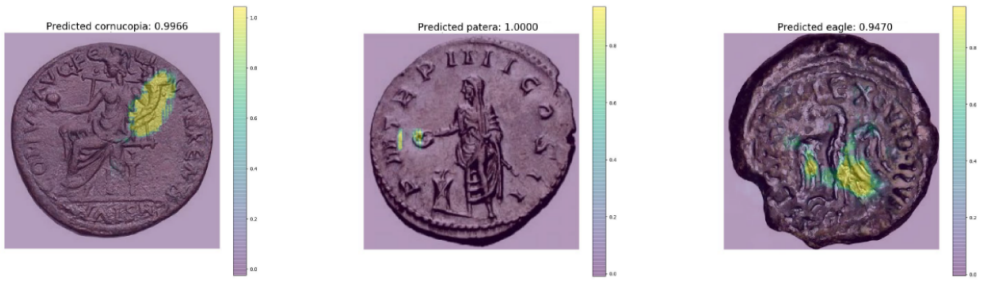


Figure 1.8: Salient regions learned with the neural network structure proposed by Cooper et al. [27] (Figure 8). According to the authors, the learning is adequately correlated with visual symbolic content within the coins which would enable a relevant, symbol-based recognition.

Machine learning for the recognition of quasi-flat objects is not a path that has been taken throughout this thesis. We argue that it is a research topic of its own and the use of learning tools often requires many examples to work well. In our case, we would need to provide a lot of examples of coins labeled by experts. We simply did not gather a sufficient amount of labeled data in the limited span of the thesis to use these techniques properly.

1.4 Summary of the contributions

In this work, we introduce warious elements for the analysis and the retrieval of quasi-flat objects given their properties. The contributions span from the very nature of such

particular objects to the routines built to recognize them.

- We propose a model, the energy map, to represent quasi-flat objects. This model stems from the best way of describing such object which is their surface gradient and is computed using various light orientations. This representation is the cornerstone of this work and thus will be used many times in the various techniques presented.

- Two datasets are presented: a set of 23 real ancient coins and a set of 400 artificial coins. The real coins data were acquired using the technique relevant to the computation of energy maps. The artificial coins followed the standard procedure for striking coins; the protocol was followed as if done in real life by an artist engraver, only it was performed computationally.

- Strategies are developped for using the object's inner contours as a means for recognition. They encompass two main pipelines: a technique to extract meaningful and relevant contours from the object, and an fast exact retrieval algorithm (the LACS) to efficiently match objects using their contours.

- Retrieval using the object's textures is experimented as well. In this regard, we introduce a detector and a descriptor related to the energy map model as well as a retrieval technique, the LAPS, which extends the LACS algorithm to high dimension descriptors. Finally, a clustering strategy is proposed in order to sort objects with respect to their similarity.

1.5 Organization of the thesis

This dissertation will follow two main perspectives: the representation and the recognition of quasi-flat objects. These axes will be broken down into three chapters.

Chapter 2 aims at finding a model to represent quasi-flat objects. There already several possibilities regarding this issue. We will try to sort through the various mathematical perspectives in order to select what can best gather relevant data about the object. The designed model that we introduce is very light and pinpoints the most important information about the such types of objects.

Chapter 3 proposes a first approach to quasi-flat object recognition. In particular, this chapter uses a solution based on contours, which requires explaining what they are, how to extract them and how to match them. A signature representation of such contour elements will be introduced; an efficient storage and retrieval system is presented; and the final pipeline for retrieving objects struck by the same matrix is explained.

Chapter 4 presents another approach to recognition via local texture matching. A process quite similar to the one presented in Chapter 3 is used, although a generalized, slightly more complex version of the retrieval system is described.

The general task has to do with recognition, which itself is indeed a vague term. While most of the state of the art focuses on classification, we insist that defining a class is sometimes not straightforward. Once again, let us specify again that the main examples employed through this thesis are ancient coins. Let us also insist on the fact that this thesis is about *objects*, not just images. Having access to the object is the main concern for gathering data for this work, which reveals to be difficult sometimes, especially to have access to large databases.

A large part of this work is focused on the implementation of low-complexity algorithms. Along with the various techniques of recognition presented, we will insist on the fact that we wish our methods to be fast, especially if one is going to deal with large amount of data.

Chapter 2

Modelization of quasi-flat objects

The first step prior to any analysis of an object is finding a way to acquire information about it. Sometimes raw data are sufficient by themselves, depicting an accurate representation of the studied object. However, there may be some cases for which raw data are not usable and it is necessary to build a model in order to get a correct measure. A model, from latin *modulus* (measure), is a system of postulates about the object which allow to arrange collected data in a way that better fits reality. In the realm of image processing, for instance, the most common model is the image. Studying quasi-flat objects gives the opportunity to pick different options to represent them, and this must not be neglected. This choice is important because it conditions the memory space, the processing time, the difficulty of the acquisition and the modelling complexity. Additionally, future processing can benefit from it insofar as the representation eliminates one or several constraints such as variations in illumination and exacerbates some key features of the object.

2.1 Introduction

This chapter firstly describes the existing literature regarding quasi-flat objects. Given that this terminology is seldom used, we take the example of ancient coins, for which a significant amount of literature has been published. After the examination of relevant works, the energy map, a simple yet informative model which has been published in the Eurographics Workshop on Graphics and Cultural Heritage in 2018 [62] is presented. A mathematical definition and interpretation is proposed as well as a method to compute this model. Secondly, various links with other models are made. This analysis reveals the close connections between models and demonstrates the advantages of the energy map over other candidates. Thirdly, we unveil a way to craft artificial ancient coins in a way that preserves the quasi-flat object topology and respects the protocol for making them in real life. The dataset thus created is useful for further experiments in the absence of a large dataset of real coins. Finally, various tests are run on the energy map so as to show some of its properties.

Figure 2.1: Two photographs of the same coins taken with different conditions of illumination. The impact of light direction on the image is clearly visible.

2.1.1 State of the art

In this section, models that can be used for quasi-flat objects are introduced. The progression of the presentation is organized according to the complexity of both the acquisition and the computation. It noticeably follows that this leads to a natural, chronological unfoldment since there is a common ground between representations and they may be based on each other.

2.1.1.1 Photograph and image

Apart from electro-mechanical sensors, a more usual way to measure the object's information is via the use of a camera, thereby producing an image. Formally, an image is a visual projection of the real world onto a 2D discrete lattice. It is very simple to capture and is close to what we humans see. This is also a very compact representation, making it easy to employ for analysis.

As for any product of a projection, information is lost in the process, and so even though the image model seems visually accurate, it is still fairly imperfect and is subject to many parameters and constraints. Kampel et al. [51] summarized the common issues encountered when capturing images. One has to select the camera, its focal length, its field of view, the sensor resolution, the exposure, etc. Moreover, it is usually essential to take into account the camera angle and the lighting. In the work of Zambanini et al. [114], a similar remark is made regarding the influence of light on the coin material: it usually is reflective and thus prone to highlights. Since such an object displays many relief variations, the acquisition is also subject to shadows; parts of the coin may not be available in the image.

Regarding the issue of light conditions, one could rely on what photographs expert say about lights placement in the case of ancient coins. It has been found [40] that the light elevation must be at the highest possible, which means placing the light sources

very close to the camera lens: "High angles improve the overall lighting of the coin, which prevents dark spots and improves the lighting of the relief". Also, two different setups are recommended. The first one uses two light sources about 90 degrees from each other and the second uses three light sources about 60 degrees apart. Illumination can also be controlled with specific devices featuring one or several light sources and an enclosure. Tompa et al. [96] built an opaque cube on top of which a camera is placed. To ensure equal illumination, light is produced by a ring of light which surrounds closely the camera lens, in accord with previously stated recommendations [40]. This sort of setup is common and convenient to control the illumination and avoid stray light.

In general, using images is somehow mandatory since it constitutes an entry point for visual analysis. The question is rather about how many lights there should be and in what way they need to be arranged. Illumination of quasi-flat objects is a critical issue because, besides determining the brightness, it will produce a two different images given two different light positions because images stem from the conjunction of the object, the light sources and the viewing position (see [77]). This remark has been made by Zambanini et al. [111], stating the difference of interaction with light between textured flat objects and textureless non-flat objects. Other authors [85] even argue that features extracted from a coin image alone such as color, edges or textures are not reliable enough cues due to the specular nature of coins.

2.1.1.2 Normal map and multi-images models

The process of recovering the shape of the object from an image is called shape from shading and constitutes an ill-posed problem: two different surfaces may produce the same shaded image. While the image model is definitely dependent on where the light rays come from, one can propose the use of several images with various illumination conditions so as to eliminate this dependency. Such a model is thus inherently closer to the real object, in a sense, because it becomes a function of only the object and the camera angle. In this context, the object can be seen as a collection of normal vectors, that is a normal map. The goal is therefore to build the normal map from a series of images. This is what Woodham [104] proposed in 1980, designing a technique called photometric stereo. To do so, he hypothesized that the object is a Lambertian reflector [60], which is to say that it diffuses light rays in all directions with the same intensity regardless of the viewing angle. From this assumption, Woodham mathematically derived a way to compute back the normals from a series of three or more images taken with different light directions. With only three pictures taken with different light conditions $\mathbf{I} = (I_0, I_1, I_2)$, one can retrieve the normals by solving the Lambertian equation for the normals $\vec{n}$:

$$\vec{n} = L^{-1} \cdot \mathbf{I} \tag{2.1}$$

where $L = \left(\vec{l_0}, \vec{l_1}, \vec{l_2}\right)$ is a 3×3 matrix of light vectors coordinates. Figure 2.2 shows an example of a normal map of the coin previously presented, for which Equation (2.1) has been solved using least squares. This leads to an approximation since we assumed the

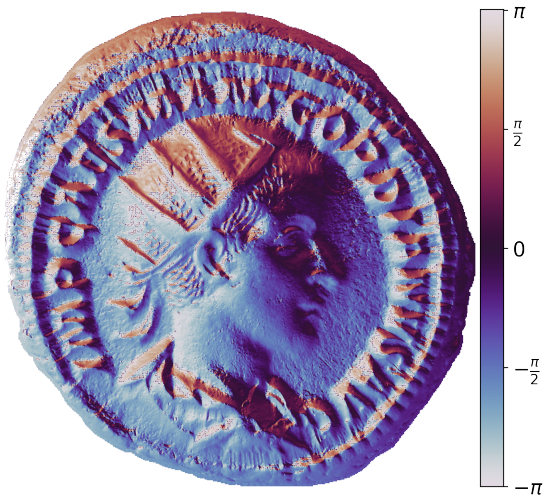


Figure 2.2: Normal map representation.

object to be perfectly diffuse. Yet, the results are often convincing, even in the case of ancient coins, which may be sometimes quite reflective.

Marchand [71] proposes a hybrid registration method which uses both image information and lighting information. The technique uses an algorithm which successively updates the viewing conditions and the lighting conditions. Images are taken from above with a varying light azimuth. Each image undergoes a random Euclidean transformation, then a cross-correlation is performed between each image and a control image to find rotation, scaling and translation. A similarity score is then computed between the two registered images. Results show that correlation alone does poorly when the light conditions are not the same, while using an hybrid model that combines image and normals yields good similarity scores as long as the transformation is limited to translation.

The Polynomial Texture Maps [70] are another model closely connected to the normal map. Information about pixel color and reflectance is stored in parametric pixels. Again, the authors built a device with constrained conditions, featuring a camera on top of the object and LEDs evenly spread over a dome. The combination of the images allow to synthesize a virtual light source which enables the selection of any light condition, source type and reflection model. The general technique here is Reflection Transformation Imaging which was also applied to ancient coins [76], with the intention of making coin examination safer for the coins as well as easier and available for everyone. The setup used to acquire the images consists of a camera, a platform to hold the coin and a light which is manually moved around the object.

Brenner et al. [19] analyzed the impact of light arrangement on the normal map obtained via photometric stereo. Results show that the more light sources there are, the better the model fits the actual object. Moreover, the best light elevation angle was found to be 51°. However, their elevation sampling was coarse, and for this reason

it is reasonable to infer that 45° is the best elevation angle for reconstruction since it separates equally the colatitude range.

Finally, Hossfeld et al. [45] proposed a pipeline for automatic recognition of modern coins which uses a device that simultaneously captures three images of an object. The strategy is to use three different color light sources arranged in three sectors: red, green and blue. This trick enables a fast acquisition and the processing is then performed on a combination of the three resulting images. There is no model creation strictly speaking, but rather than relying on one image, the authors thereby proposed to circumvent the illumination problem by using several light sources.

2.1.1.3 Height map

Starting from the normal map, it is possible to retrieve the height map of the object. Doing so requires integrating the normals which is an well-posed problem, although an integration constant has to be fixed and the outcome is still dependent on the continuity conditions observed. In [81], a survey of shape from shading methods is presented. The authors underline that such a problem is not a straightfoward one, even though it appears to be so, and thus there are lots of different approaches to solve it. The two main techniques for surface recovery remain Horn's method [44] using finite differences, and Frankot-Chellappa's method [35] using complex integration via Fourier transforms. An example of height map is displayed in Figure 2.3 and was computed via Frankot-Chellappa's method from the previous normal map. Namely, via the normal map, one calculates the Fourier transforms N_x and N_y of the first two normal components and derive the Fourier transform of the height map with:

$$\mathcal{F}[z](\omega) = \frac{-\imath\omega_x \mathcal{F}[\vec{n}_x](\omega) - \imath\omega_y \mathcal{F}[\vec{n}_y](\omega)}{\omega_x^2 + \omega_y^2} \tag{2.2}$$

where $\imath^2 = -1$ and $\omega = \omega_x + \imath\omega_y$ is a point in the Fourier space. Then, the height is simply retrieved by inverse Fourier transform.

In general, one notice that height maps are seldom utilized for coin analysis. There are indeed some assumptions made along the way (Lambertian material, continuity, etc.) which may lead to errors, which of course accumulate via the integration. Also, it is more than likely that we do not need such a model to perform recognition tasks given that information usually is extracted via gradients, which normals already represent.

2.1.1.4 3D model

Surface reconstruction, as presented in the previous subsection is not 3D *per se* since there is no way to go "around" the model. Neither is it 2D, one could argue, but 2.5D since it does nonetheless feature one spatial coordinate as a function of the two others. In the case of 3D models, the goal is to obtained a full 3D view of the object, usually in the form of a point cloud or a mesh. This idea was exploited for the 3D representation for ancient coin analysis [115, 43]. Coins were scanned via a stereo scanner, which is a dual

Figure 2.3: Height map obtained with Frankot-Chellappa's method [35].

Figure 2.4: Three-dimensional model of a coin. Courtesy of S. Zambanini [43].

camera featuring a structured light projector. An example of scanned coin is presented in Figure 2.4. The results are visually detailed and satisfying given that they feature measurements of diameters and volumes comparable to the ones obtained directly from the real coins.

The possibility of the use of such a model for automatic recognition has been suggested [115], but so far no direct analysis for recognition on 3D models has yet be presented. Even though 3D models of coins are the closest representation of real objects, are they really necessary in the case of coin recognition? One could argue that it constitutes a fine strategy to employ such a close rendering. However, not all of the data available is relevant. For instance, there is close to no way of using the edge of the coin as a mean of extracting data. Additionally, the visual information can still be shown with PTM techniques [70, 76], which do not require stereo scanning. 3D acquisition

remains moreover laborious and costly, and the final rendering is quite heavy in terms of spatial memory.

The available models range from the simple image to the entire 3D representation of the object. On the one hand, there are lots of information lost in the projection involved in image acquisition, which makes it difficult to compare two objects with different illumination conditions. On the other hand, retaining the data gathered in a point cloud, a mesh or even a polynomial texture map may not be necessary, as the majority of the information can be synthesized into intermediate representations such as the normal map. We posit that we can extract meaningful details about the object only with the use of the object's surface derivatives. In the following sections, we introduce the multi-light energy map; a model which is easy and fast to compute as well as memory-efficient.

2.2 Multi-light Energy Map

In this section, a different perspective is taken to analyze quasi-flat objects in general. Images are sensitive to light conditions and a model which solves this issue right from the beginning may be preferable. Normal maps or multi-images models appear to be a reasonable choice for this purpose. However, the interest of using all the information provided by the normals is questionable. Here the concept of energy map is presented. Its properties are studied and some relationships with other models are established. The energy map is not a 3D model, nor is it explicitely a 2D model, since we will see that the information encoded in it reflect the surface variations of the object. It is more akin to the height map model and is therefore a 2.5D model.

2.2.1 Mathematical approach and definition

Let us consider the configuration depicted in Figure 2.5. The object is located on point O, while the camera aim towards it directly from above. Let $U \subset \mathbb{R}^2$ be the 2D lattice over which an image I is defined. Let $\vec{l}$ be the light vector, defined by its azimuth angle φ and its colatitude angle τ (complementary angle of the elevation).

$$\vec{l}(\varphi,\tau) = \begin{pmatrix} \cos\varphi\sin\tau \\ \sin\varphi\sin\tau \\ \cos\tau \end{pmatrix} \tag{2.3}$$

Let us denote by $\vec{n}(x,y)$ the normal vector of the object at the point $(x,y) \in U$. Likewise, $\vec{n}$ obeys the same equation, albeit respectively with the angles α and σ depicted in Figure 2.5. The normal and light vectors are normalized as such:

$$\|\vec{l}\| = \|\vec{n}\| = 1 \tag{2.4}$$

In Equation (2.3), the angles in this configuration are $\varphi \in [0, 2\pi)$ and $\tau \in [0, \frac{\pi}{2}]$. When $\tau = 0$, the light source coincides with the camera. In Equation (2.4), $\|.\|$ denotes the Euclidean norm.

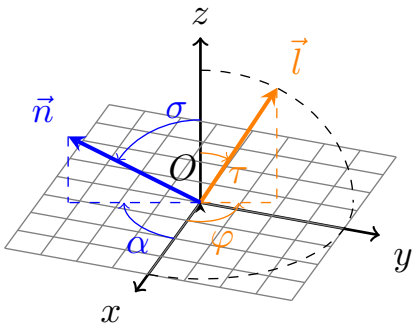


Figure 2.5: Space configuration featuring light and normal vectors.

In general, the reflective properties of a material are simulated with the use of a bi-directional reflectance distribution function (BRDF). There are many types for almost any kind of materials but, in order to simplify calculations, we make the assumption that the object is a perfect diffuser [60]. Therefore light rays hitting the surface of the object are reflected in all directions and the reflected energy is distributed according to the cosine law:

$$\forall (x,y) \in U, \quad I(x,y) = \vec{l} \cdot \vec{n}(x,y) \tag{2.5}$$

From now on, we will drop the variables (x,y), knowing they are implicitly there. It is possible to use the Cartesian coordinates of $\vec{n}$ but it strongly complicates the calculations. Therefore we rearrange Equation (2.5) in order to derive a simpler expression.

$$\begin{aligned} I &= (\cos\varphi\cos\alpha + \sin\varphi\sin\alpha)\cdot\sin\tau\sin\sigma + \cos\tau\cos\sigma \\ &= \cos(\varphi-\alpha)\cdot\sin\tau\sin\sigma + \cos\tau\cos\sigma \end{aligned}$$

Therefore,

$$I = a\cdot\cos(\varphi-\alpha) + b \tag{2.6}$$

where

$$\begin{cases} a &= \sin\tau\sin\sigma \\ b &= \cos\tau\cos\sigma \end{cases} \tag{2.7}$$

Our principal objective is to formulate a model which is not dependent on light (or, rather, which takes it fully into account). The simple idea is to acknowledge that it is indeed quite hard to fully bypass the light conditions when performing recognition on such an object. Therefore we choose to take *all* possible incoming light into account. Given the nature of the object, a postulate is made that the information is located in the relief of the object. More precisely, it makes sense to consider that the relief variations contain relevant data for analyzing such surfaces. Following this logic, it is desirable to incorporate a criterion of distinction between flat regions and large relief variations

in our model. Since the object is quasi-flat, large variations appear as steep slopes and constant regions as flat areas. The energy received by a steep ridge within the surface is the largest when impacting from a right angle. This justifies that the angle τ be the closest to $\frac{\pi}{2}$ as possible. For the remaining equations though, τ will still be represented as a variable.

The azimuth φ remains the key illumination variable. There is no a priori reason to choose one or another and it certainly may differ from one acquisition to another since coins especially may not feature an inherent or even obvious orientation. Consequently we chose *all* orientations. A natural way to finally solve this issue is via an integration over $\varphi \in [0, 2\pi)$. However, we argue that it is not the finest solution and prefer another approach. This idea of azimuth summation will nonetheless be discussed later (see Section 2.2.5.4).

The key argument here is to consider what impact the variation of φ has over the image. To illustrate this insight, Figure 2.6 features an example of a coin taken under two different light orientations, separated by an angle of $\frac{\pi}{4}$. Figure 2.6c displays the absolute difference between them. One can see that the crown, for instance, is differently lit in Figure 2.6a (no bright spot) than in Figure 2.6b (salient edges). This translates into brighter regions in Figure 2.6c. The same remark can be made for other parts such as the forehead or some letters. It is not advisable to integrate over such variations, since the outcome is zero. Instead, we integrate over the squared differences, which accentuates the contrast and pinpoints the aforementioned regions (Figure 2.6d). Consider the normalized integral $\mathcal{J}$:

$$\mathcal{J} = \frac{1}{2\pi} \int_0^{2\pi} \left(\frac{\partial I}{\partial \varphi} \right)^2 \mathrm{d}\varphi \tag{2.8}$$

Using Equation (2.6), we derive:

$$\frac{\partial I}{\partial \varphi} = a \cdot \sin(\alpha - \varphi) \tag{2.9}$$

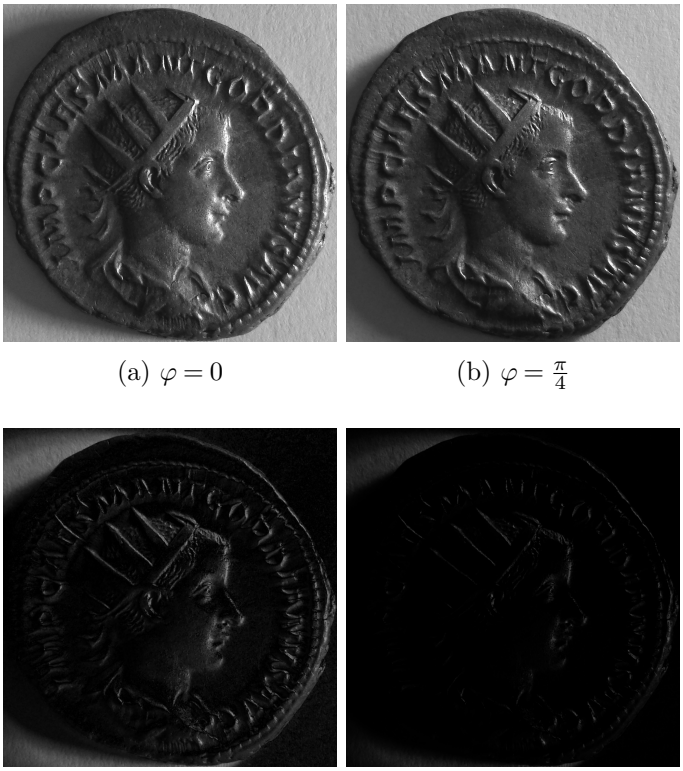
Thus,

$$\begin{aligned} \mathcal{J} &= \frac{a^2}{2\pi} \underbrace{\int_0^{2\pi} \sin^2(\alpha - \varphi) \mathrm{d}\varphi}_{= \pi} \\ &= \frac{a^2}{2} \end{aligned}$$

By substituting a for Equation (2.7), we obtain:

$$\mathcal{J} = \frac{1}{2} \sin^2 \tau \sin^2 \sigma \tag{2.10}$$

Interestingly, the result has a symmetrical dependency on τ and σ. Similarly there is a symmetry in Equation (2.9) between the light azimuth φ and the normal azimuth α: one could interchangeably integrate over φ or α. This simply corresponds to the fact



(a) $\varphi = 0$ (b) $\varphi = \frac{\pi}{4}$

(c) Image difference (d) Squared image difference

Figure 2.6: A simple intuition: two different light azimuths make "shine" different parts of the coin. Calculating their difference reveals where large irradiance variations occur in white.

that turning the light around the object is the same as turning the object while keeping the light still.

Now, we focus on the impact on light direction, so the object orientation is considered fixed. We can therefore consider $\mathcal{J}$ as a function of the normal variables. According to Equation (2.10), our integral $\mathcal{J}$ depends on the normal colatitude σ. The closer σ is to $\frac{\pi}{2}$, the greater $\mathcal{J}$ becomes, hence $\mathcal{J}$ is a mathematical testimony of high variations of the surface of the object. Let $\vec{n} = (n_x, n_y, n_z)$ be in Cartesian coordinates, we notice:

$$\begin{aligned} sin^2\sigma &= 1 - cos^2\sigma \\ &= 1 - n_z^2 \\ &= n_x^2 + n_y^2 \\ &= \|\vec{n}_{xy}\|^2 = g^2 \end{aligned} \tag{2.11}$$

The variable $\|\vec{n}_{xy}\|$, which we denote g, is the norm of the projection of the normal vector onto the (xOy) plane. This value is small when the relief is flat (the normal vector points upward) and large where the relief is sloping (the normal vector points sideways). Thus $\mathcal{J}$ becomes:

$$\forall (x,y) \in U, \quad \mathcal{J}(x,y) = \frac{1}{2} \cdot \sin^2\tau \cdot g(x,y)^2 \tag{2.12}$$

Equation (2.12) connects the integral $\mathcal{J}$ to the amplitude of relief variations at point (x,y). The constant factor depends only on the light colatitude τ, which was set to $\frac{\pi}{2}$. Using a lesser value for τ would in theory only dim the resulting $\mathcal{J}$. With the intention of retrieving g, we define:

$$\mathcal{E} = \sqrt{\mathcal{J}} \tag{2.13}$$

Since all values are positive,

$$\boxed{\forall (x,y) \in U, \quad \mathcal{E}(x,y) = \frac{\sqrt{2}}{2} \cdot \sin\tau \cdot g(x,y)} \tag{2.14}$$

We call the newly defined $\mathcal{E}$ the multi-light energy map of the object. The name was chosen in reference to two details. Firstly, it physically refers to a sum of irradiance per unit of angle across all light azimuths, which amount to a form of energy received by the object. The second reason stems from the signal processing definition of energy, to which Equation (2.8) is akin. Another intriguing fact is that Equation (2.8) may be seen as a variance (up to a constant multiplier $\frac{1}{2\pi}$), provided that $\frac{\partial I}{\partial \varphi}$ is considered as a random variable. This observation again leads to see $\mathcal{E}$ as a indicator of high relief variations, which g embodies. Moreover, Equation (2.14) demonstrates that the values of g are available from a simple computation and that calculating $\mathcal{E}$ bypasses the normal map.

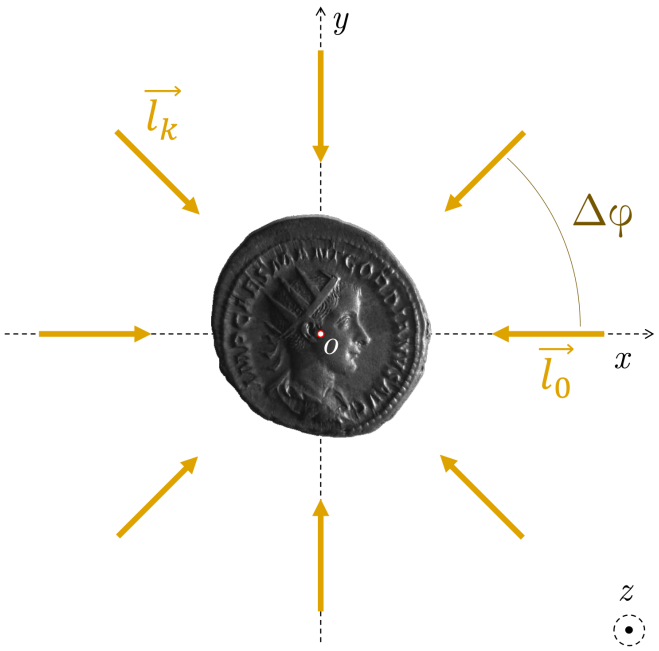


Figure 2.7: Acquisition device. Lighting is seen from above, in the (xOy) plane. With $n_l = 8$, the angular shift $\Delta\varphi$ is $\frac{\pi}{4}$.

2.2.2 Acquisition and computation

In order to be able to properly construct the energy map $\mathcal{E}$, we need to take the acquisition into consideration. It is not possible to use an infinite number of lights and discretizing too coarsely may impact the quality of the final energy map. Consider the schemas in Figures 2.5 and 2.7. The object is placed at point O and n_l light sources $\vec{l_k}, k \in \{0, ..., n_l - 1\}$ are arranged in a circle, equally spaced around the object. The angular shift between each light source is $\Delta\varphi = \frac{2\pi}{n_l}$. The camera is placed above the origin O, facing the object along axis z. Given the discretization of the angles into $\varphi_k = k\Delta\varphi$, by using Equations (2.6) and (2.7),

$$\begin{aligned} \forall k \in \{0, ..., n_l - 1\}, \quad \Delta I_k &= I_{\xi(k+1)} - I_k \\ &= a \cdot (\cos(\varphi_{k+1} - \alpha) - \cos(\varphi_k - \alpha)) \\ &= -2a \cdot \sin\left(\frac{\pi}{n_l}\right) \cdot \sin\left(\frac{(2k+1)\pi}{n_l} - \alpha\right) \end{aligned}$$

with

$$\xi(k) = \begin{cases} 0 & if \quad k = n_l - 1 \\ k & else \end{cases}$$

Therefore,

$$
\begin{aligned}
\mathcal{J}_{n_l} &= \frac{1}{n_l} \cdot \sum_{k=0}^{n_l-1} \left(\frac{\Delta I_k}{\Delta \varphi} \right)^2 \\
&= \frac{1}{n_l \Delta \varphi^2} \cdot 4a^2 \sin^2\left(\frac{\pi}{n_l}\right) \cdot \underbrace{\sum_{k=0}^{n_l-1} \sin^2\left(\frac{(2k+1)\pi}{n_l} - \alpha\right)}_{=\frac{n_l}{2}} \\
&= \frac{4a^2}{n_l \left(\frac{2\pi}{n_l}\right)^2} \cdot \sin^2\left(\frac{\pi}{n_l}\right) \\
&= \frac{a^2}{2} \cdot \left(\frac{\sin\left(\frac{\pi}{n_l}\right)}{\frac{\pi}{n_l}} \right)^2 \\
&= \frac{a^2}{2} \cdot \operatorname{sinc}^2\left(\frac{\pi}{n_l}\right)
\end{aligned}
$$

The function $\operatorname{sinc} : u \mapsto \frac{\sin u}{u}$ is the unnormalized sinc function. By considering the same definition as in Equation (2.13), replacing a by its definition (Equation (2.7)) and making the same remark regarding the positivity of all the concerned variables, we get:

$$
\boxed{\forall (x,y) \in U, \quad \mathcal{E}_{n_l}(x,y) = \frac{\sqrt{2}}{2} \cdot \operatorname{sinc}\left(\frac{\pi}{n_l}\right) \cdot \sin\tau \cdot g(x,y)} \tag{2.15}
$$

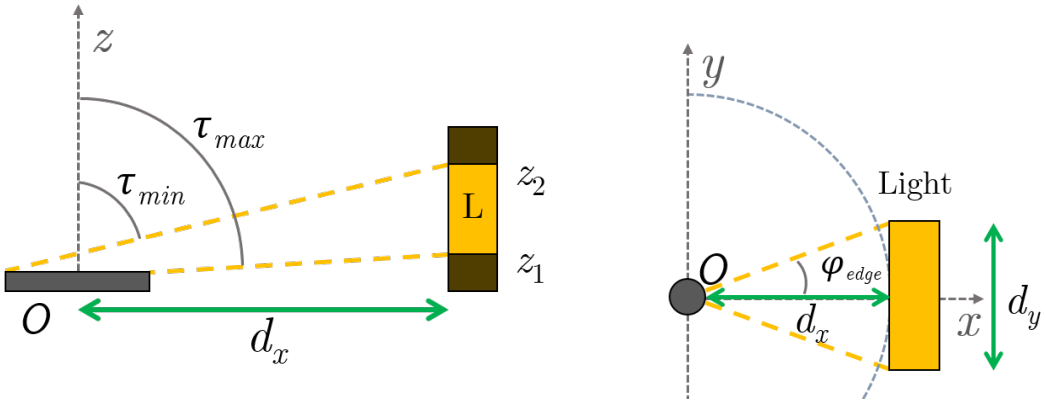
Of course, we notice a clear similarity between Equations (2.14) and (2.15), one being the limit of the other as the number of light n_l tends to infinity:

$$
\lim_{n_l \to \infty} \mathcal{E}_{n_l} = \mathcal{E} \tag{2.16}
$$

It is therefore possible to approach the theoretical formulation of the energy map by using a limited number of lights. A study of the influence of this parameter is presented in Section 2.5.1. Although Equation (2.15) is an exact equation linking the energy map and the gradient norm in the transverse plane, one may argue that every calculation will always lead to the same multiplicative factors. Therefore the energy map can be computed as:

$$
\boxed{\widehat{\mathcal{E}}_{n_l}(x,y) = \sqrt{\sum_{k=0}^{n_l-1} (\Delta I_k)^2}} \tag{2.17}
$$

The acquisition was performed using a smartphone (iPhone 6S) and a single light source composed of several white LEDs illuminates the object. The light source is therefore quite wide, which means that the light covers a range of angles (for instance from τ_{min} to τ_{max} in vertical direction). In real life, occlusions play a major role by



(a) Sagittal plane (b) Transversal plane

Figure 2.8: Specifications of the acquisition setup. A light source L of width d_y and applicate z between z_1 and z_2 illuminates the object O from a distance d_x.

Orientation	d_x	d_y	z_1	z_2	φ_{edge}	τ_{min}	τ_{max}
Horizontal	25 cm	4 cm	1.5 cm	4 cm	4.57°	80.9°	86.6°
Vertical	25 cm	2.5 cm	3.5 cm	8 cm	2.86°	72.3°	82.0°

Table 2.1: Measures for the acquisition setup for two positionings of the light source.

casting shadows and rendering certain points completely absent from the energy map because they are not seen by any image. This aspect in not taken into account in the previous calculations. A wide source is useful since it covers a wider angular range and allows for the former shady points to be lit (see Figure 2.8a). Details about the setup geometry can be found in Figure 2.8 and the corresponding values are gathered in Table 2.1. Casting many rays at different angles may however have a tendency to blur the details because the outcome of Equation (2.5) will actually be the sum of the contribution of each normal lit by the various light rays. Note that the azimuth breadth φ_{edge} differ less significantly between the two positions than do the angles τ_{min} and τ_{max}.

The whole process is summarized in Figure 2.10 and a detailled example is displayed in Figure 2.9. This is a fine example since the coin is well preserved and features many details. The energy map is rather clean, showing the principal high variation areas in the coin: the crown, the face, the hair texture, the clothing and the letters. Whiter parts are synonym of steep slopes and blacker parts are flat zones. Of course, depending on the quality of the object, the energy map may not be as clean.

Figure 2.11 shows eight different examples of results along with on image of the corresponding coin. Figure 2.11a, 2.11b and 2.11h are also very clean energy maps, with a clear distinction between flat areas and relief edges. Because erosion essentially alters the depth of the object's relief, it consequently reduces the contrast in the energy maps. Figure 2.11c, 2.11d and 2.11g illustrate this tendency: these coins present soft edges which may not therefore be obvious in the energy map (this happens mostly because

Figure 2.9: Energy map of a coin.

there is a high value area in the energy map which flattens the rest of the dynamic range). Another noticeable phenomenon happens when there are soft slopes purposely engraved as such, which translates in grey areas in the energy map. Some example of this are shown in Figure 2.11e and 2.11f. These semi-flat areas may not be the most informative part of the coin, and may disrupt future detection and recognition algorithm since there is really no distinctive characteristics to be derived from them.

2.2.3 Segmentation

One can notice some light artifacts around the coin area in the energy map in Figure 2.9. This is due to the projections of the coin shadow onto the support. Since only the coin

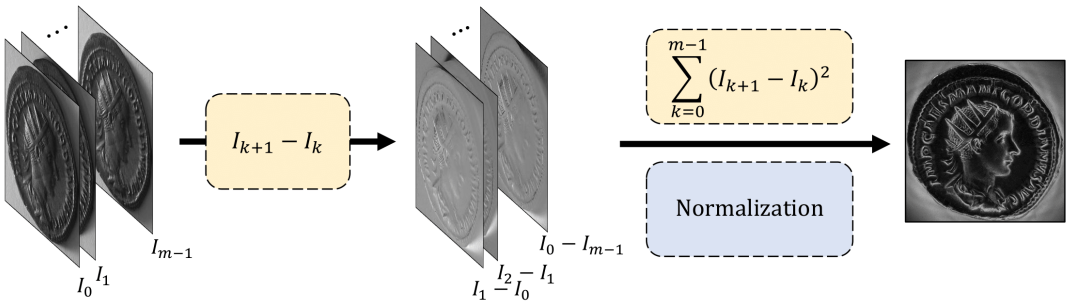
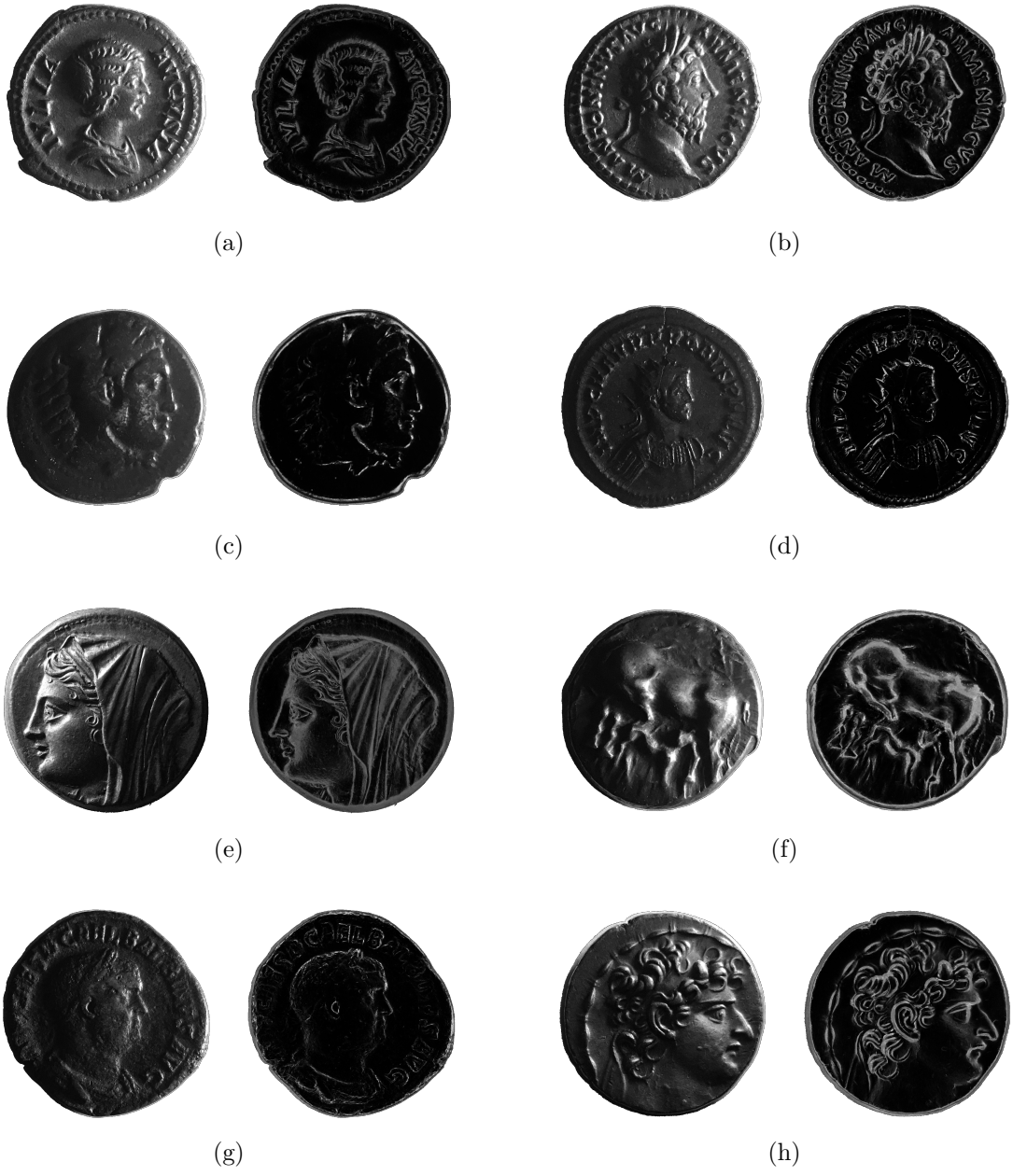


Figure 2.10: Pipeline of the energy map computation.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 2.11: Eight examples of coin images and their energy map.

area is needed, it has to be segmented. To perform the segmentation, one may rely on several methods, including color segmentation if the support color is distinctive enough, active contour models or ellipse fitting models.

Our choice is to use the segmentation technique described by Zambanini et al. [109]. The main principle is to use the information in the image as a clue for the presence of the object. Two filters are applied to the image. The first one is an entropy filter, which calculates the local entropy of a small image region around each pixel. The second is a range filter, which evaluates the local grey value range in the image. Those two resulting filtered images are combined and thresholded. The threshold parameter is automatically guessed by maximizing the circularity of the resulting blob.

The results are convincing enough to be used as is given that the energy map is obtained via controlled illumination. Another similar tactic is to get a photograph of the image with $\tau \approx 0$ (see Section 2.2.5.3). This is equivalent to using the flash of a smartphone for example. This produces an image without shadows around the coin, which facilitates a good segmentation. This type of segmentation has been used in Figure 2.11.

2.2.4 Orientation map

Besides being able to recover the magnitude of the object variations in the (xOy) plane, one can also easily recover the angles. Since the collections of images is by itself a consequence of a varying light azimuth at $\tau \approx \frac{\pi}{2}$, it also translates into the orientation of the relief edges: edges facing the light shine brighter.

Given the collection of images $\{I_k\}_{|k\in\{0,\dots,n_l-1\}}$, the orientation map is calculated as the mean angular value weighted by I_k.

$$\forall (x,y) \in U, \quad \mathcal{A}(x,y) = \arctan\left(\frac{\sum_{k=0}^{n_l-1} I_k(x,y)\cdot\sin(\varphi_k)}{\sum_{k=0}^{n_l-1} I_k(x,y)\cdot\cos(\varphi_k)}\right) \tag{2.18}$$

The result of such calculation is displayed in Figure 2.12, which represents the orientation map corresponding the energy map in Figure 2.9.

The tools and models described so far are the core of the energy map model. Not only the energy map can be used by itself but the orientation maps is readily available by using the set of captured images. In the next sections, we will explore the possible connections with other models. As will be discussed, all the models described before and the energy map are closely related, but the energy map remains the fastest way to obtain the relief information.

2.2.5 Links with other models

Several models are compared to the energy map, including the image model, the normal map and the height map. We also provide an insight of other potential model candidates which are not standard and we will show that the energy map still holds as the best tool to describe the relief.

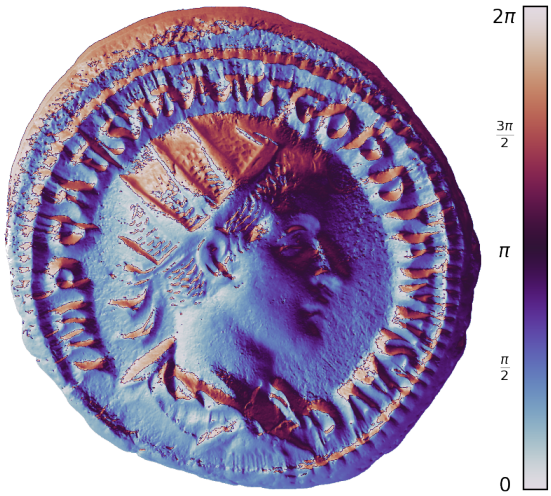


Figure 2.12: Orientation map.

2.2.5.1 Images

Images are what the energy map model stems from. The connection can therefore be immediately made. Moreover, it is also possible to retrieve a new image out of the energy map and the orientation map. One can for instance project both of them back to Cartesian coordinates, which then may be used as a basis for any new image.

$$\begin{cases} \mathcal{E}_x &= \mathcal{E} \cdot \cos \mathcal{A} \\ \mathcal{E}_y &= \mathcal{E} \cdot \sin \mathcal{A} \end{cases} \tag{2.19}$$

One can choose λ and μ such that $\lambda^2 + \mu^2 = 1$. Using Equation (2.19), I can be calculated as:

$$\forall (x,y) \in U, \quad I(x,y) = \lambda \cdot \mathcal{E}_x(x,y) + \mu \cdot \mathcal{E}_y(x,y) \tag{2.20}$$

Of course, the azimuth φ is implicitly chosen here, since it follows naturally from the normalization that $\lambda = \cos\varphi$ and $\mu = \sin\varphi$. The energy map Cartesian components form a natural basis for the image.

2.2.5.2 Normals and height map

Equation (2.20) is just another formulation of Equation (2.6) in Cartesian coordinates. Naturally, $\mathcal{E}_x$ and $\mathcal{E}_y$ are directly linked the two first normal components n_x and n_y. Since normals are normalized vectors, this leads to:

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \frac{1}{\sqrt{1 + \mathcal{E}_x^2 + \mathcal{E}_y^2}} \cdot \begin{pmatrix} \mathcal{E}_x \\ \mathcal{E}_y \\ 1 \end{pmatrix} = \frac{1}{\sqrt{1 + \mathcal{E}^2}} \cdot \begin{pmatrix} \mathcal{E}_x \\ \mathcal{E}_y \\ 1 \end{pmatrix} \tag{2.21}$$

(a) Energy map. (b) Inverted $\check{I}$. (c) Gradient magnitude g.

Figure 2.13: Comparison between $\mathcal{E}$ and $\check{I}$.

By definition, the height is related to normals by $\frac{\partial z}{\partial x} = n_x$ and $\frac{\partial z}{\partial y} = n_y$, from which we can infer:

$$\|\vec{\nabla} z\| = \frac{\mathcal{E}}{\sqrt{1+\mathcal{E}^2}} \tag{2.22}$$

Considering the normalization (shown in Figure 2.10) and the final computed model (Equation (2.17)), Equation (2.22) becomes

$$\|\vec{\nabla} z\| \equiv \mathcal{E} \tag{2.23}$$

which leads to the same conclusions as above regarding the nature of $\mathcal{E}$.

2.2.5.3 Light from above

It has been discussed earlier that using one image taken with one light source placed at $\tau = 0$ is convenient for segmentation. It is compelling to understand that this image produces a result akin to the energy map itself. Indeed, it assumes that light rays coming from above will be reflected the most by flat surfaces and that steep slopes will render black pixels. Using Equation (2.5),

$$\check{I} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} n_x & n_y & n_z \end{pmatrix} = n_z$$

In the same fashion as what was done in Equation (2.11), the equation becomes:

$$\forall (x,y) \in U, \quad \check{I}(x,y) = \sqrt{1 - g(x,y)^2} \tag{2.24}$$

It can be interesting to compare $\check{I}$ and $\mathcal{E}$ in order to visually assess their quality. Figure 2.13 compares the energy map of a coin and two versions of the inverted image lit from above: Figure 2.13b is simply the "negative" of the image lit at $\tau = 0$ and Figure

2.13c shows g as the reciprocal version of Equation (2.24). Although Figure 2.13b could be a good candidate for a model, it appears to feature a coarse and low quality depiction of the object contrarily to the energy map in Figure 2.13a. The rendering of g clearly lacks precision in order to be used as a model of the coin. This whole calculation is always under the assumption that the object is a perfect diffuser, but the conditions with which $\check{I}$ is taken seriously violate this assumption. Coins remain reflective objects and are subject to stray specular highlights in the field of the coin, which compromise the quality of the final model.

2.2.5.4 Summing the images

As explained in the introduction, when it comes to photographing ancient coins, it is advisable to use not one but several light sources placed at various azimuths. The main idea is to make the best rendering of the coin, making it visually identifiable and artistically beautiful. Given the proclivity of the human brain to attach importance to the texture variations such as edges and contours in the image it perceives, should it make sense to consider a sum of images as being more visually relevant than a sole image?

Let us take again the previous assumption (Equation (2.6)) and calculate the following:

$$\begin{aligned} \mathrm{S} &= \frac{1}{2\pi}\int_0^{2\pi} I(\varphi)\,\mathrm{d}\varphi \\ &= \frac{a}{2\pi}\int_0^{2\pi} \cos(\varphi-\alpha)\,\mathrm{d}\varphi + \frac{b}{2\pi}\int_0^{2\pi} \mathrm{d}\varphi \end{aligned} \tag{2.25}$$

For this calculation, it is essential to analyze the behavior of this image model. I, as defined in Equation (2.6), may reach negative values (in the case $\tau \approx \frac{\pi}{2}$ for instance). Although this is not a too bad approximation for most calculations (for example, Equation (2.10) is not impacted since it uses a squared sine function), it does in this case lead to false results in practice. Integrating over the cosine produces 0, and this is a direct consequence of the image formula. In practice, this would mean that we consider that the irradiance on a small surface hit by a light ray is the negative of the irradiance produce by a ray coming from the opposite direction on the same surface.

Two choices may be applied here: either considering that both sides produce positive irradiance, or considering that one side is never hit. The latter makes more sense in the case of a quasi-flat object and therefore the integration is performed over the positive values:

$$\begin{aligned}\mathrm{S} &= \frac{a}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos(\varphi-\alpha)\,\mathrm{d}\varphi + \frac{b}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \mathrm{d}\varphi \\ &= \frac{2a}{\pi} + b \\ &= \frac{2g}{\pi}\cdot\sin\tau + \sqrt{1-g^2}\cdot\cos\tau\end{aligned}$$

Relying on Equations (2.14) and (2.24), we can formulate a linear relationship such that:

$$\forall (x,y)\in U,\quad \mathrm{S}(x,y) = \frac{2\sqrt{2}}{\pi}\cdot\mathcal{E}(x,y) + \breve{I}(x,y)\cdot\cos\tau \tag{2.26}$$

Thus the outcome is a connection between the sum of images, the variance of image differences per azimuth and the image lit from above. It simply shows that, depending on the value of τ, one can give more or less weight to the flat areas of the object. At $\tau = \frac{\pi}{2}$, one gets in theory $\mathcal{E}$.

However, in practice the results are not satisfactory. In Figure 2.14 are represented the three models linked via Equation (2.26). Apart from Figure 2.14c, the colatitude τ is set to be close to $\frac{\pi}{2}$. Overall, the sum S tends indeed to underline the large surface variations, as one can easily notice white areas over the letters, the face or the hair. Even so, the result is much less contrasted than the energy map.

There are two major issues regarding S, both have to do with the use of the integral and are problems solved by the energy map. The first problem is the accumulation of noise in the final image. The image presented here has a resolution of 1024×1024 and each image composing it has been smoothed by a Gaussian filter of variance 2 pixels. Without this pre-processing, the sum remains extremely noisy and unexploitable. The other issue is that, in practice, there is also an additive component of ambient light, which remains constant for any φ. While the sum of images gather both noise and ambient light, the energy map gets rid of them by differentiation. Noise is statistically reduced to 0 and the ambient constant disappears.

2.2.6 Specular case

To conclude this presentation of the energy map, this subsection gives some hints about the influence of specularity on the energy map model. So far, the Lambertian hypothesis has driven all the calculations, but what if a specular component is incorporated in the model? Of course, there are a lot of BRDF for simulating metals, but in order to keep a simple approach, we use the Phong reflection model [80], described in Equation (2.27), assuming there is only one light source.

$$I = k_{am}I_{am} + k_{di}\vec{l}\cdot\vec{n} + k_{sp}(\vec{r}\cdot\vec{v})^s \tag{2.27}$$

The subscripts am, di and sp stand for ambient, diffuse and specular. The vector $\vec{v}$ is the viewing direction, which is $(0,0,1)$ in our case, and $\vec{r} = 2(\vec{l}\cdot\vec{n})\vec{n} - \vec{l}$ is the direction

(a) Energy map. (b) Sum of images. (c) Lit from above.

Figure 2.14: Comparison between $\mathcal{E}$, S and $\breve{I}$.

of perfect reflection. The parameter s controls the width of specular highlights. To simplify even further, it is assumed to be 1. One can notice the Lambertian model in the equation, so the integral $\mathcal{J}_{\text{phong}}$ in Equation (2.28) is a linear combination of squared energy maps.

$$\mathcal{J}_{\text{phong}} = k_{di} \cdot \mathcal{J}_{di} + k_{sp} \cdot \mathcal{J}_{sp} \tag{2.28}$$

$\mathcal{J}_{di}$ is the diffuse part expressed in Equation (2.12). Now consider the specular image I_{sp}:

$$\begin{aligned} I_{sp} &= \vec{r} \cdot \vec{v} \\ &= 2 \cdot I_{di} \cdot n_z - \cos\tau \end{aligned} \tag{2.29}$$

Then,

$$\mathcal{J}_{sp} = 4 \cdot n_z^2 \cdot \mathcal{J}_{di} \tag{2.30}$$

The specular version of the energy map is therefore a function of the diffuse energy map. To finish, using Equation (2.12) and using the fact that $n_z = \sqrt{1-g^2}$,

$$\boxed{\forall (x,y) \in U, \quad \mathcal{J}_{sp}(x,y) = 2 \cdot g(x,y)^2 \cdot (1 - g(x,y)^2) \cdot \sin^2\tau} \tag{2.31}$$

The function $u \mapsto u(1-u)$ has a maximum for $u = \frac{1}{2}$, which means that Equation (2.31) reaches its maximum when $g = \sin\sigma = \frac{\sqrt{2}}{2}$, thereby confirming that the brightest highlights happen when $\sigma = \frac{\pi}{4}$.

A summary of diffuse and specular highlights is displayed in Figure 2.15. In any area presenting a high relief variation, there has to be a step slope surrounded by two almost flat areas. So there is a location at which the diffuse highlight is maximum and two locations at which the specular highlighs are maximal. Specular highlights on high relief variations produce a splitting in two edge lines whereas the diffuse highlights produce one central line.

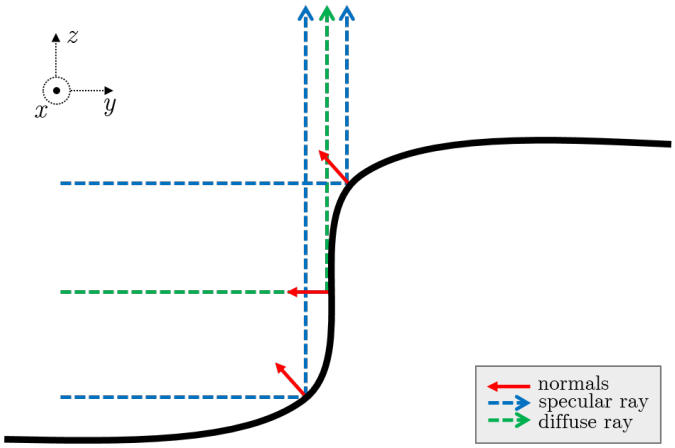


Figure 2.15: Diffuse and specular highlights on a steep slope.

2.2.7 Conclusions on the energy map model

The energy map is a light, easy-to-compute and informative model of any quasi-flat object. It reveals the variations of the object's surface, making appear steep slopes whiter and flat regions darker. The acquisition is straightforward and uses several light azimuths to capture the images. The sequence of images can be used to calculate the orientation map, which represent the angles of the slopes in the object's relief. The energy map is itself quite close to many other models, which partly comes from the fact that all are dependent on the Lambertian assumption.

Now that we have established the theoretical aspect of the model, we introduce two datasets: one containing real coins and another one containing artifial coins, both of which will turn out to be very useful for the experiments. The next two sections presents these datasets, their construction

2.3 Real coins dataset

In this section, we present a dataset of 23 coins. This dataset will mainly be used in experiments in Chapters 3 and 4 in order to visually assess the quality of some algorithms. The coins are introduced in Figure 2.16, which shows both obverse and reverse sides. The major asset of this database is that it contains a variety of coins which could be clustered into classes and which contains coins struck with the same die. Coins from the same die have been highlighted with the same color. Additional information about the coins can be found in Appendix A.

One die is used for one side and both sides of a coin may not have been struck by the same pair of dies. For instance, the blue highlight in Figure 2.16a does not have a reverse counterpart in Figure 2.16b. We can therefore consider that the dataset has 46 coins, composed of some obverses and some reverses, which extends our database.

Figures 2.16a and 2.16b show one picture per coin. Of course, the acquisition is done on each coin (obverse and reverse) according to the process described in Section 2.2.2 so as to be able to compute the energy map and the orientation map.

The main drawback of this dataset is that it does not contain enough elements to derive statistically relevant results. The difficulty of acquiring a larger dataset comes from the fact that it is necessary to have access to the objects themselves – and not mere images of it – so that we can perform a proper acquisition. The places where large amounts of coins can be found are most likely unwilling to authorize direct contact with the objects, even more so if no functional and portable acquisition device is available. Furthermore, it is even more complex to find groups of coins from the same die; such sets of coins are very valuable for our research but unfortunately quite rare. For all these reasons, we decided to take a different approach to obtain data. Namely, instead of acquiring them, we propose to computationally generate them via a series of processes described in the next section.

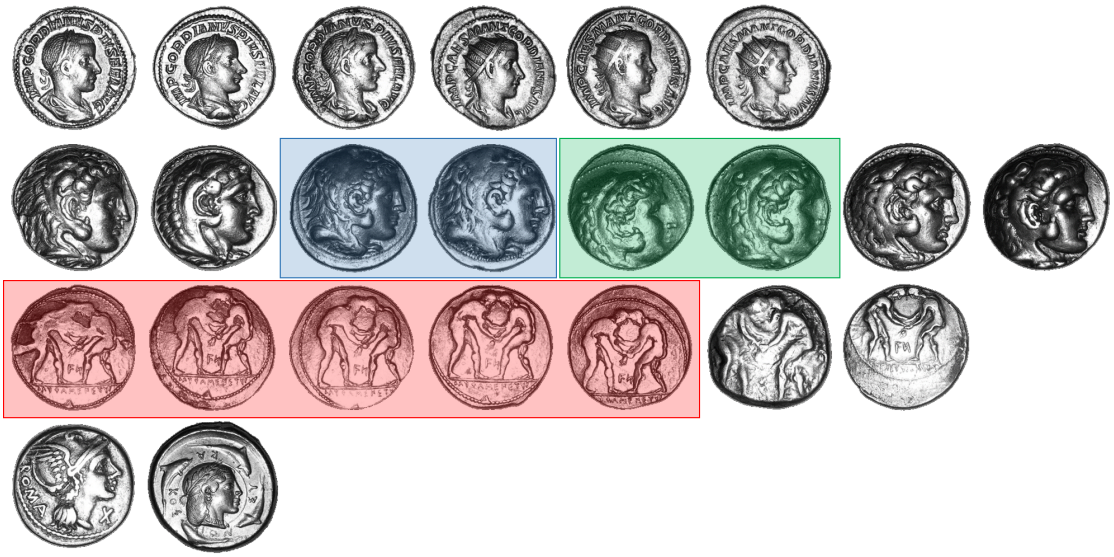
2.4 Artificial coins dataset

An important issue regarding the field of classification and recognition is the choice of an appropriate dataset for measurements and tests. Coins are suitable candidates to analyze quasi-flat objects since they are numerous and present some challenging difficulties. However, the main problem with coin datasets is that the ones already acquired are only image-based datasets. This is understandable since the image is the primary focus of computer vision. Yet it is paramount for us that we access the object itself, so that many captures can be taken in order to build the energy map. Some museum would be willing to help in that regard but because these models and tools are somewhat new, and without a suitable acquisition device, it does not seems reasonable to bet on acquiring the dataset before having some algorithms ready.

In order to perform computations, build algorithms and collect results, the idea has been to use artificial coins instead of real ones. This enables the creation of a suitable dataset and test objects close to theoretical models to begin with. However, the artificial objects have to remain coherent with the topology of quasi-flat objects and feature relevant details in accord with real objects. This artificial dataset will be used in many experiments starting in this chapter and in the next chapters.

2.4.1 Specifications and process

The process of creating artificial coins tries to emulate the one used for ancient coins: a die, carved by an artist, is struck against a metal flan on which it marks its relief. A single die is used a certain amount of times, wears out and is replaced by a new one. This implementation yields several types of coins, for each of which several dies have been used.



(a) Obverse of the dataset.

(b) Reverse of the dataset.

Figure 2.16: Dataset of real coins. Obverses and reverses are placed at the same position in the figures. The organization of the display is made so that coins which "look like" each other have been grouped together, except the two coins in the bottom-left corner which stand apart from every coin. Coins with the same color come from the same die. More information are listed in Appendix A.

Figure 2.17: Examples of small symbols for the engraving.

2.4.1.1 Metal flan

The first step is to create a metal flan which is the support for the relief. The flan is simulated via a height map whose contour is randomly generated since an ancient coin is never perfectly round. Therefore the flan shape is calculated using the following protocol:

- Use a circle of fixed radius as a baseline.
- Select equally spaced points (around 10 is fine) in the circle and add a random value to each coordinate.
- Interpolate the knots with a cubic spline.
- Fill the interior of the shape.

The radius of the coin (the initial disk) is chosen to be 512 pixels. The height map itself is of size 1024×1024 px^2. The height of the flan is set to be 500 pixels. After generating the flan shape, a Gaussian filter of standard deviation 5 is applied in order to smooth the borders, making them more realistic.

2.4.1.2 Die and engraving

The die is also a metal piece, the shape of which is also random. The baseline is thus a disk of radius 466 pixels (the die is slightly smaller than the flan, so that the entire die engraving can fit inside of it). The most crucial part is the design of the engraving. In order to simulate it, we select a set of randomly selected symbols, which will be artificially carved on the die. If the symbol S is engraved in a die D, then the die becomes $D \leftarrow D - \frac{1}{2}S$. There are of course lots of ways to create the symbols. The chosen approach is to consider large symbols and small symbols. One large symbol is always present at the center of the die, and small symbols are placed around it, much like

Figure 2.18: Examples of large symbols for the engraving.

letters would be engraved around a roman emperor's bust. Examples of such symbols can be found in Figure 2.17 and 2.18. They are represented as negative height maps, which are thus added to the die.

2.4.1.3 Minting and wearing

This step involves using the die to mint the coin. The positioning of the flan with respect to the die cannot be rigorously the same for each coin, and so a random shift parameter decides the amount of decentering. The shift is set between -50 px and 50 px for each axis. The die will be used for a limited amount of iterations and, with each strike, the die wears out. So, after each strike, a Gaussian filter of standard deviation 2 is applied to the die. The flan F becomes $F \leftarrow F - \frac{1}{2}D$.

After that, we simulate the erosion by smoothing parts of the coin. Namely, a random position is chosen within the flan and a Gaussian filter is applied within an circular area of radius 400 px. The filter has a standard deviation that depends on the distance from the center of the area so that the strongest smoothing occurs at the center and the weakest happens at the borders. This allow for the production of a nicely distributed erosion, without any discontinuities.

The last step is to apply a texture and noise to the flan. The texture is meant to embody the various deteriorations the coin can go through. A texture is randomly chosen among four presented in Figure 2.19 and is also randomly rotated and scaled. The result is added to the flan F with a strength coefficient. Those coefficients have been chosen empirically to be 0.05, 0.02, 0.02 and 0.0075 in the order of Figure 2.19. Finally, another noisy layer is superimposed by adding a normal noise smoothed by a Gaussian filter of standard deviation 13.

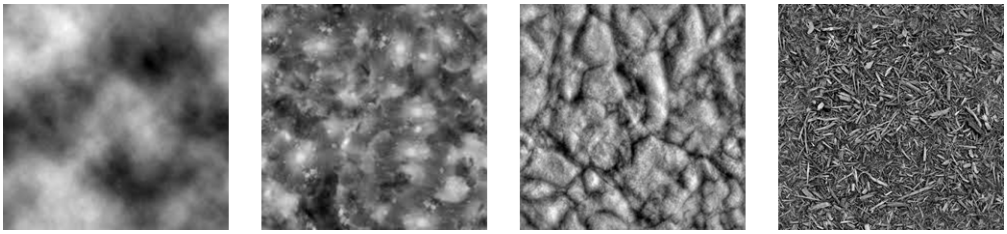
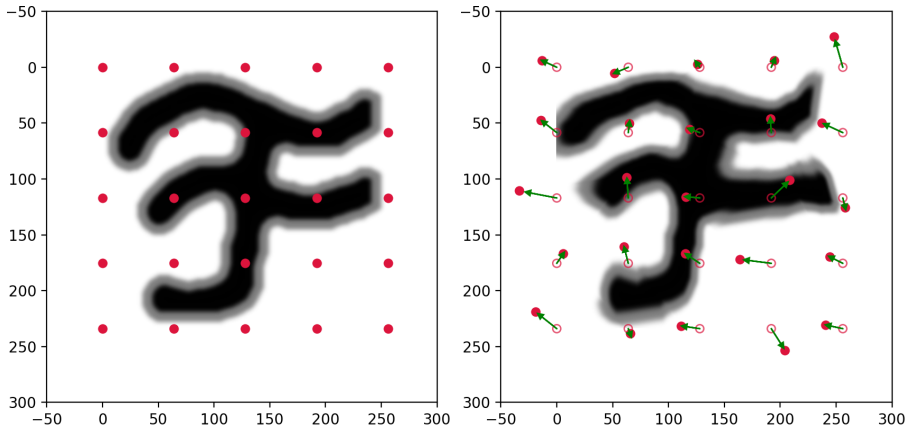


Figure 2.19: Textures added to the flan to simulate small deteriorations.



(a) Original symbol. (b) Deformed symbol.

Figure 2.20: Symbol deformation via randomized piecewise affine transformation. The green arrows in (b) show the displacement of the control points.

2.4.1.4 Replacing the die

When the die is too deteriorated, it is replaced by a new one representing the same set of symbols. In reality, there is of course no way to reproduce the exact same engraving since it was crafted by an artist.

In order to emulate this, the exact same symbols are used and, for each of them, a random piecewise affine transformation is applied (see Figure 2.20). A grid of regularly spaced points is set across the 2D lattice containing the symbol (Figure 2.20a). Then, the same grid is modified so that each point undergoes a small random shift from its original position. The intensity of this shift is chosen to sufficiently deform the symbols while keeping the difference realistic. This intensity multiplies a normal random variable, which gives the displacement of each point along the x and y axes. The transformation between both sets of control points is then modeled as a piecewise affine transform, which is finally applied to the symbol (Figure 2.20b) itself.

2.4.2 Results

Using the entire protocol described above, a series of 16 types of coins is generated. These types are displayed in Figure 2.21. Each type is different and is represented essentially with a large symbol at the center with or without a ring of small symbols. There is no distinction between obverse and reverse. For each type, the process designs 5 dies and 5 coins are made out of each die.

Various dies of one type are shown in Figure 2.22. As one can note, the difference between each die can be subtle but nonetheless always sufficiently significant so as to clearly visually distinguish them. The result respects the usual proximity between coins from the same types but from different dies, which translates into a low inter-die variance.

Each time a die is used, it wears out a bit more. For example, see in Figure 2.23, how the fine details of the first coin disappear and only the coarse ones remain. Each coin stemming from the same die will be geometrically close but will nonetheless present some dissimiliarities, mostly due to wear and erosion.

The resulting coins are constructed as height maps. In order to obtain the energy map, artificial photographs have to be taken. A series of images is thus computed via the use of normals and a Lambertian model of reflectance.[1] The height map h is normalized between 0 and 300, which becomes the maximum height of the coin. Normals are calculated with Equation (2.21), replacing $\mathcal{E}_x$ and $\mathcal{E}_y$ by $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$. Figure 2.24 displays the results of the energy map and orientation map calculation.

To conclude, the artificial coin dataset is designed to follow the logic of the making of ancient coins. It enables us to render as many coins as desired, with various symbols and textures. The main advantage of this dataset is that it comprises several coins per die and various dies per type; this is convenient when one wants to study the recognition of coins originating from the same die.

In the next section, we provide a study of the energy map through various robustness experiments.

2.5 Study of the energy map

The energy map is an entry point in the recognition framework. Thus its computation must be thorough and must lead to stable results. Various details can impact more or less significantly its quality, and we shall see to what extent they may be detrimental and how they can be managed.

2.5.1 Number of light sources

The energy map is generated from several light azimuths. The calculation (Equation (2.17)) does not initially specify the number of lights.

[1] This choice has been made with the intention of letting the hypothesis of perfect diffuse material aside in order to test recognition algorithms. More complex BRDFs could be used, but might nonetheless also not be representative of what ancient coins are made of. On average, they seem less specular than they are diffuse, due to years of wearing, eroding or oxidation.

Figure 2.21: The 16 types of coin produced by the algorithm. Each type is made of a large symbol with or without a ring of small symbols. There are in this set some very distinct types and some others quite close to each other, making them hard to differentiate visually.

Figure 2.22: Dies from the same types. They all feature the design of the type, but exhibit some noticeable difference due to non-rigid deformations.

Figure 2.23: Five examples of coins from the same die, in order of creation. One die is used to make 5 coins and wears out in the process, leading to less detailed versions of the coin.

(a) Height map. (b) Image. (c) Energy map. (d) Orientation map.

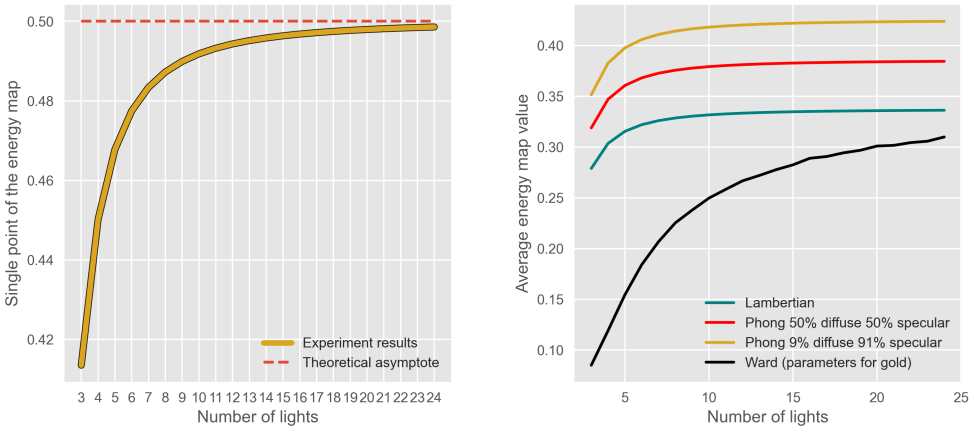
Figure 2.24: Result of computations after generating the height map. Images are taken using the Lambertian model and enable the energy map to be found along with the orientation map of the coin.

2.5.1.1 Theoretical computation

At first, one can use Equation (2.6) to generate an image pixel and see how the single corresponding energy value varies. The point normal is set such that $\alpha = 0$ and $\sigma = \frac{\pi}{4}$. The light colatitude is $\tau = \frac{\pi}{2}$ and the azimuths are defined as equally spaced around the point. The number of light sources m varies from 3 to 25.

The results are presented in Figure 2.25a. The curve shows an increasing function, which converges to an asymptote at $\mathcal{E}(x, y) = 0.5$. Equations (2.14) and (2.11) shows that it should indeed be $\frac{\sqrt{2}}{2} \sin\tau \cdot g = \frac{\sqrt{2}}{2} \sin\tau \sin\sigma = \frac{\sqrt{2}}{2} \times 1 \times \frac{\sqrt{2}}{2} = 0.5$.

The same type of result can be obtained with an artificial height map/normal map of a coin, from which we compute images, just like a camera would do. In Figure 2.25b, results of four different types of reflectance models are presented. The mean value of the energy maps follows generally the same pattern as in Figure 2.25a, except for the Ward model which features a slower increase and barely reaches an asymptote at 25 sources. In general, the behavior of the energy maps is the same, but it is of course preferable for the object to be a diffuse as possible. It takes around 15 light sources to reach 90% of the asymptote value.



(a) Single pixel.

(b) Four types of BRDFs.

Figure 2.25: Evolution of $\mathcal{E}$ with the number of light sources

2.5.1.2 Real object comparison

With real objects, measurements are tricky because images have already been formatted during the acquisition and are loaded as arrays of normalized values in the software. Therefore, measurements on the resulting energy maps make no sense. However, it is still possible to visually assess the quality of the energy map.

We take the acquisition setup in Figure 2.7 and 2.8 with the horizontal positioning and various number of lights to compute the energy maps. In Figure 2.26, the evolution of the energy map with m are presented. From left to right and top to bottom, the map tends to even out and stabilizes after around 7 lights. The first image (top-left corner) features some highlights, for instance in the crown. Those highlights disappear as the number of lights increases. The explanation is that with very few lights, some very specific parts are lit directly (light aligned with normals) while others are spared. This results in a uneven distribution of energy which creates a high contrast during visualization. This effect disappears as lights cover more and more of the surface. It seems that, while one or few images may create specular highlights, adding up more images dims them, which is likely due to the fact that specular highlights are precise and appear only on a fraction of the images while diffuse highlights are distributed over half of the images, following a sine law.

It appears visually clear that the energy maps tend towards a stable version, no longer improved by adding more lights. We verify this assumption experimentally by calculating the variations between two consecutive energy maps.[2] The evaluation presented in Figure 2.27 is based on three coins and shows that a decreasing tendency for all of them, reducing

[2]This is possible here as a way of showing the convergence, since the images will tend to stabilize when m grows larger, no matter what process had them normalized in the first place.

Figure 2.26: Visual evolution of the energy map with respect to the number of lights. The first row displays the energy maps with 3 to 6 lights and the second row from 7 to 10 lights.

the differences towards zero. This implies that the energy maps tend to an asymptotic version of itself.

Visually, the energy map stops improving at around 8 light sources, which is numerically plausible given the variation values after 8 sources (RMS around 0.12 down to 0.10). In terms of practicality, 8 light sources is a good compromise between a good quality and the simplicity of acquisition. Energy maps in Figure 2.11 were computed out of 8 images.

It should be briefly mentioned that the quality of the orientation map (Section 2.2.4) is also dependent on the number of angles. The circular mean described in Equation (2.18) converges toward an asymptotic value as m grows larger. Eight orientations are also satisfactory in this regard.

2.5.2 Azimuth variation

Capturing the images during the acquisition requires that the object O be at the center of the system. It is however unlikely to be so, given that there is no "center" in certain objects like ancient coins. In this subsection the impact of decentering the coin is studied. The center is defined as roughly the place where to put the object so that it visually aligns with the optic axis of the camera (z-axis).

We perform an analysis of the quality of the energy map given a displacement of the coin. More precisely, the coin is moved along the x axis by a fraction of the distance d separating the center from the LEDs circle. Figure 2.28 shows some results on a real coin. Even at 40% of the distance, the energy map stays roughly the same. The coin is still lit by all the light sources, albeit with slightly different angles. For a given point in

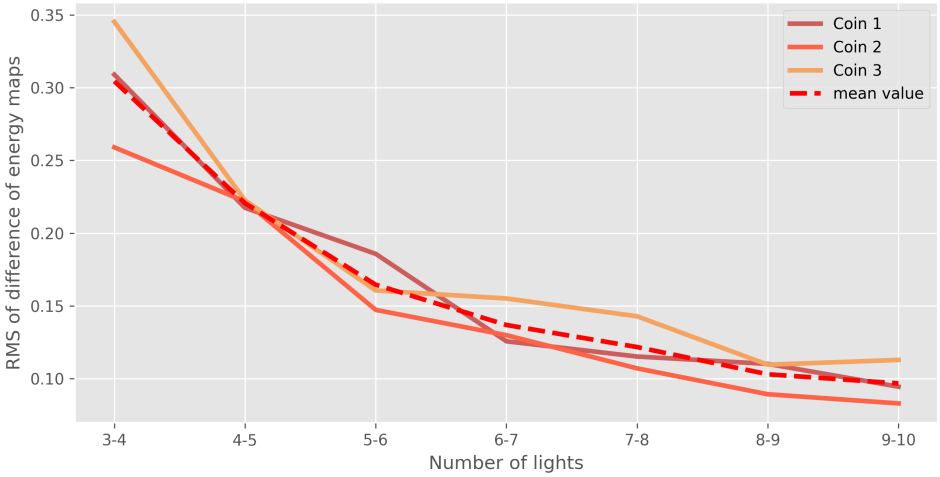
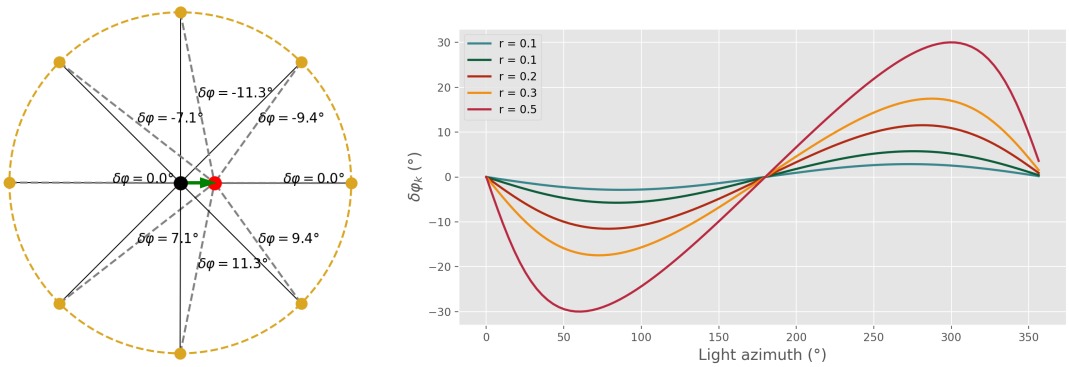


Figure 2.27: Root mean squared error obtain for three different coins.

0% 20% 40%

Figure 2.28: Energy maps for three fractions of the distance d.



(a) Perceived azimuths for $r = 0.2$. (b) Amplitude of the variations against azimuth.

Figure 2.29: Impact of an horizontal decentering on the azimuths. The displacement provokes angle variations which define new perceived azimuths.

U, the variation of the perceived azimuth is:

$$\tan d\varphi = \frac{r \cdot \sin(\varphi - \alpha)}{1 + r \cdot \cos(\varphi - \alpha)} \tag{2.32}$$

The variable r in Equation (2.32) is the fraction of d_x, thus $r \in [0,1)$. The impact of the decentering on the light azimuths in illustrated in Figure 2.29. Particularly, Figure 2.29a shows that from the coin's point of view the azimuths slightly changed. Everything becomes as if the lights had been unevenly distributed around the coin. However, although this change seems to make a significant difference, it largely depends on the direction of the displacement. On Figure 2.29b the maximum amplitudes are reached for 90° and 270°, which are perpendicular to the displacement, and are really small considering small displacements: 2.9° for $r = 0.1$ and 5.7° for $r = 0.2$. Major decenterings are unlikely to happen, which means that the energy map has to account for rather small variations of light positions. Given the same conditions and decentering, the energy map[3] suffers a 5% relative difference[4] from its original version for $r = 0.5$ and does not even reach 1% for values smaller than 0.2.

2.5.3 Colatitude variation

In the same fashion, we analyze the influence of the colatitude on the quality of the energy map. The theory suggests that the colatitude is only an intensity factor (Equation (2.15))

[3]The calculation of the energy map has been performed on artificial coins, so that the true value per pixel can be obtained. The results constitute a lower bound, because they stem from an ideal version of coin. Nonetheless, it provides a clue as to why the differences are so imperceptible on real examples (Figure 2.28).

[4]Calculated as: $\epsilon_{\text{rel}} = \frac{|\mathcal{E} - \mathcal{E}_d|}{\mathcal{E}}$ for each pixel, and averaged.

90.0° 80.9° 72.3° 64.4° 57.4°

Figure 2.30: Energy maps for different light colatitude values.

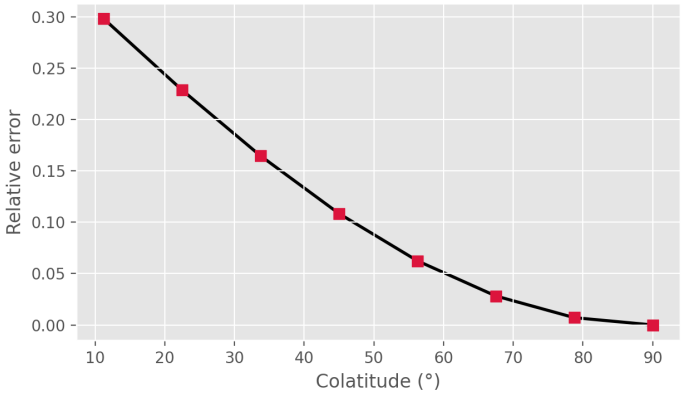
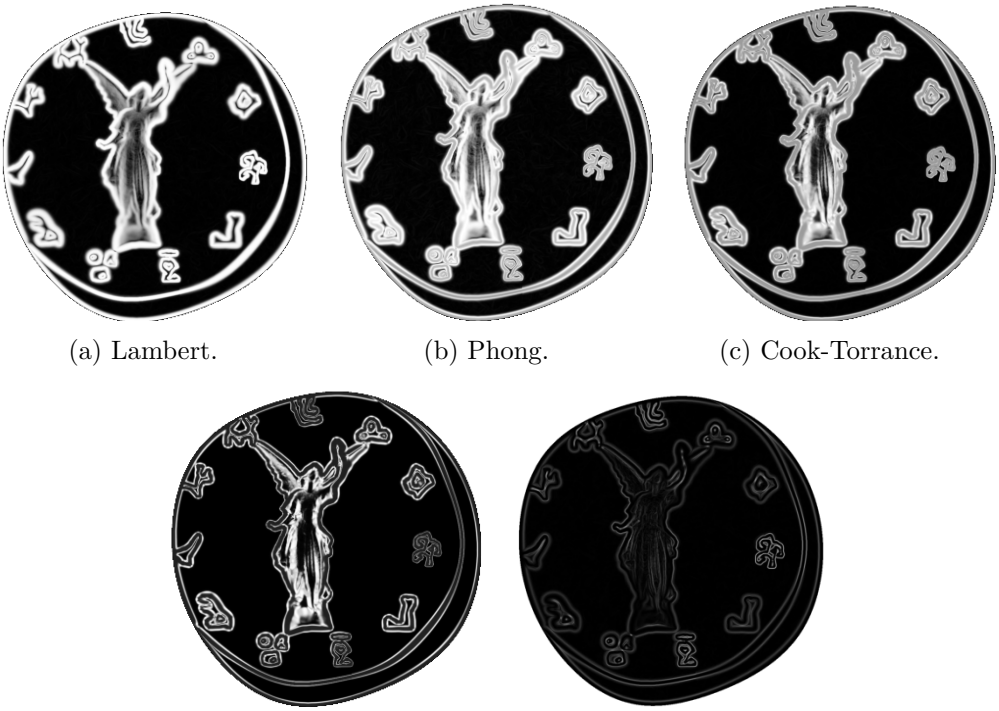


Figure 2.31: Relative error from the energy map obtained at 90° vs the colatitude τ.

which sets the contrast. This phenomenon can be noted in Figure 2.30: coin ridges appear brighter for near-grazing lights and get dimmer and dimmer by diminishing the colatitude. In Figure 2.31, a numeric evaluation of the phenomenon is presented. The graph shows the relative error between two energy maps of artificial coins, one being obtained with the colatitude 90°. In practice, the light source is not a point but a rectangle of LEDs. Light illuminates the object with a range of colatitude, for example from 68.2° to 90.7° (see Table 2.1). According to Figure 2.31, the relative difference stays lower than 5%. This remark echoes what has been mentioned in Section 2.2.2 regarding the fact that the light source covers a range of angles. The degree of "blur" due to relative errors may occur but stays low.

2.5.4 Reflectance

So far, the main assumption which has been made is that the objects are essentially diffuse. Of course this is hardly the case in real life, where one is more likely to deal with a composition of both diffuse and specular reflections. The energy map model thus does not explicitly account for these specular highlights. The coin material, assumed to be perfectly diffuse, gives satisfactory results on most coins even though this is a rough



(a) Lambert. (b) Phong. (c) Cook-Torrance.

(d) Ward. (e) Gradient norm of (a).

Figure 2.32: Energy maps for four types of BRDFs with increasing specularity and gradient magnitude of the Lambertian energy map.

approximation. Some of those results have been presented in Figure 2.11 and none of these examples exhibit a high specularity, which translates into quite diffuse energy maps. Of course, since there is no simple way to verify the quality of an energy map with respect to the theoretical model, we may assume that, according to Section 2.2.6 and especially Figure 2.15, a diffuse energy map should feature single ridges, contrarily to a specular map which features double ridges.

In Figure 2.32, various models of BDRFs have been employed to render images and compute the energy maps by simply using Equation (2.17), regardless of the specularity. The results are placed in order of specularity. Ward's set of parameters is chosen to render pure gold and is the most specular. One interesting phenomenon is the one cited in Section 2.2.6, which is that the energy maps appear to be linear combinations of a diffuse and a specular part. Edges in a specular energy map are somehow a doubled version of the diffuse energy map. One can see for instance how none of those double ridges are present in the purely diffuse coin whereas the opposite happens for the most specular one.

Besides this remark, a complementary note on the relationship between diffuse and specular energy maps is that, in a way, the diffuse model reveals the first derivative of the relief while the specular model is akin to a second derivative. An illustration of this

relationship is presented in Figure 2.32e, which represents the norm of the gradient of a diffuse energy map. This observation is intriguing: it means that some mathematical trick could perhaps reconcile both reflection patterns in a single model and avoid the Lambertian assumption altogether.

In any case, the computation of the energy map is still relevant as long as the specularity is the same for any coin of the same class (understood as groups of coin that one wishes to cluster together). On average this will not be the case but one can expect more deteriorated coins (and therefore diffuse objects) than specular ones, which would represent a particular case for which, depending on the recognition strategy, may fail to be appropriately recognized.

2.6 Conclusions

In this chapter, the main scope was to investigate possible models to represent quasi-flat objects and especially ancient coins. Several models can be relevant for the purpose of recognition, such as normal maps, height maps or simply images. The core information is believed to be captured by the relief in the object and therefore its variations seem to be an appropriate model for analysis and comparison.

The energy map, which description has been published in the Eurographics Workshop on Graphics and Cultural Heritage in 2018 [62] is presented as a very cheap and easily computable model. Through some mathematical calculations of the way it is computed, we have shown that it is in fact equal to the magnitude of the variations within the object's relief. We posit that it is that information which is the most relevant to analyze in order to implement a recognition scheme. Adding information is even possible by computing an orientation map which corresponds to the gradient direction. This takes no more time than the acquisition of the energy map since the normal orientations can directly be deduced from the irradiance of the object as a function of the light azimuth.

These tools sure are close to the models cited above and do not actually provide brand new material. Rather, it is the acquisition and the computation which makes it attractive, as well as its simplicity and interpretability. The energy map is robust to variations of the light source positions and to the object's imprecise positioning. The number of sources required can remain reasonable without losing much content. Overall, we regard this model as an effective and elegant solution to capture the information about the object.

Besides the energy map, a dataset of artificial coin has been introduced. It is thought to be faithful to the actual manufactoring process, while obviously simplifying some operations or events such as hammer strikes and erosion. The resulting dataset can be adjusted for different goals; the proposed one makes distinctions between the coin type and the coin die.

Chapter 3

Coin recognition via contour analysis

In 1976, David Marr [73] introduced a paradigm of visual processing which described its purpose as a construction of a simple but rich description of the image through the analysis of reflectance, illumination, orientation and distance of a given surface. This idea was originally named the *primal sketch* and was decomposed into two parts. The first is to explain the intensity variations in the image via a primitive language of edge-segments, bars, blobs and terminations. This raw sketch is then made explicit with the selection and grouping of the primitives in various ways and scales. This paradigm is useful to conceive the idea of our quasi-flat object as a sum of features out of which meaning is extracted by judiciously arranging them into a larger, interpretable complex. The quasi-flat object is in essence rather simple because it is a simple surface with intensity changes which can be captured via imagery. One can therefore naturally imagine decomposing it into small parts which reveals a more primitive way of recognizing the object: its contours.

3.1 Introduction

In this chapter, we investigate the possibility of using contours to recognize coins. To be more specific, we will focus on the recognition of coins struck from the same die. Those coins share very similar topographies, therefore it seem adequate to build a recognition system based on contours. As it has been demonstrated in Chapter 2, energy maps are a suitable model since they highlight the relief variations of the object. We want to exploit this information to extract the contours from the object.

As a first definition, a contour should represent a distinct feature of the object in the form of a set of contiguous pixels without overlap or loop. The outer border of the coin, defined by the contour of the coin's mask, is an example of rather well-defined contour because its definition is clear and repeatable. However, this is not always the case for inner contours. Even we humans cannot exactly pinpoint their location, where they start and where they end with an absolute certainty. Therefore, in the next sections, a

broad definition will be formulated in order to write an algorithm for their extraction.

The explanations of how such a system is built, as well as some results, are presented in three axes. First, the difficult task of contour extraction is presented; the process of contour matching is explored in a second part; finally, a third section shows some results on coins and conclusions are drawn on the current version of the recognition system.

3.1.1 State of the art

A summary of the state of the art in contour recognition is provided. In a first part, we present ways to extract edges and contours in order to use them for comparison; in a second part, techniques for matching via various tools are investigated.

3.1.1.1 Contour detection

Contour extraction has been under investigation for many years. The general definition of an edge that can be found is that it is a set of pixels which are located on a area of high image gradient values in one direction. From this definition, several filters were described, among which the Sobel detector [91]. This detector takes the gradient information into account through a isotropic 3×3 gradient operator composed of two convolution kernels, one for each direction. The resulting derivatives can then be used to measure the magnitude and the orientation of the gradient within the image. It appears, then, that strong intensity variations in the image are clearly underlined in the magnitude image. However, the edges presented in this way are rather thick and uneven, which makes them unusable as single feature elements.

In order to improve simple algorithms such as the Sobel detector, John Canny introduced in 1986 the Canny edge detector [21]. The algorithm follows three aims. The first one is a good detection: maximize its signal-to-noise ratio, marking as many edge points in the image. The second one is a good localization: find the edge as close to the 'real' center of the edge as possible. Finally, the third one is to obtain a minimal response: edges should be marked only once and noise should not produce false edges. The algorithm itself goes through five main steps: smoothing the image, compute the gradients of the image (basically Sobel detection), non-maxima suppression (thereby creating thin edges), double thresholding and hysteresis edge tracking to remove noisy edges and validate true edges. The sound definition of the Canny detector makes it very reliable to extract meaningful image contours which capture the variations.

The usual issue with such an edge detection technique is that it leaves us with an edge map which is likely to contain many individual disconnected small edges or large edges broken in several pieces. Such a split representation makes the extraction of informative parts difficult. Removing all small isolated edges is a difficult task because one cannot be sure to spare relevant edges which have been disconnected.

One can consider the possibility of reconnecting broken parts of larger edges by inferring their continuation towards other edges. Song et al. [92] introduced an edge

connection technique based on the Canny edge detection algorithm which aims at improving the connectivity of such an edge map. The method relies on the use of a Hough transform instead of the double thresholding performed in the original Canny algorithm. The result of this method indeed improves the connectivity the edges. Nonetheless, such a method is not perfect nor exactly adequate to extract single descriptive edge segments; the Canny detection finds lots of irrelevant, highly interconnected small edges which are still present in their method.

The alternative would be not to detect edges *per se* but to force the detection of an edge network with image segmentation. Segmenting an image does not fulfill the same purpose; it relies on the possibility of finding regions in the image that presuppose the existence of boundaries between these regions. The rationale is different, yet it can be very effective at finding edges provided we consider that they are found in the boundaries. Many methods have been developed with respect to segmentation, but the aim of this state of the art is not to list them all. Rather, we wish to communicate the potential of such a technique for our purpose.

The watershed algorithm [97] is a technique to segment an image based on its topography. The image is thus seen as a collection of basins separated by ridges as its intensity defines the height. This is an interesting perspective to take with respect to the energy map, since it is itself sort of a topographical map, being the gradient magnitude of the object's surface.

In a different manner, the active contour segmentation allows to find closed areas to segment in an image. Examples of this type of algorithm are the snakes [54], which iteratively infer the location of the boundaries with the use of the edges information, or the Chan-Vese method [23], which performs segmentation via level sets, bypassing completely the edge information. The use of active contours to find closed regions can be tedious, especially with snakes, for which a fine parametrization conditions the quality of the result.

Raskar et al. [82] presented an innovative technique to detect depth edge in a scene in order to produce a stylized rendering. Their acquisition method uses a multi-flash imaging via several lights placed around the lens. The method of depth edge detection relies on the projected shadows of the objects in the scene and find the edges by analyzing variations along epipolar lines defined by the light positions. The system rely however on a distinctive background for the shadows to be cast effectively. The authors report that depth edges may also be missed because of detached shadows, holes or valleys or if edges lie in shadowed region.

Finally, connected to the principles of the energy map, Brognara et al. [20] tackled the problem of edge detection on polynomial texture maps ([70]). Because texture maps are a representation of an object surface via multiple imaging, they convey the information about the surface normals. The authors thus suggest to use this gradient information in a classic Canny edge detection. Results of the gradient magnitude are akin to the energy map which is expected given that they are mathematically connected. The final result of the edge detection shows a detailed edge map, much more informative than the

detection on a mere single relighted image.

3.1.1.2 Contour matching

Contour-based methods are widely considered in the literature because they are invariant to lighting conditions and do not take into account colors or textures [84]. The main strategy to characterize the contour is the use of a contour descriptor, which can characterize globally or locally the contour. The corner stone of contour similarity techniques is the shape context descriptor [14], which locally features the distribution of contour points by the means of radial histograms. Although it allows to deal with scaling, rotation and deformations, the similarity cost is rather heavy and cannot cope with partial occlusions. Many works are based on this technique and propose the use of Dynamic Programming to match points [31], a hierarchical strategy to deal with affine and projective transformations [49] or the use of the earth mover's distance to perform comparisons [90]. Chang et al. [24] propose a descriptor for open and closed contours based on the Triangle-Area representation. Their work efficiently deals with scaling by combining several scales and can tackle occlusions. Similar properties can be found in [106, 59], which propose multiscale descriptors based on invariant aspects of contours. They also demonstrate invariance to articulated deformation and intra-class variations. Wang et al. [100] introduce another type of descriptor relying on height functions. The works of [33] and [84] use a descriptor which can efficiently match parts of contours. The general method uses a fixed sampling of the contour from which they extract angles between chords within the contour. Yang et al. introduced the Triangle Centroid Distance descriptor [105], based on distances between each point of the contour and Fourier descriptors. They achieve fine results in both part-to-whole and part-to-part matching while dealing with significant non-rigid deformations. However, their matching technique is not invariant to the position of the starting point and therefore needs as many comparisons per contour as there are points in it. The work of [119] also proposes the use of Fourier descriptors with a multiscale technique using group features, which uses combination of spatial features to improve matching accuracy. Finally, to refine contour description, the work of [89] combines contour and skeleton by analyzing the geometry of contour parts in relation with their associated skeleton.

Alternatively, several other works use signature-based representations to describe the contour, which synthesize the contour information into a compact representation. Commonly used signatures are, for instance, centroid distances, the chord length signature, the area function, the cumulative angular signature and the turning angle signature. Curvature is a paramount element in human contour recognition [9]; Cui et al. [29] formulate a scale-invariant signature based on this principle. The abscissa is calculated as the integral of unsigned curvatures. This mathematically shows to be indeed scale-invariant in theory but in practice fails to take into account the changes in sampling of the contour when perturbations are applied to the image itself. To compare two contours, they use normalized cross-correlation. In [52], a Distance Interior Ratio signature is introduced. It is invariant to rotation and scaling and yields promising retrieval results

on popular datasets.

Various frameworks can be formulated for the matching step, depending on what the desired goal is. Research works describing novel methods for contour representation generally adopt distances or similarity measures such as the edit distance [31], the earth mover's distance [41], the χ^2 distance [14], or dissimilarity costs [105, 33]. In any of the aforementioned cases, pairwise comparisons must be performed if one desires to find the best match among a database of contours which leads to a linear complexity when performing retrieval. Several authors propose alternatives to standard comparisons that can involve space search reduction via statistical tools [64], or the use of Bayesian inference [118]. However, the speed gain is achieved for each comparison, and not for the database as a whole, which is still a limiting factor for a fast retrieval. Other works rely on learning to perform classification. The classic method of Bag-of-Words model is used by [101, 116, 88]. Bai et al. [11] extended this principle by using both local information and global properties of the contour to gain discriminative power for the recognition. The same idea of contour segments is investigated in other works such as [94] which uses Bayesian inference for learning. Learning methodologies can be very effective for contour classification, but require a specific training set, which cannot account for large databases of very different contours.

The locality-sensitive hashing is another way to partition the space to perform an effective search [39]. The algorithm uses hash functions to split each of the descriptor component in two buckets (leading to a one bit encoding). This method is used several times so that the probability of retrieving a correct match for a query in the same bucket is high. This subsequently means that the collisions are maximized, which differs from the common way of using hash tables. The time complexity is sub-linear.

Other methods using hash tables for retrieval were implemented, such as the geometric hashing [61]. In geometric hashing, the idea is to take the feature points for a specific object and then create bases out of pairs of feature points. For each pair of points, each new coordinates of all the other feature points are calculated and quantized, and the values are used as an index to store the basis in a hash table. The retrieval proceeds by performing the same steps for a set of query feature points, choose an arbitrary basis and see which values can be retrieved out of the newly quantized coordinates. These verifications are performed for each query point basis.

3.2 Contour extraction

In this section, the objective is to find a way to detect and extract contours as main features of the object's relief. Their extraction should be discriminative so that a given contour is as unique as possible in the realm of possible contours; it should be repeatable so that contours are extracted exactly the same way every time with a high probability; and it should be informative so that a match can occur between two coins having the same contour. Furthermore, since the goal is to recognize coins originating from the same die, detected contours should be close to one another if and only if those coins are

indeed from the same die; this is not always the case since coins from the same *type* can also feature very similar contours, so this should be at least statistically true.

3.2.1 Overview

The most important detail about this subject is probably the definition of a contour. Surely, it must have to do with underlining specific features of an object and it is understandable that it refers to some boundaries within the object or scene's texture. Yet, no definition perfectly suits them intrinsically; they are always assumed to exist via the algorithm which detects them. We define a contour as a set of contiguous, 8-connected pixels, in the form of a 2D vector of coordinates. Each pixel cannot be connected to more than two others. An open contour has therefore one start and one end point, whereas a closed contour has none. It is meant to represent a part of the object, emphasizing meaningful borders within it. A contour, defined like so, does not have to embody an outer border of the object; any significant variation between two regions of the object qualify as a candidate for a contour. Thus, gradient variations in an image is a powerful tool for detection. Edges are, in general, quite comparable to contours but we posit that edges are mere groups of pixels indicating high variations. We will also refer to the product of an edge detection as an edge map.

In the following, two contributions are introduced to integrate the information of the energy map to the edge detection scheme, from which we derive a proper formulation of contours and design a selection based on the information they carry.

3.2.2 Detecting edges in the energy map

The contours must be characteristic of the object they are extracted from. Most work seems to rely on the image model to infer edges via variations in the image gradient. In our application, we have access to the object itself. In the two next parts, we introduce the use of classical method with the energy map in order to find contours characteristic of the object, and not just images of it.

3.2.2.1 Canny edge detector with the energy map

The Canny edge detector remains the best choice for a first attempt at extracting contours from our objects. This would by definition provide thin, well-localized and repeatable edges, which is very convenient for recognition tasks. Edges have the advantage of being acquired regardless of light conditions. However, this is mostly true for textured, flat objects. In our scenario, light is the fundamental tool for revealing the relief of the quasi-flat object. Using only one image limits the relevancy of the edge detection to only one part of the object, that is the side exposed to light. Therefore, the energy map is used to extract the contours.

An example of the result is presented in Figure 3.1a. One can notice that the detector efficiently captures the informative details of the relief by highlighting the highest

variations with thin edges. However, these edges are not simple high variations indicators. Because the energy map is already a depiction of the surface's variations and because the Canny detector begins with the application of a gradient operator, the resulting edges actually embody the energy map's variations. Those are the product of the second derivatives of the relief. The observable edges, although visually satisfactory, are thus twice as numerous as there are ridges in the energy map.

The main idea to circumvent this effect and to improve the quality of the edge map is to take into account the variations of the relief itself. This information has already been captured in the energy map $\mathcal{E}$ and the orientation map $\mathcal{A}$ (Equations (2.17) and (2.18)). Usually, the Canny detector involves the use of a Sobel operator to infer the coarse version of the edges. Instead, the gradient information, in the form of $\mathcal{E}$ and $\mathcal{A}$, is inserted prior to the maxima suppression and thresholding operations.

The outcome of such a modification is shown in Figure 3.1b. There are less contours and these are exactly positioned where the ridges of the energy map were located. Avoiding duplicates like so is important because otherwise it would be akin to using each contour twice in the matching process, which costs time and space. The various assets of the Canny edge detection adapted to the energy map are clear: relevant localization; repeatability, so long as the energy map is; no superfluous contours; detection of informative features (corners, complex contours, small details,...); invariance to rotation and shift; and invariance to illumination conditions since it encompasses and takes into account several light azimuths.

However, there remain some issues regarding the extracted information so far. The Canny detector takes some aspects of the connectivity between edges into account – mostly during the hysteresis thresholding – which make no assumption about the possibility of connecting various distant edges together. Thus, holes and discontinuities are present. In the form of an edge map, which is a binary image, those discontinuities do not seem like an issue since edges ends remain spatially close. Our goal, however, is to extract those edges in the form of contours, lists of contiguous pixels which will definitely be impacted by those discontinuities. Unfortunately, the product of some extraction operation will be different according to the occurrences and the positions of such breaks, which is why we must rely on a statistically sufficient repeatability of the detection.

3.2.2.2 Energy map ridges and skeleton

In order to detect edges so that it is possible to extract contours out of them, we elaborate a different strategy. On the one hand, the Canny detector is a really efficient technique to find edges. The results are promising for contour extraction, but we have noticed that many discontinuities may occur and jeopardize the quality of the final contours. On the other hand, segmentation techniques ensure the connectivity of the edge network but force the creation of irrelevant edges which do not respect the principal underlying motive of detecting characteristic parts of the object. Moreover, the principal goal of segmentation is incompatible with the topography of our object; there are no distinct regions to find and it would make no sense to follow this path.

(a) Original Canny detection. Edges are doubled with respect to the amount of ridge visible in the energy map.

(b) Adapted Canny with the energy map information. One can observe single edges positioned at the energy map's ridges location.

Instead, let us consider the energy map as it is: a representation of the high relief variations, where ridges mark highly characteristic areas. One can imagine drawing lines on the crests and find relevant edges this way. The tricky part is therefore to know where the edge stands on the ridge. There is a priori no better definition than the thickness of the ridges, which is insufficient to infer proper edges. However, those ridges form elongated shapes from which it may be possible to extract a midline.

Firstly, the energy map $\mathcal{E}$ undergoes a local thresholding operation. A general thresholding is not relevant due to the variations of intensity throughout the energy map. For each point in $\mathcal{E}$, an area characterized by its length ℓ is defined. This parameter enables the extraction of thicker or thinner blobs out of the energy map's ridges. In prevision of the next operation and from trying various parameters, a value of 101 px turns out to be good choice. The thresholding is thus performed inside this area, within which the Gaussian weighted mean $\mathcal{G}$ is calculated. The final product of the thresholding is given by Equation (3.1), where every value ranges between 0 and 1, and an offset parameter has been set to 0.04. The resulting binary map may contain small blobs and large blobs featuring holes. The small blobs must be eliminated altogether while small holes must be filled. There is no apparent way to theoretically determine threshold parameters for these purposes. So we rely on a pragmatic choice which considers that blobs must be large enough to contain relevant edges and that holes must be large enough to stem from obvious topography features of the object. The area of a blob must therefore be larger than 256 px^2 and holes must be larger than 128 px^2.

$$\forall (x,y) \in U, \quad \mathcal{B}(x,y) = \begin{cases} 1 & \text{if} \quad \mathcal{G}(x,y) - 0.04 < \mathcal{E}(x,y). \\ 0 & \text{else.} \end{cases} \tag{3.1}$$

We call the resulting binary image a blob map, since it features characteristic blobs representing the position of ridges in the energy map. An example of such blob map is given in Figure 3.2a. The blob map is merely a representation of the energy map where the largest variations occur. The size of local neighborhood blurs some of the information; for instance the hair part has disappeared with the chosen value. If we had wished to keep it, we would have selected a lesser value, which would also mean thinning the ridges with the risk of segmenting them and create discontinuities.

The second step is to infer edges from those blobs. Once again we are facing a definition problem. To remain coherent with previous definitions and because the blobs already represent highly informative areas, edges are considered to be the center line of the blobs which is equidistant from the boundaries. This definition makes sense for three reasons: firstly, an edge is designed to be at the top of a crest which is very likely to be a maximum point in $\mathcal{E}$. Secondly, this definition minimizes the energy that would cost such an edge to run through this area, given a potential $-\mathcal{E}$. In order words, there are no superfluous detours on the edge; details on the edge cannot be due to noise. Thirdly, the product of such an operation is a set of connected networks of edges, which enables discernible contours to be found. To obtain this result, a topological skeletonization is applied to the blob map via [25]. Figure 3.2b presents the skeleton obtained from 3.2a.

Note that this represents a coarser edge map than the one obtained via Canny detection. There appears to have less edges overall than in the Canny edge maps. In this example, there are around 14000 edgels (pixels of edge) in the skeleton edge map against 27000 in the adapted Canny edge map. By contrast, the standard Canny edge map reaches a total of around 94000 edgels! This does not mean that less is better though; it just mean that the information is hopefully better condensed and purified in the two last approaches than in the first one.

The main caveat with the skeleton method is that it relies on the fact that all ridges are long enough to produce coherent edges. This is, however, not always the case since some blobs may be rounder or feature protrusions. These produce very short edges which therefore carry no information. Another artifact is the apparition of branches. In theory, this is unavoidable because there are many connections between the initial ridges. What is not relevant however is any spurious branch which occur in blobs getting thicker at the end. These produce short edge segments which are once again devoid of information.

3.2.3 Extracting contours

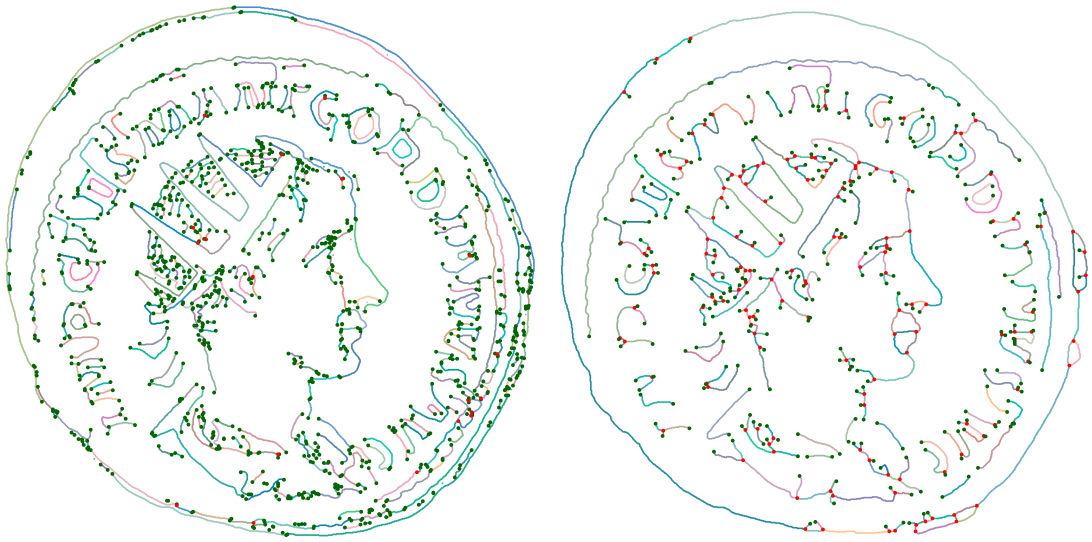
So far, we have been essentially tackling the detection of edges while preparing a fertile ground for extracting good contours. Remember that a contour has to be a contiguous set of edgels, which is why the quality of edge detection matters a lot. The extraction of contour is the operation which allows us to go from the edge map to a set of contours. However, this is trickier than it looks because of two major drawbacks. The target of the extraction is any parts of the edge map forming a contiguous sequence of edgels (and respecting the contour topology). Yet, there are many intersections and branch points, which therefore extends the possibilities of candidate edge segments. Is keeping inner segments (between two intersections or end points) enough? Should the method create as many contours as there are connected segments along the intersections? If so, how many intersections are enough? The second drawback is the presence of spurious small edges which are either connected to other edges via a node (mostly in the skeleton) or isolated from all the other edges. Should these edges be removed? If so, what is the minimum admissible length?

3.2.3.1 Segments, nodes and end points

The first step is to convert the edge map into a collection of segments, end points and nodes. A segment is any contiguous set of edgels which starts and ends with either an end point (there are no other connection to that point) or a node (intersection). The detection of nodes and end points is done via the hit-or-miss transform with an exhaustive list of possible structuring elements. Some exceptional cases such as nodes composed of four connected points must also be taken into account. The result is shown in Figure 3.3 in the form of a segment map, also displaying nodes and end points. Note that there are more segments in the Canny map (Figure 3.3a) than in the skeleton map (Figure 3.3b), which makes sense given their respective amount of edgels. There are

(a) Blob map $\mathcal{B}$ of a coin. Ridges in the energy map appear as elongated blobs which cover areas of large variations.

(b) Skeleton of a coin. Elongated blobs naturally give long edges without discontinuities.



(a) Adapted Canny segment map. (624 segments, 19 nodes and 1138 end points).

(b) Skeleton segment map. (387 segments, 157 nodes and 308 end points).

Figure 3.3: Segments, nodes and end points. Each segment has a distinct color, red dots represent nodes and green dots represent end points.

much more end points and much less nodes in the Canny version than in the skeleton version. This is the consequence of the low connectivity of the former and the desired high connectivity of the latter. In terms of graph theory, we should refer to this quantity as the segment density [26], which is the ratio between the number of segments and the maximum possible number of segments,[1] which is be defined as:

$$D = \frac{2|\mathcal{S}|}{|\mathrm{V}|\,(|\mathrm{V}|-1)} \tag{3.2}$$

where $\mathcal{S}$ is the number of segments and V is the set of nodes. In this example, the segment density is $20\mathrm{e}-4$ using the skeleton map and $5\mathrm{e}-4$ using the Canny edge map. The difference in density is conspicuous and it appears blatant that less density leads to more potential for connecting segments into longer ones.

3.2.3.2 Pruning branches and spurious segments

The second step is to clean the segment map so as to remove uninformative parts. There are two kinds of spurious segments: the small isolated ones and the small useless connections to a node, which create branches. For both of these categories, we use Algorithm 1, in which we suppose known a function $ep\colon\ \mathcal{S} \to \{0,1,2\}$, which returns the number of end points of a given segment. The process eliminates iteratively any segment, featuring at least one end point, that does not satisfy the desired length $\ell_{\min}$,

[1] This definition holds because the graph is undirected.

which is chosen to be 20 px. Because small segments are successively removed, some new small segments with at least one end point may appear, and those will be removed in the next iteration. The algorithm always terminates because, in the worst case scenario, all segments would be connected and have a length inferior to $\ell_{\min}$. The process would eliminate every single one of them and the output would therefore be an empty set.

Algorithm 1: *prune*: eliminate spurious segments

Input : $\mathcal{S}$: set of segments
$\ell_{\min}$: maximum accepted segment length

```
run ← False
while ¬ run do
    run ← True
    Ŝ ← empty list
    for s ∈ S do
        if ep(s) = 0 then
            // The segment is connected to other segments
            continue
        else
            // The segment has one end point or is isolated
            if length(s) ≥ ℓ_min then
                Ŝ ←(append)— s
            else
                run ← False
    Create a new segment map from Ŝ and partition the new segment map so as
     to obtain a new set S.[2]
return S
```

As a result, the algorithm efficiently cleans the segment map while preserving the density. Figure 3.4 shows the pruned segment maps for both the adapted Canny method and the blob skeletonization method. These represent cleaner versions of Figure 3.1b and Figure 3.2b. For the adapted Canny version, the number of segments has diminished by around 56%, while for the skeleton version, it is around 64%. In both cases, this implies that a large amount of segments have been eliminated, which contributes to lesser memory needs and computational run time for further algorithms. Nevertheless, this cleaned version is not necessarily free from short segments; they can be connected

[2]This operation may be done with various methods. For instance, one can consider the set $\mathcal{S}$ to be a binary image instead of a list of segments, and iteratively add images of the segments. As a caveat, the nodes and end points must be taken into account for the final re-segmentation.

(a) Pruned adapted Canny (273 segments left)

(b) Pruned skeleton (141 segments left)

Figure 3.4: Result of pruning the segment map with Algorithm 1.

to long segments at both sides and are therefore spared in the pruning step.

3.2.3.3 Connecting segments

In order to complete the contour extraction, we need to go from a set of segments to a set of contours. Even though it seems that segments and contours are one and the same, the mere selection of individual segments as the final contour set is inadvisable. One can nonetheless play with the connectivity of the segment network to define contours as any combination of connected segments. Therefore, the number of contours is larger than the number of segments and this enables the discovery of more complex structures which carry information of good quality.

To perform such a task, the segment network has to go through Algorithm 3 via the core method Algorithm defined in 2. We suppose known two mappings $s\_\mathrm{v} : \mathbb{N} \to \mathbb{N}$ and $\mathrm{v}\_s : \mathbb{N} \to \mathbb{N}$, which respectively associate a set of nodes to an segment (referenced by its index) and a set of segments to a node.

Algorithm 2 is recursive. From a single segment argument, it examines its nodes (if it has any) and considers every segment connected to them. Each adjacent segment is then processed by a second recursion, which analyzes the next set of connected segments. At each recursion, the initial segment is taken as a contour, the connection with its adjacent segments as well and every single sub-connection calculated in deeper recursions are also considered as contours.[3] The algorithm can go as far as necessary, but it seems unwise

[3]The sequence of pixels in each contour may need to be ordered by contiguity, depending on their future purpose.

to let it run too far because it will produce lots of contours without necessarily adding information; there is a high chance of getting overlaps stemming from noisy segments. The parameter *depth*, which can be interpreted as the length of the final contour (number of consecutive segments), is set to 3.

Small contours have a higher chance of being fragments which are not useful. Because we desire to avoid those small segments, any contour having a length less than 200 pixels is removed. All contours may also feature a large amount of points, which turns out to unnecessarily increase the computational time of future tasks, including recognition. Each contour is therefore resampled into a set of 100 points obtained via uniform interpolation. This is deemed to be sufficient to improve run times without loosing important details in the contour.

3.2.3.4 Complexity criterion

In order to perfect the final set of contours, one needs to take into account the fact that not all contours are suitable candidates for matching. For instance, many contours, small and long, will often be close to straight lines. Now, of course this can be an informative type of elements – after all, they represent building blocks of many geometric figures. However, it would be irrelevant to keep them; they appear often and have therefore a high matching potential. The same reasoning also goes for full or partial circles.

To efficiently select the contours, a one-dimensional measure has to be used. The goal is to eliminate the contours that have a high probability of occurring, which encompasses lines, slightly bent contours, circular shapes, etc. Eliminating those elements can be done in a simple way since the contours they represent have a constant curvature. In the case of lines, it is 0; in the case of circles, it is the inverse of the radius. Any part of a contour which features a line or a circular part can be said to add little information, whereas a complex contour has a less predictable curvature and is therefore highly informative.

There are several ways to define the curvature of a contour, the most commonly described being the analytic formulation. This formulation is not really appropriate in the plane U. Instead, we use the turning angle, which for a set of 100 ordered contour points $\{P_s\}$ is defined as:[4]

$$\psi = \angle(\overrightarrow{P_sP_{s-k}}, \overrightarrow{P_sP_{s+k}}) \tag{3.3}$$

The step parameter k is important in order to tune the angle value so that a more precise measurement is obtained; choosing $k = 1$ for instance would let us with only eight possible values. The choice of k is dependent on the scale and it has been found empirically that $k = 5$ gives satisfactory results. Finally, we define the complexity of a contour as the quantity of information brought by its turning angle values. Using the definition in Equation (3.3), we formulate:

$$\mathcal{C} = \mathrm{H}(\psi) \cdot (\max \psi - \min \psi) \tag{3.4}$$

[4]The number of turning angle values is therefore less than the length of the contour because of boundary restrictions. Only $100 - 2k$ values are computed.

Algorithm 2: *probe*: (recursive) find all sub-segments given a maximum depth starting at a given segment.

Input : $\mathcal{S}$: set of segments
seg_0: index of initial segment
$depth$: current depth of search
$depth_{max}$: maximum depth of search
$checked\_nodes$: list of tracked nodes
$checked\_segments$: list of tracked segments

$nodes \leftarrow s\_\mathrm{v}(seg_0)$
$\widehat{\mathcal{S}} \leftarrow$ empty list of segments
if $depth = depth_{max}$ *or* $length(nodes) = 0$ **then**
  return $\widehat{\mathcal{S}}$ `// we have reached the final depth or there is no node to check`
else
  for $\mathrm{v} \in nodes$ **do**
    if $\mathrm{v} \in checked\_nodes$ **then**
      continue `// node has already been checked`
    $segments \leftarrow \mathrm{v}\_s(\mathrm{v})$ `// all segments connected to node v`
    $checked\_nodes \xleftarrow[\text{append}]{} \mathrm{v}$
    for $seg \in segments$ **do**
      `// for all segments attached to the node`
      if $seg = seg_0$ *or* $seg \in checked\_segments$ **then**
        continue
      $\mathrm{V}_{seg} \leftarrow s\_\mathrm{v}(s)$ `// Set of nodes connected to` *seg*
      `// prevent loops from happening`
      $loop \leftarrow$ False
      for $\mathrm{v}_{seg} \in \mathrm{V}_{seg}$ **do**
        if $\mathrm{v}_{seg} \neq \mathrm{v}$ *and* $\mathrm{v}_{seg} \in checked\_nodes$ **then**
          $loop \leftarrow$ True
      if $loop$ **then**
        continue
      $\widehat{\mathcal{S}} \xleftarrow[\text{append}]{} probe(\mathcal{S},\ seg,\ depth + 1,\ depth_{max},\ checked\_nodes,\ checked\_segments)$ `// continue probing at depth + 1 and append the result`
return $\widehat{\mathcal{S}}$

Algorithm 3: *find_segments*: find all sub-segments given a maximum depth

Input : $\mathcal{S}$: set of segments
$depth_{max}$: maximum depth of search

$\widehat{\mathcal{S}} \leftarrow$ empty list of segments
checked_nodes $\leftarrow$ empty list
checked_segments $\leftarrow$ empty list
for $seg \in \mathcal{S}$ **do**
  $\widehat{\mathcal{S}} \xleftarrow[\text{append}]{}$ *probe*($\mathcal{S}$, *seg*, 0, $depth_{max}$, *checked_nodes*, *checked_segments*)
  checked_segments $\xleftarrow[\text{append}]{}$ *seg* `// this segment should not be taken into account for future contour creations`
return $\widehat{\mathcal{S}}$

H is the entropy operator $\mathrm{H}(\psi) = -\sum_i \mathbb{P}(\psi_i) \log \mathbb{P}(\psi_i)$, where $\mathbb{P}(\psi_i)$ is the probability of occurrence of the angle ψ_i. The probabilities are obtained via a 10 bins histogram. Note that Equation (3.4) uses the range of values of the turning angle. It has been found to be critical to differentiate between high entropy contours with and without large variations.

Some results are presented in Figure 3.5, which features contours obtained via the skeleton of energy map blobs. One can observe that the general tendency is that simple contours of constant curvatures tends to have a low complexity and that it seems correctly correlated with visual assessments. As an arbitrary decision based on observations, a threshold of $\mathcal{C} = 2.0$ marks the acceptable lower limit of contour complexity, thereby completing the selection of the final contours.

The efficacy of the recognition in future algorithm will depend on various factors such as the number of contours, their complexity and their length. Statistical values regarding these aspects are listed in Table 3.1. In particular, one can note the gap between contours extracted from the skeleton of energy map blobs (Section 3.2.2.2) and the ones extracted via the Canny detector coupled with the energy map (Section 3.2.2.1). There are way less contours in the latter, which is essentially due to the selection of the minimum acceptable contour length, set to 200. The number of contours from the Canny version of the edge map is ridiculously low and doubtlessly condemn the recognition to fail. Reducing the minimum to 100 increases the number of contours up to 355 and 85 after thresholding the complexity.

The correlation between the contours' length and complexity is on average around -0.35 before thresholding and inferior or equal to 0.1 in magnitude afterwards. This means that the removed contours which carry little information are likely to be long contours (probably circular arcs surrounding the coin border). The final contours have a complexity independent to their length which has been, on average, logically reduced.

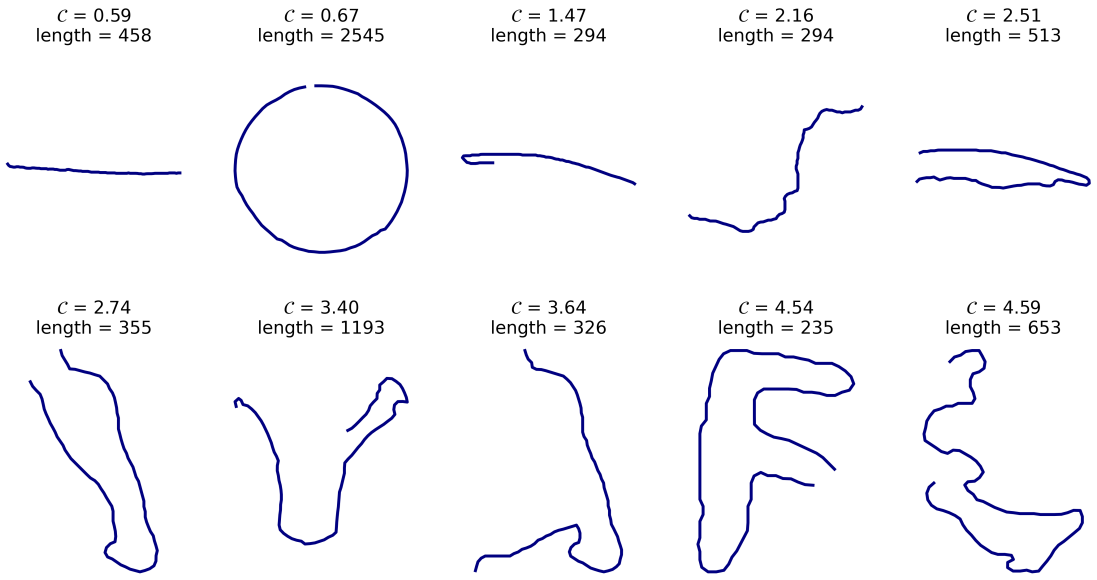


Figure 3.5: Complexities of various contours, in ascending order. The length is calculated before interpolation and is measured in pixel.

Detection method	Skeleton	Skeleton ($\mathcal{C} > 2.0$)	Canny	Canny ($\mathcal{C} > 2.0$)
Number of contours (%)	207 (100%)	143 (69%)	12 (100%)	3 (25%)
Mean complexity (*st.d.**)	2.4 (± 0.9)	3.0 (± 0.6)	1.3 (± 1.0)	2.9 (± 0.8)
Mean length (*st.d.**)	424 (±352)	342 (± 200)	444 (± 363)	263 (± 45)
Correlation length/complexity	-0.36	0.09	-0.34	-0.10

Table 3.1: Various results regarding complexities and lengths obtained with 10 coins of different types. This shows the differences between contours extracted from skeletons and from Canny.

* *st.d. = standard deviation.*

3.3 Contour recognition

Now that contours have been extracted, we want to perform the coin recognition with a contour matching system. One can hope for an efficient comparison of two contours so as to determine if they match or not. However, the set of contours that we extracted in Section 3.2 can be as large as several hundreds of elements and, in these conditions, the total pairwise comparisons can cost a lot of time. In this section, we specifically tackle the exact retrieval of contours.

3.3.1 Overview

A significant part of the literature focuses on contour classification considering occlusions, perturbations due to noise, scaling, rotation and even sometimes large deformations. Contours, in our framework, are to be matched if they are extracted from two coins struck by the same die, which means that the matching criterion is even finer, making the intra-class variations supposedly extremely low. Therefore a large part of the existing state of the art cannot discriminate visually similar but differently labeled contours since they admit large amounts of deformations. Also, most of the current literature revolves around the use of direct descriptor comparisons, which can lead to high computational times for retrieval. We decide to tackle the problem of exact contour retrieval in a large database, in constant computational time. The objective is to retrieve only coins that are produced by the same matrix. This is a problem of exact contour retrieval different from typical contour retrieval approaches, which would retrieve all coins of the same type. Hence, two contours are considered here to be the same if one can be mapped onto the other by applying a rigid transformation or noise, as presented in Figure 3.6.

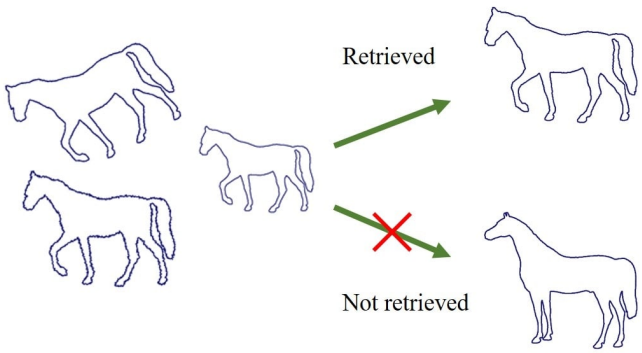


Figure 3.6: Exact contour retrieval. A contour is retrieved if and only if there is a rigid transformation or noise between it and the query. If the contour seems visually to belong to the same class, but is in fact geometrically different, it will not be retrieved.

The intuition for this work was given by a conjunction of ideas from both the audio

field and the contour recognition community. The work of [29] is a very simple and compact way of describing the contour by its curvature. However, the cross-correlation is not robust and quite slow. Wang et al. introduced in [99] the Shazam system for musical audio recognition, where retrieval is performed using a hash map to efficiently store feature pairs of the signal as well as a binning technique to ensure the alignment between similar parts. We therefore investigate this idea in the context of contour recognition, without the need for a hash key for the creation of the feature pairs. The use of associative arrays for exact contour matching resembles geometric hashing. However, geometric hashing was originally developed in computer vision for matching point features [103], such as vertices of shapes, and not contour signatures. Moreover, dealing with small deformations involves the use of a robust signature. Since the cumulative sum [29] accumulates noise along the signature, this representation is not at all robust to perturbations. Our signature is a modified version of the Triangle Centroid Distances (TCD) [105].

This section introduces the following contributions: 1) A signature based on a modified version of the TCD [105], which makes a simple yet very robust 1D representation of the contour. 2) A selection of feature points and feature pairs extracted from the signature. 3) An efficient storage system via associative arrays. 4) A general algorithm of constant time retrieval using pairwise geometric alignment. We call this whole method *Low-complexity Arrays of Contour Signatures*, shortened in LACS, which was published in Pattern Recognition in 2021 [63]. In order to derive meaningful results from the algorithms which are described below, the experiments are performed on various contour data sets. The objects which are investigated are outer contours of 2D binary images and are only a special type of contour forming a closed loop.

3.3.2 The contour as a signature

Our choice is to convert the contour into a 1D signature so as to get a compact representation which contains relevant information of the contour. Attneave [9] showed in 1954 that the human mind seems to privilege a very few, but really precise quantity of data, gathered in its boundary curvature. In order to comply with this psychological background, a natural choice is to study signatures which feature this kind of attributes. Most naive signatures [117] are not invariant to scale and features such as curvature also suffers heavily from noise [29]. Namely, the main issue with the computation of a good signature is its stability under perturbations.

3.3.2.1 Signature definition

We modify the TCD descriptor designed by [105] and extract a signature from it. The TCD is based on the Fourier transform of distances between points of the contour. Namely, if the contour has ℓ points, for each pair of points P_r and P_s where r and s are integer curvilinear abscissae such that $s \neq r$, let g_{rs} be the barycenter between P_r, P_s and the total center of gravity of the contour G. Then, the distance d_{rs} is defined as the distance between P_r and g_{rs}. Figure 3.7 provides a illustration of this calculation.

Each distance is normalized by the maximum of each row, giving Equation (3.5).

$$\forall (r,s) \in [0,\ell-1]\times[0,\ell-2],\ \widehat{d_{rs}} = \frac{d_{rs}}{\max_{r'} d_{r's}} \tag{3.5}$$

Finally, a Fourier transform is applied for each column of (d_{rs}) for 16 frequencies, from which the absolute value is taken, which leads to the final descriptor $(f_{\omega s}) \in \mathcal{M}_{16,\ell-1}(\mathbb{R}^+)$ in which each element is given by Equation (3.6).

$$\forall s = 0,...,\ell-2, \quad \forall \omega = 0,...,15\ , \quad f_{\omega s} = \left| \frac{1}{(\ell-1)\cdot \max_{r'} d_{r's}} \cdot \sum_{r\neq s} d_{js} \cdot \exp\left(-\frac{2i\pi r\omega}{\ell-1}\right) \right| \tag{3.6}$$

where $i^2 = -1$ and $|.|$ is the absolute value.

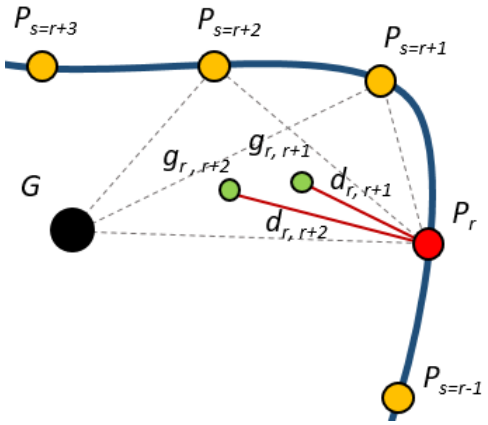


Figure 3.7: Graphical explanation of the TCD. d_{rs} is the distance between P_r (red) and g_{rs} (green). G is the center of gravity of the whole contour.

The difference in description compared to the original TCD is that we consider the d_{rs} to be the direct norm of the vector $\overrightarrow{P_r P_s}$. This formulation is much simpler and yields better results because it deals with exact positioning. Our principal signature, which is sufficient for a large number of cases, is the case of zero frequency in Equation (3.6), and is simply the ratio of the average and the maximum of the d_{js}:

$$f_s = \frac{1}{(\ell-1)\cdot \max_{r'} d_{r's}} \cdot \sum_{r\neq s} d_{rs} = \frac{\overline{d_{rs}}}{\max_{r'} d_{r's}} \tag{3.7}$$

This signature is robust to rigid transformations as well as noise and characterizes the contour. As explained below in Section 3.3.4.3, one can nonetheless use at least the first few rows of $(f_{\omega s})$ to improve the robustness. It should be noted that each contour is read forward and backward, since there is a priori no way to select a stable direction. The query contour will thus have two signatures to compare.

3.3.2.2 Behavior of the signature

This simple formulation is very efficient as a signature since it is, in theory, invariant to shift, scale, rotation, starting point (resulting in a shift in abscissa) and very robust to noise, being the low frequency of the distance signal. Here we focus on the formulation presented in Equation (3.7). Three types of behavior regarding transformations are cited below: adding noise to the contour, scaling the source and rotating the source (which changes the starting point). For illustration, the first contour of the Kimia99 dataset [87], displayed in Figure 3.9, is used. All original images have an average resolution of around 120×120. A binary blob is extracted from the image and the contour is extracted as a contiguous sequence of pixels.

Scaling the image

Scaling is applied to the image prior to the contour extraction and the calculation of its signature. The larger the image, the more precise the contour details. Figure 3.8 presents the two versions of the signature of the same contour having undergone a scaling by 2. It is paramount that the signature is as precise as possible so that its extrema stay very stable (see Section 3.3.3). The work in [105] uses a uniform interpolation of 100 points whereas we choose 1000 to gain precision.

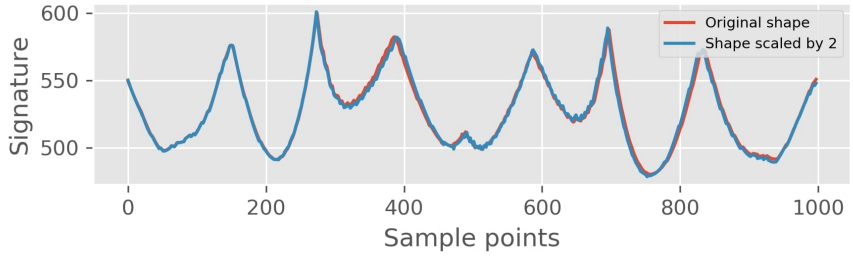


Figure 3.8: Signatures obtained with two different scalings of the contour. There is a slight difference, but overall the signatures are almost identical.

Rotating the image

Rotation changes the location of the starting point within the contour. Given a binary blob image from which the contour is extracted, there is no way to determine robustly which point to start the contour with, and therefore it changes with rotation. This is particularly visible in Figure 3.9, where the red dot denotes the starting point of the contour. Figure 3.10 reveals that the signature shifts accordingly.[5] The only major

[5]The starting point is decided via the method which extracts the contour; in our case, it was an OpenCV algorithm. The point which starts the contour is the leftmost point of the first row containing contour points (the ordinate is considered a top-down axis).

issue with this phenomenon is that it can hide a potentially relevant feature point being located at the break point that would appear as a peak or a trough in the signature.

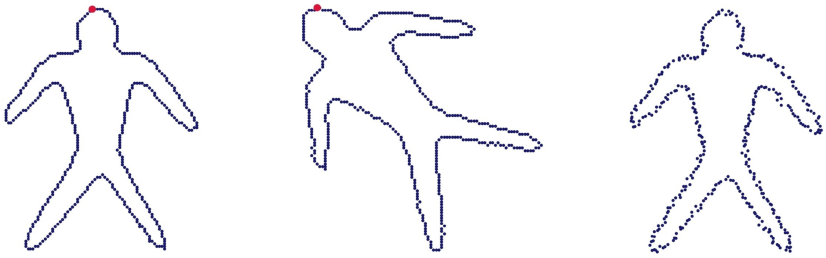


Figure 3.9: Left and middle: Impact of rotation on the starting point (red dot). The contour here is rotated by $\frac{\pi}{4}$. Right: Contour with Gaussian noise of standard deviation 0.5.

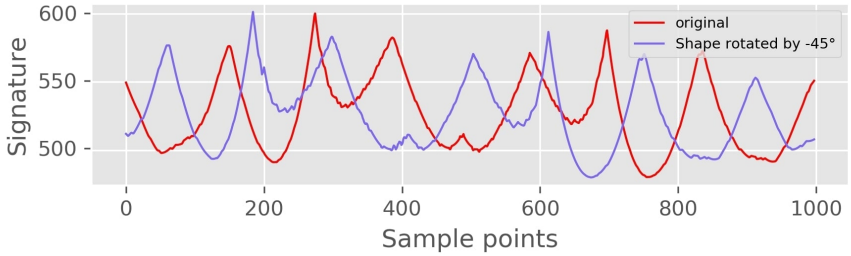


Figure 3.10: Impact of rotation on the signature. The signature shifts by the same amount as the curvilinear abscissa.

Adding noise to the contour

In order to simulate noise, we simply add a Gaussian noise to both coordinates of the contour pixels independently. Figure 3.11 shows the impact of adding noise on the signature. A Gaussian noise of standard deviation 1.0 is rather strong for the size of this contour, therefore it can be considered as a limit case. The signature can be very corrupted by noise even though the lowest frequency signal is used. This is considered to be a limit case since we do not intend to deal with large deformations.

3.3.3 Feature points and feature pairs

To reduce both spatial and time complexities, feature points are extracted from the signature values and organized in pairs. That way, we ensure to keep the model very compact while maintaining a geometrical consistency.

Feature points of our signature are defined as the points which exhibit a distinct variation in signature. The feature points are chosen to be the local extrema of the

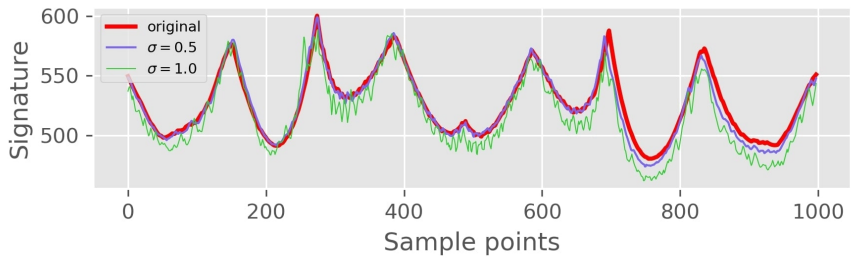
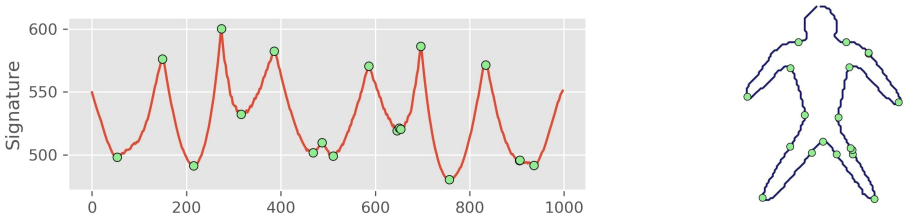


Figure 3.11: Impact of noise on the signature. Noise in the contour implies noise in the signature. Strong noise can significantly corrupt the signal and disturb the local extrema locations.

signature of coordinates (s, f_s) such that:

$$P_{feature} = \Big\{(s, f_s) \mid f_s > f_{s-1} \text{ and } f_s > f_{s+1} \quad \text{or} \quad f_s < f_{s-1} \text{ and } f_s < f_{s+1}\Big\} \tag{3.8}$$

Figure 3.12 shows these feature points in our example in both the signature and the contour. A simple analysis of Equation (3.7) reveals that extrema in the signature appear where points are the closest and the furthest from the rest of the contour. This may occur at various positions in the contour, but this set of feature points noticeably includes high curvature points. However, they carry more information than the sole extrema of the curvature. This is at the heart of a relevant and compact representation of the contour for an efficient recognition; contours that carry very little curvature extrema cannot be analyzed properly. Here, on the contrary, there is a nice distribution of peaks and troughs that enable a rich feature representation of the contour. Finally, by keeping only the extrema, a large part of the information carried by the other parts of the signature is eliminated. This pruning is relevant because we postulate that the informative content is gathered within the relative positions of these extrema.



(a) Feature points in signature.

(b) Corresponding positions in the contour.

Figure 3.12: Feature points in our example contour. Note that these extrema are closely related to the high curvature points, but give however far more information than a simple curvature signature.

Out of the selected feature points, our idea is to generate feature pairs. The main argument behind such a choice is the following: points by themselves do not carry sufficient information; there is no way to compare two points if there is a shift between them. Pairs of points add structural information about the signature and the contour, and because the signature f is very robust to rigid deformations, the pairs will remain very stable.

The generation of pairs is done by reading the feature points in the signature from left to right. For each point, the algorithm looks ahead within a certain window inside which the second point is selected. An example of such pairs can be found in Figure 3.13. In practice, we set this parameter to 20.

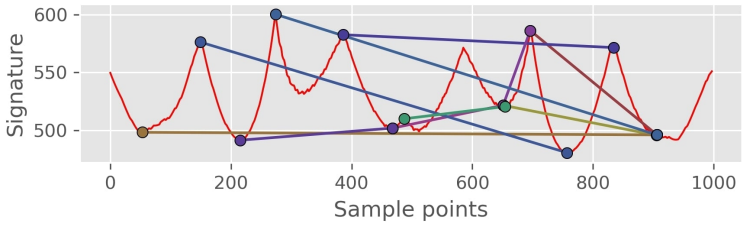


Figure 3.13: Examples of pairs of feature points (10 are presented). Pairs add structural information to the positional information of the feature points.

To handle the case of the rotation of a closed contour and thus a shift of the signature, one needs to take into account the points in a circular manner. The list of ℓ feature points is appended with the first ℓ' elements, making it a list of $\ell+\ell'$ feature points. This way, when processing the last points of the list, we create pairs with the first feature points of the list, thus perpetuating the continuity in the generation of pairs. In the same fashion, we calculate another list of feature pairs for the backward signature.

3.3.4 Contour storage

Now that the contour features have been gathered in the form of pairs of signature points, we introduce in the next sections the data structure into which these features will be stored. We begin by describing structure and its mechanism; we then explain how the previously defined signature is encoded into it.

3.3.4.1 Core structure

We choose to store the information in a system of associative arrays that serves as a tool for an efficient retrieval method. An associative array is a data structure consisting of a collection of (key, value) pairs. Such an array is assimilated to its key function h which enables the conversion of an input value from a descriptor to an integer key value. A whole database of feature pairs, each belonging to a certain contour, can thus be stored in this structure. In essence, this is very close to the concept of hash tables. Some

works have used hashing for contour recognition and clustering [61, 53] or for finding unusual contours [102]. However, our technique allows to retrieve contours among a large database and does not require learning any clustering pattern. Although the hashing technique has also been used in the Shazam algorithm [99] by which our method was inspired, the difference is that we do not explicitly create hash functions *per se* because the associative functions described below are not injective.

To be more specific, h is a set of nested associative arrays. Nested arrays are a set of associative arrays such that the existing values associated with each key contains another associative array. Such a set of nested arrays is referred to as a table. Each bucket (not empty) of the first array h_0 contains one independent array, which is indexed by the second key function h_1, and so on. This system of nested arrays is akin to a dense tree structure, in which the retrieval is progressively refined. Figure 3.14 graphically explains the structure of our system and presents a query example. In our case, we use three different keys and therefore the system has a depth of 3.

The last array buckets should contain information about the contour. Therefore, each time a new entry is added to the whole array (Algorithm 4), it is possible to append the information to a list already containing previous data. The associative array h_v does not handle collisions – there are permitted and fosters multiple retrieved values per query – and the whole storage system allows to retrieve several potential contour indices for each query pair (Algorithm 5). Also, if no corresponding pair was found – which should often occur if the contour is unknown in the database – then the value None is returned.

The associative array is composed a set of buckets (or bins) indexed by a key function and containing a certain value. Let $h = \{h_0, h_1, ..., h_{n_v-1}\}$ $(n_v \in \mathbb{N}^*)$ be the set of associative arrays. Each array does not have a fixed number of buckets. In the following, each array is assimilated to its key function so that $\forall v = 0, ..., n_v - 1,\ h_v \in \mathbb{Z}^{\mathbb{R}}$.

3.3.4.2 Encoding the signature in the arrays

Every time a pair is stored in an array, the bucket indexed by the key might not exist and is therefore created in the process. For each pair, there is a descriptor $\mathcal{D}f = \{d_0, d_1, \cdots, d_{n_v-1}\}$ with n_v components. Each key function takes as an input respectively each component d_v of the descriptor. Let $q = \{q_0, q_1, ..., q_{n_v-1}\}$ be a set of n_v positive values. These values are quantization parameters of the key functions defined in Equation (3.10):

$$\forall v = 0, ..., n_v - 1, \quad h_v(d_v) = \lfloor q_v \cdot d_v \rfloor \tag{3.9}$$

The parameters q virtually allow to increase or decrease the width of each bucket in the arrays. This way, one can manipulate those parameters in order to get a finer or coarser storage. Small perturbations can be taken into account by setting a sufficient coarseness.

For our application and for any pair of extrema $\{(s_A, f_A), (s_B, f_B)\}$ defined in Equation (3.8), we use a descriptor of three components $(n_v = 3)$ defined by Equation (3.10)

Algorithm 4: Adding a pair descriptor to the associative arrays

Input : h: table of associative arrays
$\{q_0, q_1, ..., q_{n_v-1}\}$ quantization parameters
$\mathcal{D}f = \{d_0, d_1, ..., d_{n_v-1}\}$: input descriptor
j: index of the contour
s_j: curvilinear abscissa of the first point of the pair

$L \leftarrow h_0$
for v *from 0 to* $n_v - 1$ **do**
  $\widehat{d_v} = h_v(d_v)$
  if $v = n_v - 1$ **then**
    if $L[\widehat{d_v}]$ *does not exist* **then**
      $L[\widehat{d_v}] \leftarrow$ empty list
    $L[\widehat{d_v}] \xleftarrow[\text{append}]{} (s_j, j)$
  else
    if $L[\widehat{d_v}]$ *does not exist* **then**
      $L[\widehat{d_v}] \leftarrow$ empty array
    $L \leftarrow L[\widehat{d_v}]$

along with a set of parameters $\{q_0, q_1, q_2\}$ such that $q_0 = 0.1$, $q_1 = 0.1$ and $q_2 = 0.5$. The keys to the arrays are then calculated according to Equation (3.9).

$$\mathcal{D}f = \{f_A, f_B, s_A - s_B\} \tag{3.10}$$

In the last array, we store a list of tuples consisting of the abscissa of the first point s_j and the index of the contour j.

Thus, as illustrated in Figure 3.14, $\lfloor q_0 \cdot d_0 \rfloor = \lfloor q_0 \cdot f_A \rfloor$ is the key to access the corresponding bucket in the first array h_0, which contains the next array h_1 for which the second key $\lfloor q_1 \cdot d_1 \rfloor = \lfloor q_1 \cdot f_B \rfloor$ is used. Finally, at the bucket indexed by this location, the last array h_2 can be found, for which the key $\lfloor q_2 \cdot d_2 \rfloor = \lfloor q_2 \cdot (s_A - s_B) \rfloor$ leads to the tuple (s_j, j), which corresponds to the location of the match within the retrieved contour of index j.

Because the signature is deemed sufficiently stable, one can discriminate the ordinates easily, hence the choice of the first two components. Moreover, the third component of $\mathcal{D}f$ allows to neglect the impact of shift along the signature abscissa, thereby making the descriptor invariant to the starting point. The descriptors are thus very reliable, provided the signature f is sufficiently robust.

3.3.4.3 System of parallel tables

It is possible to use several tables at once. We define a system of tables as a set of parallel and independent tables. We can use several signatures, all extracted from the

Algorithm 5: Request on the associative array

Input : h: table of associative arrays
$\{q_0, q_1, ..., q_{n_v-1}\}$ quantization parameters
$\{d_0, d_1, ..., d_{n_v-1}\}$: input descriptor
Output : list of tuples

$L \leftarrow h_0$
for v *from 0 to* $n_v - 1$ **do**
  $\widehat{d_v} = h_v(d_v)$ `// Equation (3.9)`
  if $L[\widehat{d_v}]$ *does not exist* **then**
    return None
  $L \leftarrow L[\widehat{d_v}]$
return L

TCD descriptor [105] and processed separately, all in the same fashion, both for storage and retrieval. Let $\mathcal{H}$ be a system of tables. The complete definition of the signature presented in Equation (3.6) allows for the use of several levels of description by varying the frequency ω. The process of retrieval is therefore to consider the set of signatures characterizing the contour at several levels of description $f_s(\omega = 0)$, $f_s(\omega = 1), \cdots$ and perform retrieval on each pair of each signature. The advantage of using more signature frequencies is that it allows to be more and more exact in the recognition. An illustration of this configuration is presented in Figure 3.15. Each table of nested arrays is treated in parallel to the others. The storage and retrieval processes are treated independently for each table h^u according to the principle illustrated in Figure 3.14. In the following, we will continue to use only one table in order to explain the technique while avoiding heavy notations.

3.3.5 Low Complexity Arrays of Contour Signatures

This presentation of the system of tables is indeed the core structure of our retrieval. At this stage, it is of course possible to perform a retrieval by considering the complete set of pair descriptors for a query contour. However, it is possible to refine and harmonize the process. This section completes the mechanism of retrieval. It consists of two general steps: finding the matches in the database and aligning them with a binning technique. The complete method, from the use of contour signatures to the storage system and the retrieval, is coined Low Complexity Arrays of Contour Signatures, or LACS.

3.3.5.1 Pulling correspondences from h

Let Λ be the database of contours stored in h. The contour of index j in the database is Λ_j, $j \in \{0, 1, ..., n_\Lambda - 1\}$ and n_Λ is the number of contours in the database. The set Λ_j can be considered as the set of all pairs (or pair descriptors) describing the contour of

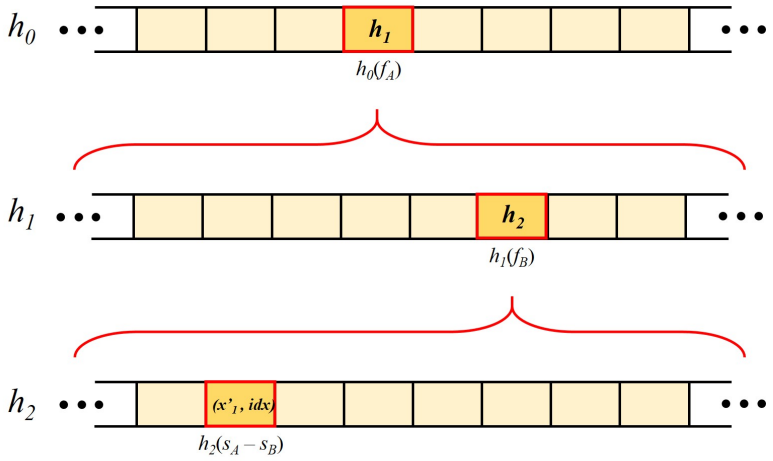


Figure 3.14: Structure of our system of associative arrays. Each bin of each array, indexed by its key function, contains a sub-array. For a query pair of signature points A and B and its triplet descriptor $(f_A, f_B, s_A - s_B)$, the result is found by successively finding the corresponding bin of the arrays, as long as it exists. The last array contains the result of the retrieval, which is the abscissa of the first point of the retrieved feature pair s_j initially stored (Algorithm 4) and the index of the corresponding contour j.

index j. Consider a query, indexed by $\mathcal{Q}$, which is a collection of pair descriptors from a query contour, and the task of comparing it with the contours already processed and stored in h.

The algorithm goes through all the pair descriptors in $\Lambda_\mathcal{Q}$ and, for each of them, retrieves a list of tuples of abscissa and index that we write $\{(s_j, j)\}$ for clarity. If the retrieved value is None, the next descriptor of $\Lambda_\mathcal{Q}$ is processed, ending up retrieving all the potential contour candidates for each investigated pair. Still, the retrieved values lack information – only one abscissa value s_j along with its corresponding contour index j for each – and may be numerous for each query pair, as a consequence of the collisions. Therefore an additional strategy is employed to select the relevant ones and prune the acquired set.

3.3.5.2 Pairwise geometric alignment

If s_j and $s_\mathcal{Q}$ are respectively the abscissa in the current retrieved tuple and the first abscissa of the current pair in the query, we define:

$$\delta s \triangleq s_j - s_\mathcal{Q} \tag{3.11}$$

Equation (3.11) refers to the shift needed to map the query point indexed by $s_\mathcal{Q}$ onto the contour point indexed by s_j. An example of such calculation is provided in Figure 3.16. For any part of the signature which matches the query, the resulting values of δs should be close to one another. The values of δs are quantized (Equation (3.12))

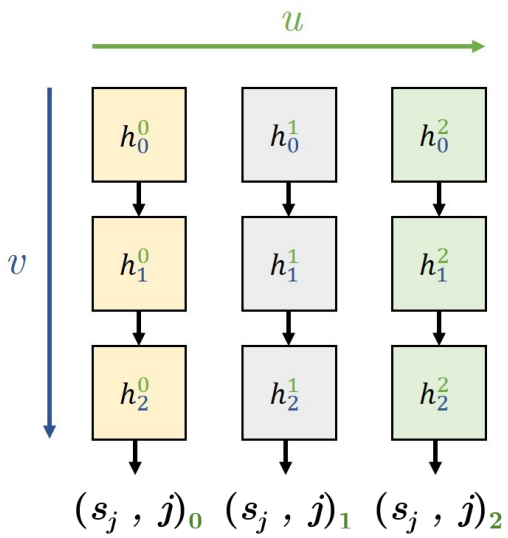


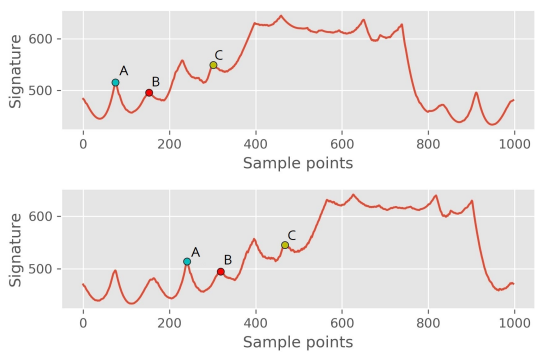
Figure 3.15: Example of a system $\mathcal{H} = \{h_v^u \mid u \in [0,2]\ ,\ v \in [0,2]\}$. Each h^u is a table composed of v nested arrays. The retrieved values (s_j, j) may not concur, as they are pulled from different signatures; even though they characterize the same contour, they are independently processed.

and we calculate histograms of δs per contour of index j in the database. The bins of the histograms range from $\delta s_{min} = -1000$ to $\delta s_{max} = +1000$ and have N_B bins.[6] The resulting binning is expressed by Equation (3.12). The number of bins is again a parameter for the finesse of the matching; values ranging from 100 to 500 give fairly good results and here we choose 200 bins. We determine the quantized values of δs for each retrieved tuple, and we successively identify the score of the current contour η_j as the value of the largest bin. The scores are stored in an array indexed by the contour indices.

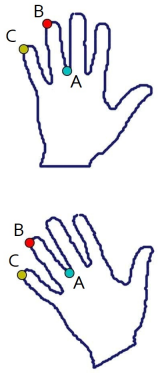
$$\Delta s = \left\lfloor N_B \cdot \frac{\delta s - \delta s_{min}}{\delta s_{max} - \delta s_{min}} \right\rfloor \tag{3.12}$$

Figure 3.17 illustrates the behavior of such histograms. If the original contour is matched against itself, the correspondences are perfect, and therefore the retrieval yields a maximum number of pairs. This number is far larger than the number of pairs in the query because it is likely to find several results corresponding to one input descriptor. As explained in Section 3.3.2.2, noisy contours disturb the signature signal and can thus corrupt the retrieval and rotation is the reason for an important shift in the signatures which can be seen in the histogram. There is a significant drop in the number of pairs

[6] The choice of the boundary is based on the chosen sampling of the contours. Since s ranges from 0 to 999, δs ranges roughly between -1000 and 1000. This range has to be set and fixed in advance in order for the bins to all have the same width for any contour.



(a) Points in signatures.



(b) Corresponding points in contour.

Figure 3.16: Example of the calculation of δs for a rotation of $\frac{\pi}{6}$. The given points A, B and C have been shifted due to rotation. Here $\delta s = 166$ in each case. Even if there were slight variations (bounded by $+2.0$ (excluded) with $q_3 = 0.5$), those values would still belong to same bin in the histogram.

retrieved, but nonetheless a very coherent shift is found.

3.3.5.3 Selection of the best contour

The aforementioned pipeline of techniques is compiled into a single Algorithm 6. There are essentially two passes to compute the scores: one for the forward query and one for the backward query. At each iteration, we retrieve the correspondences from the database and update the score and the best contour associated using the steps described in Section 3.3.5.2. We only keep the largest score for each bin of each contour. The best score, corresponding to the largest number of pairs retrieved over all contours in Λ is $\widehat{\eta} = \max_j \eta_j$. Finally, the index of the best contour is given by $\widehat{j} = \arg\max_j \eta_j$. More mathematical details about the choice of the best index can be found in Appendix **??**.

3.3.5.4 Computational complexity

The descriptors and the quantization produce collisions during the storage. This allows to get more than one possible contour corresponding to the query and also several pairs belonging to the same contour. Of course, the main issue is that it causes a linear computational complexity instead of a purely constant one. In theory, for one table of associative arrays, each having b_v buckets, the mean number of collisions can be assumed to be uniformly distributed and equals to Equation (3.13), where $|\Lambda| = \sum_j |\Lambda_j|$ is the

Algorithm 6: Finding matches in the database from a contour query

Input : h: table of associative arrays
$\Lambda_{\mathcal{Q}} = \{\mathcal{D}f_0, ..., \mathcal{D}f_{|\mathcal{Q}|-1}\}$: list of pair descriptors from the query contours
q: quantization parameters

Output : $\widehat{j}$: best contour index
$\widehat{\eta}$: best score

$hist \leftarrow$ empty dictionary
$\widehat{\eta} \leftarrow 0$
$\widehat{j} \leftarrow$ None

for *s from 0 to* $|\mathcal{Q}| - 1$ **do**
    $L \leftarrow get(h,\ q,\ \mathcal{D}f_s)$ `// Algorithm 5`

    if *L is not None* **then**
        for *all tuples* $(s_j,\ j) \in L$ **do**
            $\delta s \leftarrow s_j - s$ `// Equation` (3.11)
            $\Delta s \leftarrow quantize(\delta s)$ `// Equation` (3.12)

            $B = hist[j][\Delta s]$ `// ensure it exists first`
            $B \mathrel{+}= 1$

            if $B \geq \widehat{\eta}$ **then**
                $\widehat{j} \leftarrow j$
                $\widehat{\eta} \leftarrow B$

return $\widehat{j}$, $\widehat{\eta}$

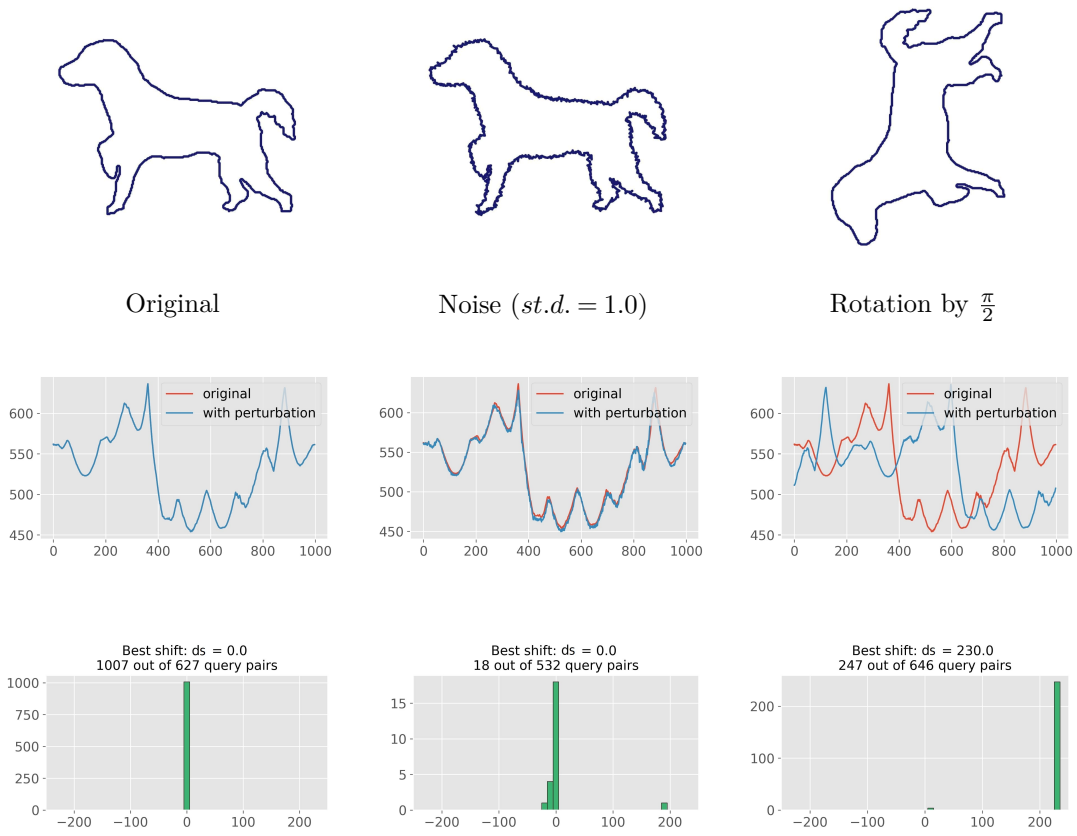


Figure 3.17: Results of the alignment for different perturbations. The first row displays the contour extracted from the transformed contour, the second row shows the corresponding signatures and the third row presents the resulting histograms. It should be mentioned that the noise in this case is quite large and represents a worst-case scenario.

total number of pairs stored in the database Λ.

$$\gamma = |\Lambda| / \prod_{v=0}^{n_v - 1} b_v \tag{3.13}$$

The total number of pairs $|\Lambda|$ is substituted by the total number of contours in the database n_Λ since the former increases linearly with n_Λ. Using Equation (3.13), the complexity becomes:

$$C(n_\Lambda) = \mathcal{O}(\gamma \cdot |\Lambda_\mathcal{Q}|) = \mathcal{O}\left(\frac{n_\Lambda}{b^{n_v}} \cdot |\Lambda_\mathcal{Q}|\right) \tag{3.14}$$

where $|\Lambda_{\mathcal{Q}}|$ is the number of pairs in the query and if we suppose $\prod_v b_v \approx b^{n_v}$. Equation (3.14) shows that there is a linear relation between the complexity and the number of pairs $|\Lambda_{\mathcal{Q}}|$, but there is also a factor which, if well-chosen, can severely diminish the overall computational time. The value of b depends on the descriptor definition and also on the parameters q; a carefully chosen set of parameters thus allows to limit collisions, to ensure that γ is relatively constant when $|\Lambda_{\mathcal{Q}}|$ (and the number of contours) varies and to only pick pairs closely related to the ones investigated as queries. We will demonstrate experimentally in Section 3.4.1.2 that we manage to keep the runtime low and γ quasi-constant.

3.3.6 Theoretical interpretation of the LACS system

The definition and the implementation of the LACS in Chapter 2 comes without interpretation. So far, it appears as a mixture between audio retrieval in image processing. The initial remark of Wang [99] is that the cross-correlation between two signals is not robust at all and is, in particular, very sensitive to any stretch of the abscissa (time or space). In the realm of image processing, it has also been noted with the example of the signature of [29]. The LACS is inspired by the Shazam algorithm, which makes it really efficient for retrieval. In any case, the core principles are the use of associative arrays and the geometric alignment. We argue that this produces a form of correlation in which these two principles elegantly combine to retrieve specific points which enables a retrieval.

3.3.6.1 Retrieved pairs

Let us denote by f a shape signature, i.e. a 1D descriptor of the shape, which is a function of the curvilinear abscissa s. From this signature, we extract a set of pairs and an associated set of descriptors. We denote the set of indices of such a set $\mathcal{I}$. Out of every pair, we retrieve a potentially matching pair from the database.

Each retrieval returns the index of the corresponding shape in the database. We virtually build an histogram in the process to find the best accumulation of pairs corresponding to the same alignment (shift) within the signatures. Thus, for a particular shape having the signature g, indexed by an abscissa u, in the database, we have a set of pairs $\mathcal{I}_k$ in which all pairs are aligned. k is the index of the corresponding bin in the histogram (the set of bins is B).

Mathematically, this translates into:

$$\exists k \in B, \forall i \in \mathcal{I}_k, \begin{cases} s_i - s'_i = u_i - u'_i. \\ f(s_i) = g(u_i) \\ f(s'_i) = g(u'_i) \end{cases} \tag{3.15}$$

Note that the $\mathcal{I}_k$ form a partition of $\mathcal{I}$:

$$\bigcup_{k \in B} \mathcal{I}_k = \mathcal{I} \tag{3.16}$$

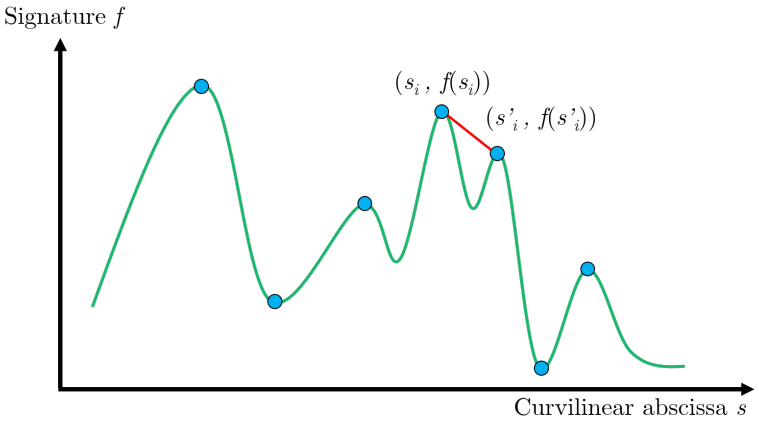


Figure 3.18: The signature f and its extrema. One pair of the set that has been retrieved and aligned is depicted in red and is indexed i.

Moreover, we can also infer from Equation (3.15) that there is a constant c_k which designates a shift between both abscissae s and u such that:

$$s = u + c_k$$

3.3.6.2 Sparse correlation

From Equation (3.15), multiplying by $f(s_i)$:

$$\forall i \in \mathcal{I}_k, \quad f(s_i)^2 = g(s_i + c_k) \cdot f(s_i) \tag{3.17}$$

Summing over all $i \in \mathcal{I}_k$,

$$\sum_{i \in \mathcal{I}_k} f(s_i)^2 = \sum_{i \in \mathcal{I}_k} f(s_i) \cdot g(s_i + c_k) \tag{3.18}$$

And therefore,

$$\boxed{(f \otimes f)_{|\mathcal{I}_k}(0) = (g \otimes f)_{|\mathcal{I}_k}(c_k)} \tag{3.19}$$

where $\otimes$ denotes the correlation product.

Although trivial to explain, the choice of specific feature points (extrema here) in the signature leads to a sparse cross-correlation equivalent to the auto-correlation of f. Assuming the values of the signatures stay within a reasonably stable range of values, this product would allow to create an order amongst the retrieved shapes, leading to the choice of the best one such that:

$$\forall k \in B, \quad (f \otimes f)_{|k_{best}}(0) \geq (f \otimes f)_{|k}(0)$$

3.3.6.3 Choice of the best shape

Again, we make the assumption that the set of values in the signature f features no significant differences in the range of values in every parts, meaning that any subset of values of f roughly has the same range of values. We also make the assumption that f is equivalent to a distribution centered around 0, meaning the mean value of f and each of its subsets is 0.[7]

In this scenario, given that $\mathcal{I}_k \subset \mathcal{I}$, the variance V is equal between each set of pairs $\mathcal{I}_k$, and:

$$\forall k \in B, \quad V(f)_{|\mathcal{I}_k} = V(f)_{|\mathcal{I}} \tag{3.20}$$

Rearranging the Equation (3.20), we get:

$$\frac{1}{|\mathcal{I}_k|} \sum_{\mathcal{I}_k} f(s_i)^2 = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}} f(s_i)^2$$

Hence:

$$\boxed{\frac{|\mathcal{I}_k|}{|\mathcal{I}|} = \frac{\sum_{\mathcal{I}_k} f(s_i)^2}{\sum_{\mathcal{I}} f(s_i)^2}} \tag{3.21}$$

where $|\cdot|$ is the number of elements of a set.

This is actually an equivalence, since if the number of elements in $\mathcal{I}$ is 0, there is no subset to analyze, and this means that $k \notin B$ (there nothing retrieved).

This result is exactly what we wanted since we connected the number of pairs retrieved in each bin to the sparse auto-correlation of the signature. We deducted in Section 3.3.6.2 that it will allow us to perform a simple comparison to determine the best bin. Since our goal is to make comparisons between the values of $|\mathcal{I}_k|$ for each k and for each shapes, we can ignore the denominator (it is always the same because f is the query signature). For each shape S, we can identify the largest bin as a score:

$$n_S = \max_k |\mathcal{I}_k| \tag{3.22}$$

Finally, all we need to do is to calculate the best score as the maximum score among all retrieved candidate shapes:

$$\boxed{S_{best} = \arg\max_S n_S} \tag{3.23}$$

3.3.7 Recognizing coins with contours

Putting together the first two parts of this chapter, it is now time to organize a coin recognition task. More specifically, the goal is to associate coins from the same die by

[7] If the mean is $\mu > 0$, as it can be the case, we can always get back to the following results by using $\hat{f} = f - \mu$ and $\hat{g} = g - \mu$, starting in Section 3.3.6.2. The mean is however assumed to be always the same for any signature.

pairing meaningful contours. In the following, we assume that the dataset has been pre-processed so that only sets of contours have to be analyzed.

The common situation is that a database has already been built, containing all the contours stored in the form of TCD signatures in a LACS system $\mathcal{H}$, along with the index of the coin they characterize. Furthermore, we would like to implement a 2D geometric alignment between the detected pairs of contours. Contours do not have a single simple localization, but they do have either two ends or none. For two ends 1 and 2, four values are considered: x_1, y_1, x_2 and y_2. If there is no end, then the coordinates of the first point (arbitrary) and the middle point (half the length, starting from the first point) are taken. The principle, then, is exactly the same as in Section 3.3.5.2, except it happens outside the LACS retrieval algorithm. The best way appreciate the final recognition algorithm is that it is a retrieval algorithm very similar to Algorithm 6, but where line 6 is replaced by the value returned *by* Algorithm 6. This is summed up in Figure 3.19. The major differences are:

- The returned values of Algorithm 6 have to include not only the index of the coin j and the retrieval score $\widehat{\eta}$, but also the pair of coordinates (x_1, y_1, x_2, y_2). This is easy to do since those values can be stored via Algorithm 4 in addition to s_j and j.
- There are four shifts to consider, so there are four quantizations to compute. Each quantization can be performed either by an equation similar to Equation (3.12), or it can be of the form $\lfloor \frac{\delta_x}{\varepsilon} \rfloor$, where ε is the desired spatial tolerance. In our experiments, we choose the latter, with $\varepsilon = 100$ px.

The final score for the best match is obtain by summing all scores obtained from each match gathered in the largest bin $\widehat{B}$ of the final alignment histogram. So, if the score of a single LACS retrieval is $\widehat{\eta}$, the total score is given by Equation (3.24):

$$\eta = \sum_{\widehat{B}} \widehat{\eta} \tag{3.24}$$

3.4 Experiments and results

In this section, various experiments are presented. Firstly, we provide some statistical results about LACS; secondly, LACS are applied to coin identification; and thirdly, LACS are combined with coins contours to implement a die classification.

3.4.1 Experiment 1: LACS performances

To assess the performance of exact contour retrieval, studies on both robustness to perturbations and retrieval time are provided. Our technique is compared to several methods from the state of the art, including [14] as a standard baseline strategy with descriptors, [29] to evaluate the use of cross-correlation and [105] because their method

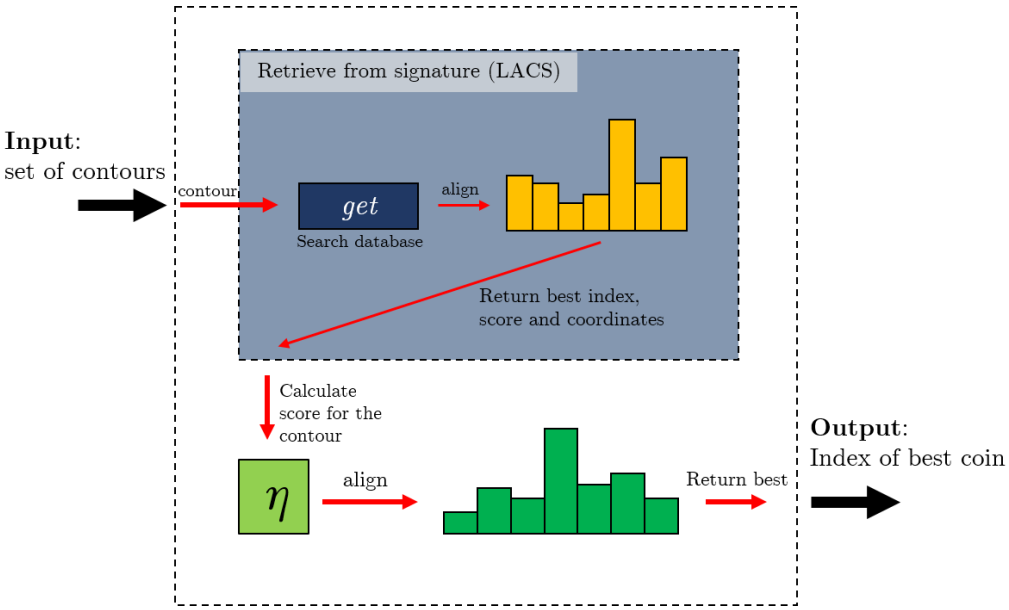


Figure 3.19: Visualization of the complete coin retrieval system. Algorithm 6 is used to match a single contour against the database. All the information retrieved are then processed in a second part, looking at the contour's score and position. Of course, every contour has to go through the same process, so that the output can be computed from the final alignment histogram.

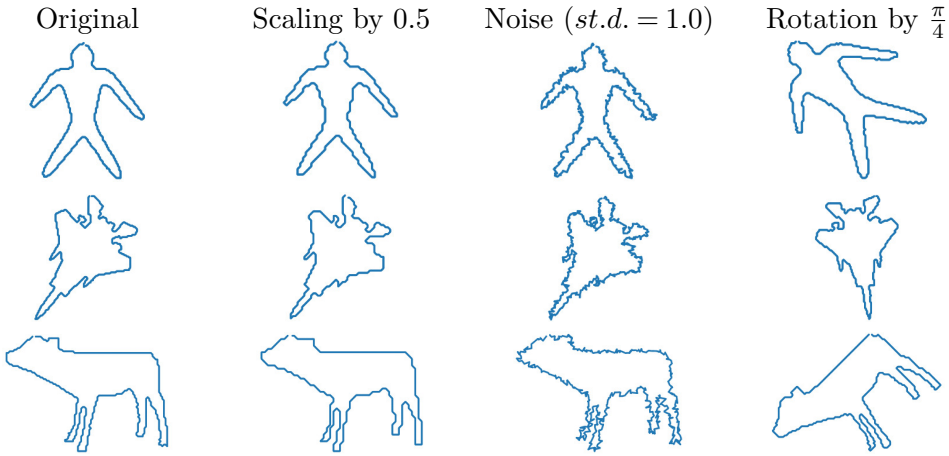


Figure 3.20: Original and perturbed contours of the Kimia99 dataset.

	Scaling ($factor$)	Rotation ($angle$)	Noise ($st.d.$)
Range	$[0.5, 2]$	$[0, 2\pi)$	$[0, 1]$
Amount	10 (+1)	16	11

Table 3.2: Parameters for each perturbation. A certain amount of values are uniformly chosen within a reasonable range. For the scaling, we include the factor 1.0, so that there are 11 values.

inspired our signature. In the graphs described below, our method is labeled "LACS n_u", where n_u corresponds to the number of tables used (see Section 3.3.4.3). These experiments are provided to ensure statistically sound results, which stem from a sufficient number of data. Hence, the datasets used in this subsection do not feature quasi-flat objects but deal with images of shapes from which a single contour is extracted.

3.4.1.1 Robustness

This subsection presents an analysis of the resilience with respect to scaling, rotation and noise. These perturbations were generated by a process explained in Section 3.3.2.2. As general protocol for this study, each transformed contour query is matched against the original contours. The set of parameters for the perturbations are presented in Table 3.2. We choose to use the Kimia99 dataset [87] as a database for two reasons: it features a reasonable variety of inter-class contour complexities, each class sometimes containing very similar contours, albeit without being a too large dataset, which leads to prohibitive computational times for some algorithms [29, 105, 14]. Figure 3.20 shows some original and perturbed contours of the Kimia99 dataset. The final values are accuracy scores corresponding to the bull's eye score for the retrieval of each independent contour, the retrieval per class and per parameter of perturbation.

Results in Table 3.3 show that our method outperforms both Cui's [29] and Yang's

[105], even with one associative array. Considerable improvements in robustness can be made by adding more associative arrays, but at the cost of more computational time. The shape contexts [14] achieves impressive results of robustness to noise, but is much less resilient for scaling and rotation. The TCD is designed to deal with non-rigid deformations, which obviously hinders its discrimination for very close contours. The accuracy per class is however still very high in general, meaning that mistakes were mostly made between very close contours (Figure 3.21). Cross-correlation [29] suffers from a very poor stability, as predicted. Our method achieves very good outcomes in general for the accuracies per transformation parameter. Scaling mistakes correlate with factors higher than 1.0. Image rotation involves drops in robustness that are more likely to appear for angles $\theta \not\equiv 0 \pmod{\pi/2}$. Finally, there is a rapid decline in robustness for higher noise standard deviations for all methods, except for the shape context, but we still manage to reach 68% for $st.d. = 1.0$.

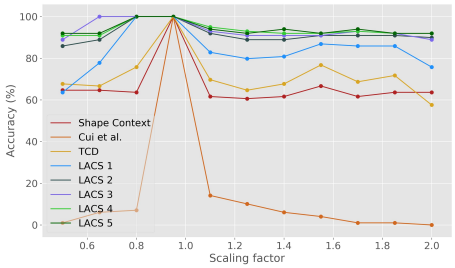
We show some further results in Table 3.3 for the MPEG-7 dataset [95] and the Tari-1000 dataset [6]. They are usual datasets for the evaluation of contour retrieval methods and contain respectively 1400 and 1000 contours. The results for LACS and TCD are comparable to those obtained on the Kimia99 dataset. As shown below, shape context and the method of Cui et al. have a quadratic time complexity. Hence, it is not possible to execute them on such large datasets in a reasonable time interval.

	Shape Context [14]	Cui et al. [29]	TCD [105]	**LACS (Ours)** (number of parallel arrays)				
Kimia99				**1**	**2**	**3**	**4**	**5**
Scaling	66.6	13.7	71.5	80.2	90.3	90.5	92.4	92.8
Rotation	35.0	13.9	88.9	88.7	94.0	95.0	95.6	96.3
Noise	95.8	26.8	63.5	66.4	77.3	82.4	85.5	86.9
MPEG-7				**1**	**2**	**3**	**4**	**5**
Scaling			70.0	86.7	89.9	90.5	91.3	91.1
Rotation			74.6	83.5	87.4	87.9	88.4	88.6
Noise			63.7	68.2	76.2	79.1	80.8	82.8
Tari-1000				**1**	**2**	**3**	**4**	**5**
Scaling			72.6	93.5	95.7	95.8	96.2	96.3
Rotation			87.4	86.0	89.9	90.4	91.2	91.5
Noise			75.3	65.6	77.1	80.7	83.0	85.1

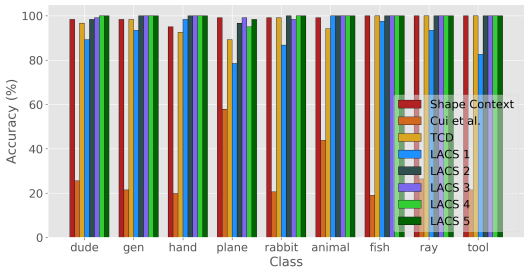
Table 3.3: Total accuracy of robustness (%) by method including various amounts of parallel associative arrays.

3.4.1.2 Runtime analysis

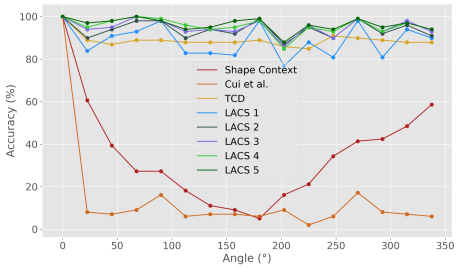
A runtime analysis is performed to evaluate the behavior of the associative arrays when changing the size of the database and the size of the query. We use the MPEG-7 dataset [95], which contains 1400 contours, because we want to evaluate the runtime for large databases. For the first experiment, we select a specific query containing $\Lambda_{\mathcal{Q}} = 380$ pairs,



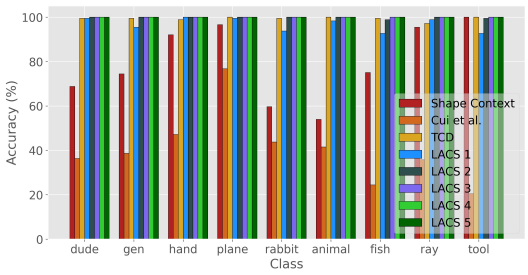
(a) per scale factor



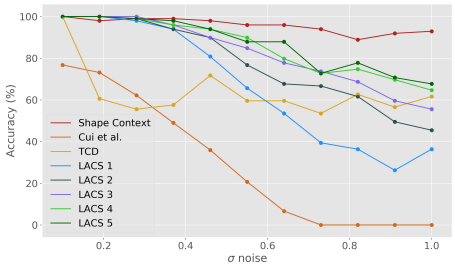
(b) per class (scaling)



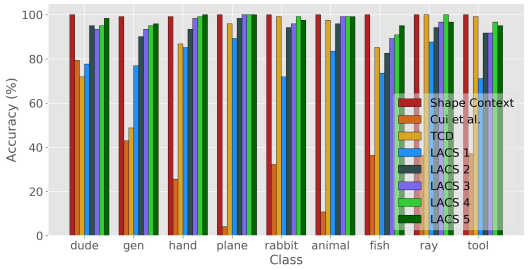
(c) per rotation angle



(d) per class (rotation)



(e) per noise *st.d.*



(f) per class (noise)

Figure 3.21: Detailed results of the robustness experiments. The left column shows accuracy against the perturbation parameters. The right column shows accuracy per class for each perturbation.

a value close to the mean number of pairs $\overline{\Lambda_{\mathcal{Q}}} = 374$. We perform successive retrievals against a database in which we iteratively increase the number of contours n_Λ; the same process is followed for every method. The second experiment is done by using the largest database (1400 contours are stored) and by computing the retrieval time for an increasing number of pairs in the contour. To do so, we take the contour with the largest number of pairs and perform a process similar to the first experiment by adding pairs. In each experiment, we do not take into account the creation of the descriptors, the signatures or the associative arrays; the database is already built and we examine only the retrieval time. Computations were run with an Intel® Core™ i7-7820HQ CPU @ 2.90GHz with Python 3.6.5 and the results are presented in Figure 3.22.

Although we indeed observe a linear tendency, the curve is very flat, making the computational time of the technique akin to a constant complexity. When the database is very small, there is a high probability of retrieving nothing (None in Algorithm 5) and so the retrieval time is negligible. This explains the soaring at first, which evens out at around 100 contours. The average number of collisions (calculated with Equation (3.13)) remains between $\gamma = 1.31$ and $\gamma = 2.92$, with a general mean of $\gamma = 2.27$. This implies the necessity to check, on average, 2.27 values (tuples) in Algorithm 6. The result is a very low retrieval time which spans between 3 ms and 13 ms, with an average of 10 ms. One can extrapolate that only 1 s is required to perform a retrieval with a query of 380 pairs among a database of around $n_\Lambda(t = 1\text{ s}) = 350000$ contours. Table 3.4 demonstrates the speed of our method with respect to more traditional approaches. On average, it takes less than 1 s to perform retrieval for any number of parallel associative arrays. By comparison with direct matching, our method is 10 to 1000 times faster for the range of n_Λ analyzed.

We also present the number of collisions with respect to the number of query pair descriptors $|\Lambda_{\mathcal{Q}}|$, which exhibit a linear correspondence between γ and the computational time. Figure 3.23 features the evolution of runtime against $|\Lambda_{\mathcal{Q}}|$, and explicitly shows a linear tendency.[8] The number of pairs $|\Lambda_{\mathcal{Q}}|$ is positively correlated to the complexity of the contour, as defined by the number of high curvature points and its length. Since we expect an average of $\overline{|\Lambda_{\mathcal{Q}}|} = 374$ pairs, the average computational time of around 100 ms for $n_\Lambda = 1400$.

Database size n_Λ	Shape Context [14]	Cui et al. [29]	TCD [105]	LACS 1	LACS 2	LACS 3	LACS 4	LACS 5
1	0.93	0.066	0.010	0.003 (1.31)	0.005 (1.43)	0.011 (1.50)	0.020 (1.53)	0.018 (1.55)
10	8.95	0.47	0.070	0.002 (1.47)	0.006 (1.50)	0.011 (1.63)	0.022 (1.66)	0.025 (1.68)
100	86.34	5.19	0.61	0.013 (1.52)	0.009 (1.52)	0.030 (1.61)	0.045 (1.93)	0.121 (2.31)
1000	823.61	65.82	6.44	0.011 (3.42)	0.011 (3.42)	0.193 (3.91)	0.241 (5.18)	0.901 (6.57)

Table 3.4: Runtimes of retrieval (in seconds) for several methods and different amounts n_Λ of contours in the database. Collisions γ for each n_Λ is also displayed between parentheses.

[8] This complexity is not presented as a function of the contour length ℓ. In theory, the number of pairs can be roughly expressed as $|\Lambda_{\mathcal{Q}}| \equiv \ell^2$. However, this relation represent an overestimation of the complexity because the number of pairs depends mostly on the number of feature points in the signature (that is naturally a linear function of ℓ), which are far less numerous.

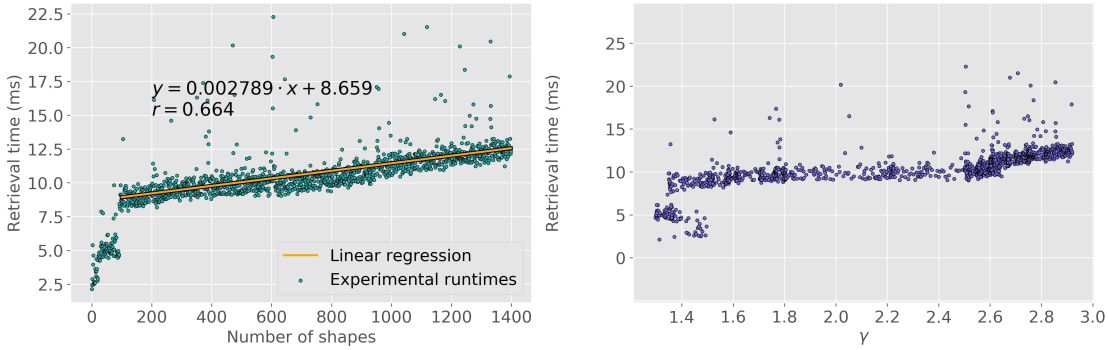


Figure 3.22: Left: Retrieval time evaluations with respect to the size of the database for our technique with one array, using the same query of $|\Lambda_Q| = 380$ pairs. A linear regression is shown beginning with $n_\Lambda = 100$. Right: retrieval time per collisions γ.

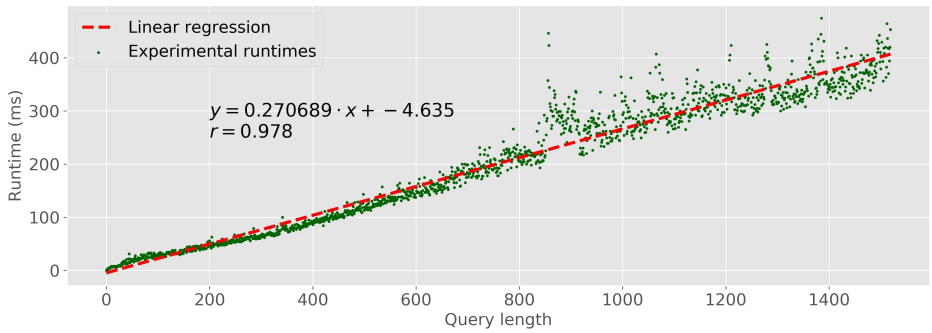


Figure 3.23: Retrieval time evaluations with respect to the size of the query, using the largest database. Runtimes are very low, showing a near constant complexity.

3.4.2 Experiment 2: Coin identification

A pertinent direct application of LACS is the task of identifying coins, which aims at recognizing a specific coin among an existing database. This is especially useful to prevent illegal coin trading and facilitate the recognition of stolen or fake coins. To this end, using the coin's perimeter is a good lead because it provides a unique signature; striking the soft metallic flan with a die did not constrain the coin's border.

3.4.2.1 Identification strategies

Although different, the perimeters are often quite circular, which makes them nonetheless very similar. The distinctive information which allows for distinguishing two coins is not obvious, but LACS works remarkably well with exact recognition, and the TCD signature is very adaptable to fine contour variations.

Previous work on such matter include notably the use of the deviation from circular

shape matching (DCSM) [48] to perform a preselection prior to the use of shape and feature descriptors to make the final decision. In order to comply with the preexisting literature, we use the same dataset, which is provided by the Fitzwilliam Museum of Cambridge, UK, and is a database of images of 240 coins acquired by various methods and under several conditions. There are 5 images per side per coin. Considering both obverse and reverse sides, there are 2400 coins. In Figure 3.24, two images of three coins are displayed. The contour variations can be subtle, but are nonetheless sufficient to identify the coins.

Figure 3.24: Six images of three coins taken under two different conditions. Each color designates a coin.

3.4.2.2 LACS configuration

Our aspiration is to perform the LACS retrieval directly on the database of the borders contours. The extraction of such contours is perform using the entropy segmentation [109], from which the periphery is kept and uniformly sampled to 256 points. A Gaussian filter of variance 1.0 is applied to the resulting contour coordinates.

The general principle for the storage is globally the same. The creation of pairs and pairs descriptors (see Section 3.3.3 is however slightly different, as we use a window of 3 feature points instead of 20. The quantization parameters are set to $q = (0.5, 0.5, 0.1)$, which corresponds to the initial choice for the previous experiments.

In this scenario we do not use the signature both ways. The contour is always unique and circular which automatically places the coordinates in the same way. On average, this saves half the time needed to test both directions, as in the previous experiments.

3.4.2.3 Fusion of coin sides

Since the available data is the object itself, we can easily use both sides to identify it. The general principle is the same, a contour signature is calculated for each side; the obverse and the revers are supposed to feature the exact same mirrored contour. The LACS structure now includes twice as many tables as desired for any number of signatures taken into account: it needs 2 tables for 1 signature frequency ω (one obverse, one reverse), 4 tables for 2 signatures, and so on. The idea is that the data stored in the LACS database is richer and gives more information for exact retrieval. However, this also means more collisions and therefore longer run time.

3.4.2.4 Results and performance

The identification accuracy is calculated using the bull's eye score. The database is already stored and the retrieval algorithm neglects the coin index used as a query. The experiments were run for the obverse, the reverse and both sides at the same time, using the procedure described above.

Algorithm	DCSM [48]	LACS 1	LACS 2	LACS 3	LACS 4	LACS 5
Obverse						
Accuracy (%)	98.0	53.7	79.3	83.4	87.0	88.3
Run time (s)	7.19	0.041	0.213	0.243	0.383	0.586
Reverse						
Accuracy (%)	96.1	44.2	71.4	73.6	77.3	81.5
Run time (s)	7.19	0.051	0.206	0.214	0.305	0.59
Both						
Accuracy (%)	98.8	61.4	83.4	86.7	88.5	89.6
Run time (s)	14.4	0.107	0.471	0.522	0.768	1.72

Table 3.5: Identification scores and run times per coin using the obverse, the reverse and the fusion of both sides.

Our method has been used with five amounts of tables n_u. Again, $n_u = 5$ gives better results, but costs more run time. The DCSM [48] performs already very well on this dataset. However, the algorithm has a linear complexity to match one coin: this leads to quite impractical calculation time when the database is large. We extrapolated the value of the run time for the DCSM from the author's estimation of the time required from one match between two coins, which is 0.006 s with their configuration (Intel® Core™ 2 CPU with 2.5GHz). For 1200 comparisons, it therefore amounts to 7.19 s. We also evaluated the one match run time on our machine (Intel® Core™ i7-7820HQ CPU @ 2.90GHz). It is approximately 0.013 s, which doubles the total run time.

LACS represents a good retrieval strategy for ancient coins identification since it is able to deal with exact contour recognition. Although overall less accurate than the DCSM, it performs the evaluation in a very competitive run time. The lack of accuracy

is regrettable but it is a fair trade-off in relation to the algorithm speed. The reason for these accuracy scores is that the amount of pairs generated is inadequate for the LACS to work very well; the shape is simply not complex enough. Nonetheless, this simple demonstration shows that LACS can be applied to such application with a significant accuracy and low retrieval time.

3.4.3 Experiment 3: Die recognition

To begin with, experiments are run on the artificial dataset presented in Chapter 2, Section 2.4. This dataset is very useful for deriving statistical results about our technique. Only a sufficient part of the dataset will be used: it comprises 16 types, 5 dies per type and 5 coins per die. The process which has been explained in Section 3.3.7 is employed for each test.

3.4.3.1 Parameters

The parameters presented in Table 3.6 are default parameters, but they may however be appointed as variables in some experiments. Note that these parameters can slightly differ from those used in the experiments of Section 3.4.1. The quantization parameters chosen in Section 3.3.4.2 have been found to cause a dramatic amount of collisions, so these new values are more appropriate. Moreover, the two first elements of q have been linked with the same variable because they both describe an ordinate in the signature. The range parameters have been adapted to the length of the contours.

Parameters	Values
q	$(q_0 = 1,\ q_0 = 1,\ q_1 = 1)$
δs_{min}, δs_{max}	-100, 100
N_B	200
ε	100 px

Table 3.6: Parameters for the experiments

3.4.3.2 Retrieval quality

Before going into numerical detail, some visual results are provided so as to assert the relevancy of the retrieval. These examples can be seen in Figure 3.25, which presents the outcome of the retrieval for 4 different query coins. Both skeleton map and Canny edge map were used to extract contours and no locality constraints were applied for the retrieval. In general, every contour match is valid (good correspondence). Clearly, a vast majority of the available contours in the coin are not retrieved, which is most likely due to the variations in the energy map provoked by deterioration of the die. See for example the case C whose query was blurred to simulate wear. Sharp energy maps with precise ridges inevitably lead to unambiguous contours, which is expected from this dataset. In particular, the borders of the engravings are the most retrieved parts of the

coin, provided there are sharp enough (see case B in particular). Another particularity of this dataset is that it does not put the emphasis on coin deterioration leading to occlusions; only additive noise and texture have been used (see Section 2.4). Therefore long contours can be found easily and they inevitably carry a significant information which makes them likely to be matched and score high. The main difference between the use of the skeleton or the Canny edge map is the number of contours matched; the former seems prone to yield more sub-matches than the latter, which, in case B, leads to as few as 1 contour found.

3.4.3.3 Numerical evaluation

To assess numerically the quality of the retrieval, we begin by setting an experiment in which we use each coin as a query to request in the database (from which it has been excluded) and proceed to compare the accuracy of the retrieval – the number of correct predictions divided by the number of coins – and the computational time per coin. The control parameter is the quantization q, where only q_0 is tweaked. This has been decided arbitrarily as q_1 impacts the ordinate of the TCD signature, which we want to be precise. The value of q_0 has to do with the location of signature peaks along the contour, for which we can afford to be more tolerant. 20 values have been equally spaced across the interval $[0.5, 5]$. Both cases where the retrieval is locally constrained ($\varepsilon < +\infty$) and where there are no locality constraints are tested.

On average, the scores of retrieval are $\eta = 61 \pm 64$ with the skeleton and $\eta = 17 \pm 18$ with Canny. These numbers are in accord with the remark made in Section 3.4.3.2: using the skeleton map to extract contours leads to more possible candidates for the retrieval and therefore generates statistically higher scores on average than when using the Canny edge map. The variations in scores are large, but this is mostly due to the initial potential of each energy map (case B in Figure 3.25 vs case D for example).

Accuracy and run times results can be examined in Figure 3.26. The accuracy reaches as high as 72.8% in the case where the coin skeleton is used and 72.5% with the Canny edge map. In both cases the slope tends to stop favoring the accuracy at the cost of run time at $q_0 \approx 1.5$. This value corresponds to the "elbow" of the graph, for which the values are (0.0065 s, 70.0%) with the skeleton and (0.16 s, 70.6%) with Canny. One can notice that there seems to be no difference between the results obtained with and without locality constraints. The root mean square error between both sets of data points are indeed very low: 1.2% (skeleton) and 0.3% (Canny). It appears therefore that, for this dataset at least, imposing that the contour be positioned harmoniously relative to the other coin is not required.

The run times are overall very low. This is especially true when the Canny contours are used. Needless to say, those results are correlated with the number of contours per coin. In the Canny version, there are, on average, 7 contours per coin, which is extremely low. This value, compared with the results in Table 3.4, would predict around

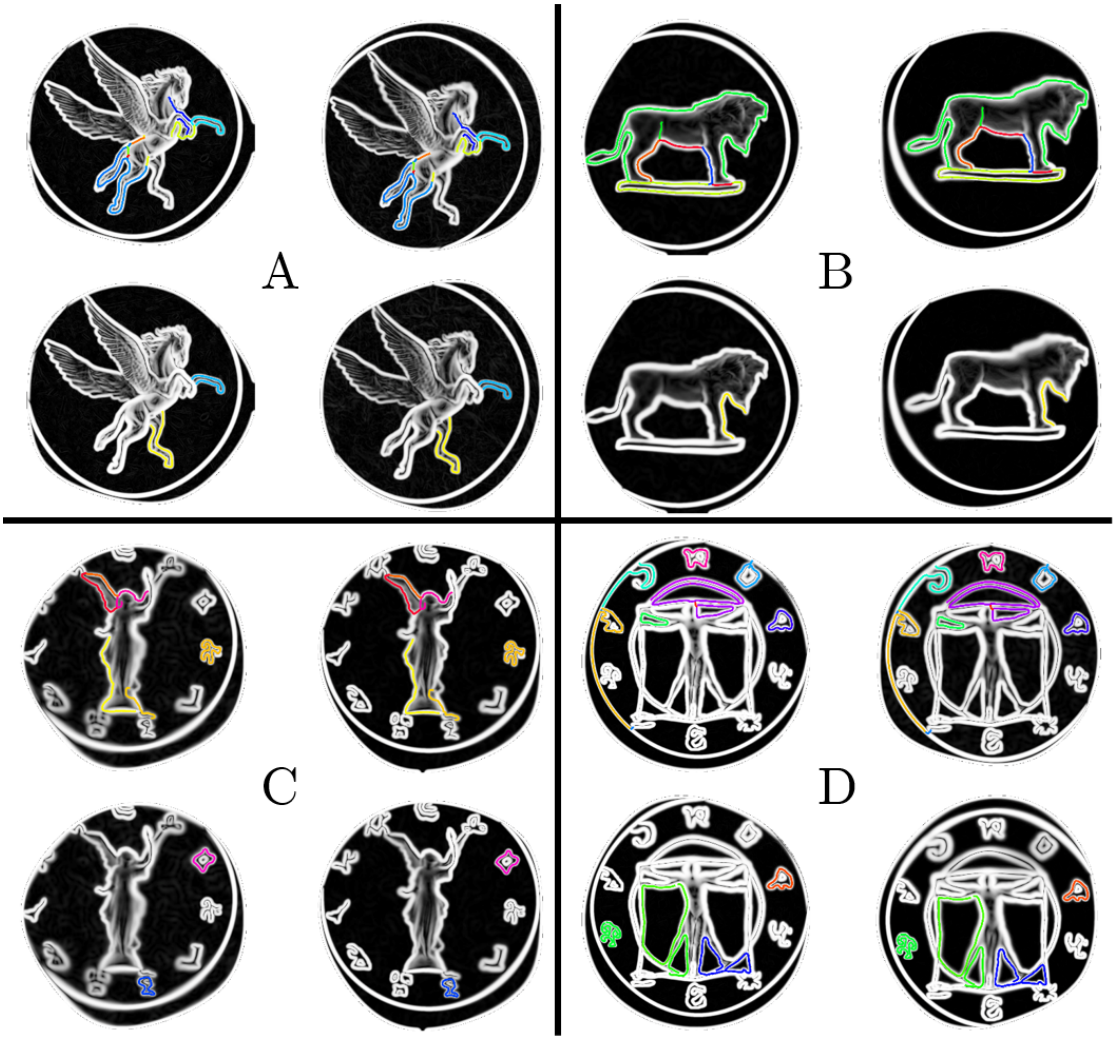


Figure 3.25: Some examples of retrieval using the skeleton extraction with the artificial dataset. For each case, the query coin is set to the left and the retrieved coin to the right. Results were obtained with the skeleton map for each top row and with the Canny edge map for each bottom row.

# coins	Skeleton	Canny
1	54.6	46.9
2	66.3	65.2
3	73.4	70.4
4	75.0	74.2

Table 3.7: Accuracy (%) of the retrieval against the number of coins per die.

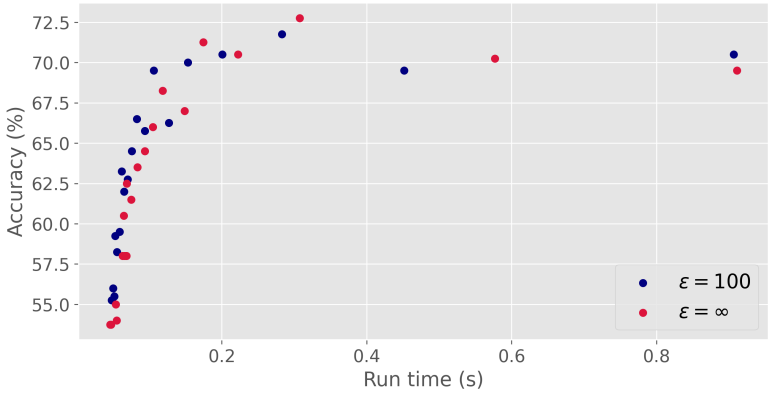
0.003 s/coin, which is what is observed.[9] By contrast, the use of the skeleton map leads to an average of 122 contours per coin, a value around 18 times higher than with Canny, which approximately justifies the observable discrepancy between run times. Similarly, the densities of the segment maps are (see Figure 3.3 and Equation (3.2)), on average, for the whole dataset, $D = 0.022$ using the skeleton maps and $D = 0.0049$ using the Canny edge maps. This difference in density which should account for a large part of the variations in run times.

Since these results are obtained with the leave-one-out strategy, they are only viable when there are 5 coins available per die (except for the query). In Table 3.7, the retrieval was performed with the same amount of coins per die ranging from 1 to 4. For each of these amounts, training and test sets were randomly sampled five times to average the results. It is clear that the accuracy lessens with less available examples. Yet, even though using only one example is not satisfactory for large scale retrieval purposes, it surpasses 50% accuracy with the skeleton extraction technique.

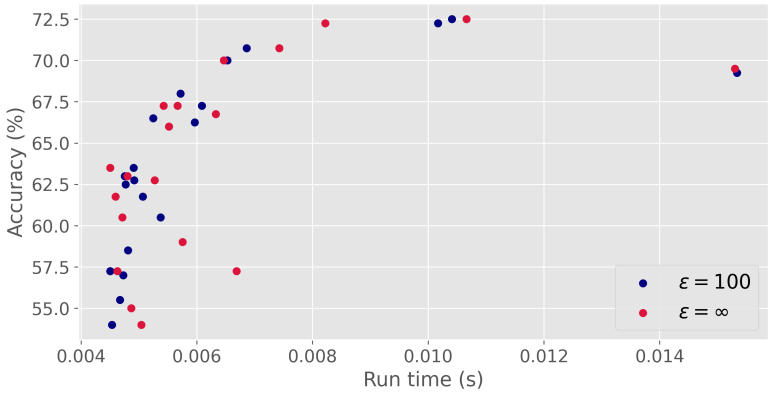
The fact that both extraction methods produce a similar top accuracy for radically different computational time is somewhat puzzling. On the one hand, using the Canny edge map is a lot more efficient because it features less segments, less nodes and therefore less contours to be stored. They produce less collisions, but each contour has to provide a significant amount of information in order to get matched. On the other hand, by construction, the skeleton map produces a lot more connections and is therefore more prone to be home to numerous informative, yet sometimes useless, contours. Besides, all very small segments located inside the segments map were not pruned out and are the root cause of the generation of "twin" contours, differing only by adding this very small segment at one end. This accounts for a lot of collisions in the table.

In the same fashion as what was done in Table 3.3, the evolution of accuracy, run times and collisions with respect to the number of tables (and therefore of higher order signatures) is displayed in Table 3.8. Those results were obtained with the parameters presented in Table 3.6, except that no locality constraints were applied ($\varepsilon = +\infty$). The same observations can be made: with each table added, the accuracy increases but the run times and collisions also increase. The accuracy remains higher for the skeleton map and features a steeper slope ($\approx +4\%$) than for the Canny edge map ($\approx +2\%$).

[9] The quantization parameters in Table 3.4 are much laxer and therefore entail much more collisions, which is not the case here. However, at a very low number of contours stored in the database, it hardly matters since the occurrence of filled bins in the arrays is likely to be very low anyway.



(a) Skeleton contours



(b) Canny contours

Figure 3.26: Retrieval accuracy vs run time per coin.

	Skeleton			Canny		
n_u	Acc	$\bar{t}$	γ	Acc	$\bar{t}$	γ
1	72.5	0.38	7.49	72.2	0.0077	2.07
2	79.0	1.03	10.4	74.2	0.014	2.26
3	83.8	1.82	13.7	76.5	0.020	2.50
4	87.0	9.10	26.2	78.2	0.057	4.00
5	90.8	28.7	46.8	79.8	0.15	5.74

Table 3.8: Retrieval accuracy (Acc, in percentages), average run time ($\bar{t}$, in seconds) and average collision (γ) for a varying number of tables n_u.

Regarding the run time, the difference in behavior between the use of the skeleton map and the Canny edge map is even more striking here. While the accuracy tends to linearly increase with each table, the run time does not and this is especially true with the skeleton map. Thus, we observe a twofold to fourfold rise in run times per each table added, which reaches almost half a minute with 5 tables. Similar comments can be made for the collisions, which values correlate at 98% with those of the run times. This result confirms once more that collisions fully explain the run time variations when the database is sufficiently loaded.

3.4.3.4 Retrieval results on real coins

In Figure 3.27, 8 query coins and their match are presented. The parameters used are presented in Table 3.6. All of them are target coins, which are the highlighted coins in Figure 2.16. The commentaries below and results presented in Figure 3.27 are obtained with the skeleton contours. None of the retrievals attempted using the Canny contours produce correct die prediction and either one (but wrong) or zero contour are matched. Therefore, we do not display the results for Canny.

Overall, the results are not very conclusive. The majority of target coins – including those which are not displayed here – are paired with a wrong match. The errors are often not even interpretable in terms of proximity with other coins (from a same type, for instance). The number of contours found in general is very low, with a mode of only one contour found in the recognition. This translates immediately in very poor scores, which never reach higher values than $\eta = 16$ (coin B). This observation is rather disappointing because the relevancy of such a recognition system is dependent on a statistical convergence towards a good match, or on the fact that very few but highly informative contours are retrieved; these behaviors are mainly not seen in the results. Moreover, the scores themselves are subsequently a combination of low LACS scores, which implies that the contours matched have themselves a low certainty of being true matches. This fact is obviously due to the choice of parameters; $q = (1, 1, 1)$ is a quite strict choice which expects almost identical pairs of signature points. Note that, although this remark seems confusing given that some matches in Figure 3.27 are clearly not the same, they are mere crumbs of signature pairs which are nonetheless the best found. It

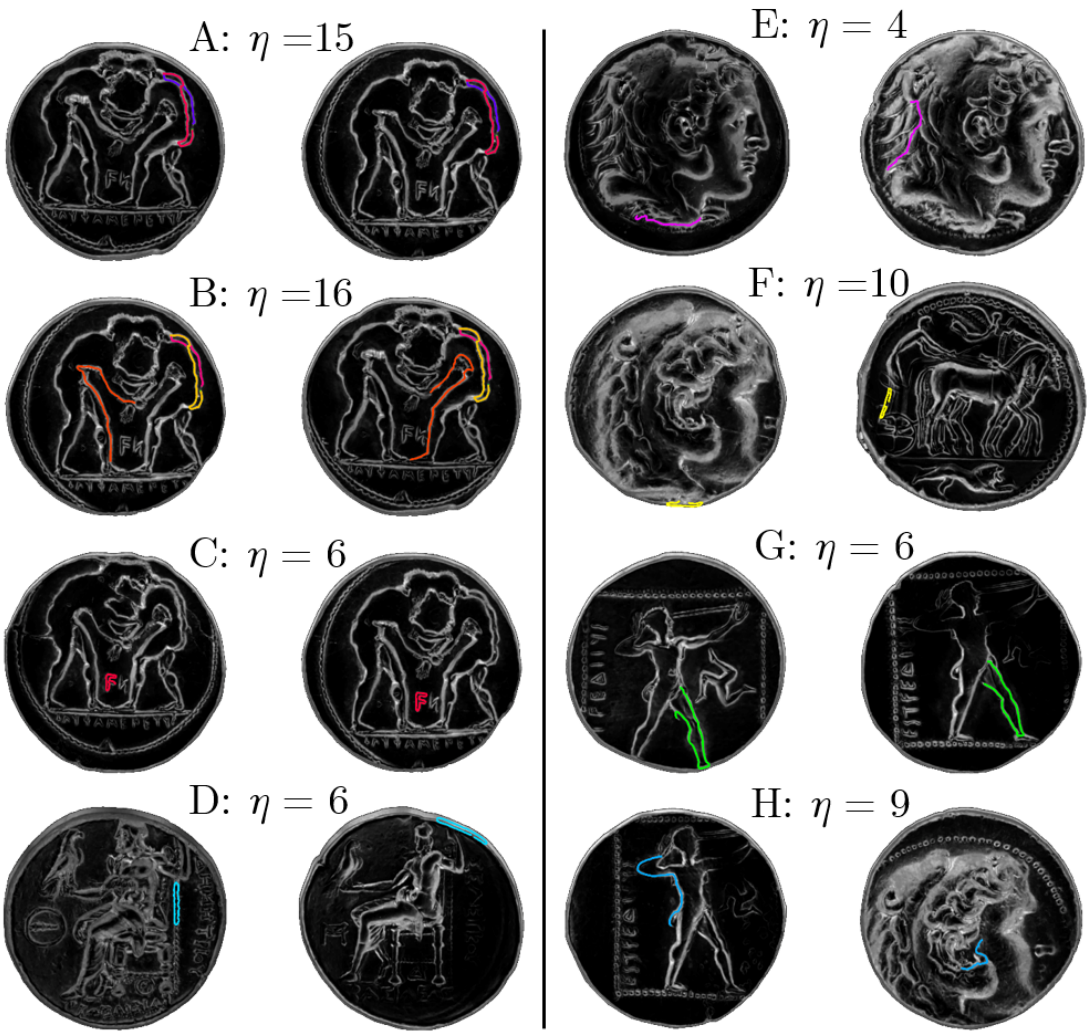


Figure 3.27: Some results of the retrieval for target coins. The extraction has been performed with the skeleton map. Corresponding scores are written above the matching pair. Coins in **A**, **B**, **C** and **E** are correctly paired although the pairing of **E** does not stem from a good contour match.

goes to show that LACS sometimes completely misses what is obvious to us humans. Thus these results do not enable an efficient discrimination between false positives and true positives (low score on coin C and higher score on coin F for instance).

However, there are some interesting tendencies which are reassuring. Even though most of the coins are wrongly paired, the contour(s) which characterize the match do on average resemble each other. This means that the algorithm does not work at random; there is a steady behavior which are due to LACS performances. For example, coins D and F features a single loop, whose match is not aberrant (unfortunately, it does not bring that much information). Looking at coin G, it is surprising to see that the leg is correctly matched even though the die prediction is wrong; what happened to the closest candidate's leg so that it does not match? The only example of correct matches are the coins (not all of them) featuring wrestlers, highlighted in red in Figure 2.16. These coins (A, B and C) have relevant contour matches and are the only one featuring more than a single contour, which propels their scores up to $\eta = 16$. One can nonetheless notice that the orange contour in match B does not corresponds between both coins. This is however remarkable because they are indeed symmetrical, and therefore are seen as the same contour. One could argue that misplaced contours which are significantly resembling but should not match can be relevant in the sense that they convey part of the engraving style, which is an important clue for coin recognition. Changes in parameters (restricting ε, removing locality criterion, changing the LACS parameters) do not significantly improve the results or are exceptions which proves the instability of the algorithm itself.

It is likely that the nature of the coin themselves as well as the occurrences of occlusions have a significant impact on the outcomes. Coins like A or G have thin and well-defined ridges, which makes them likely to generate stable contours. On the contrary, coins E and F (query) do not enjoy the same property; contours defined from thick and ill-defined ridges depart too much from tangible perceived contours.

3.5 Conclusions

The task of recognition using contours fragments represents a challenge. The main rationale that goes through this chapter is that striking a metal flan with a die should create a unique engraving and that the sharp borders and stroke patterns it features should represent signatures of the coin. Thus, the contours, as defined in this chapter, are in theory perfect elements of recognition, originating themselves from the very variations of the die.

There are two main barriers which have been dealt with. The first one is the task of finding, extracting and organizing contour segments so that they embody as best as they can the identity of the die. The difficulty to define contours, which do not exist intrinsically, makes the challenge even harder. Two main strategies have been tried: computing the skeleton of the ridges of the energy map and using the well-known Canny edge detector adapted to the energy map. For this matter, one cannot count on other clues than one's visual appreciation, which concludes to satisfactory edge maps. It appears

that using the skeleton gives the best results as it features more connections between contour segments. The following difficulty is the organization of those elements, which encompasses finding segments, pruning them and concatenating them in order to produce complex structures which are used as the corner stone of the recognition. Solving this problem by connecting consecutive segments via their connection and eliminating small segments is arguably a good strategy to add a significant amount of usable information.

The second barrier is the implementation of the recognition system, which to a set of contours associates candidate coins from (hopefully) the same matrix. In order to describe the contours, we chose to use a signature representation based on the TCD, which is robust to noise and Euclidean transformations. The retrieval is performed with the use of LACS, which is a system of associative arrays and simultaneous pairwise geometric alignment, and yields extremely short computational times on large databases compared to usual works on the subject. The work on the LACS was published in Pattern Recognition in 2021 [63]. Our technique only tackles exact matching and so far cannot deal with occlusions, which comes from the design of the signature. The pipeline for retrieval is largely interchangeable, as one may use other contour representations in our system.

The experiments, conjugating both tasks, lead to mixed results. On the one hand, artificial data enable us to conclude positively about the feasibility of the recognition task. One can expect a fast retrieval, especially with the Canny contours and a satisfactory corresponding accuracy. Those results (and the choice of parameters) do not transfer well to real data. There are some clues that the principle works, but the quality expected from real coins makes it harder to replicate encouraging results. The main culprit seems to be the extraction task. Broken contours due to occlusions can ruin the recognition; through our recognition paradigm, unfortunately, the sum of parts of a contour is not the contour itself. Provided that the contour extraction is improved to be robust for real coins, the final system should be able to yield impressive results, fast and accurate.

In this chapter, the main idea was to derive contour features out of the gradient information available in the energy map. In the next chapter, we will see how it is possible to extract more than mere 1D representations and create a solution for recognition using textures.

Chapter 4

Coin recognition via texture analysis

As pointed out in the previous chapters, using the variations of a quasi-flat object is a relevant way to extract information and highlight descriptive elements which can be manipulated for recognition. Using contours has proven to be a difficult task, even though it remains a promising lead. The alternative lies in the use of general variations within regions of the image, namely textures. This way of describing images have been widely used for object recognition and seem appropriate for us in the context of quasi-flat objects. Describing these objects via their texture is possible with various adaptations of some main techniques in relationship with the energy map model, which will be extensively explained in this part of the dissertation.

4.1 Introduction

This chapter is composed of four parts. The first one investigates some ways to detect points of interest in the object; this step is necessary in order to pinpoint precise areas of the object which can bring significant information. The second one explains how to best describe parts of the object; this is where information is extracted and organized. The third part proposes a framework making use of the structure of the descriptors to match the objects and the last part demonstrates the possibility of clustering a dataset from which no information is known *a priori*. Although the research purpose can remain quite large, the main focus of this study – especially matching and clustering – will be on the pairing of coins struck using the same die. Those coins have at least a very precise definition which make the labeling clearer.

4.1.1 State of the art

We will briefly list some potential candidates of detection and description techniques for the task at hand. Most works on coin recognition and classification have already been tackled in Section 1.3 and use many common techniques which will be elaborated below.

Given the task at hand, the state of the art is focused on three points: the detection, the description and the comparison of interest points.

4.1.1.1 Interest points

Interest point detection is an old and well-developed branch of computer vision which led to many techniques. Our goal here is not to present an exhaustive list of detectors. Rather, we underline the properties of such detection methods with respect to the model we deal with, that is the energy map. We need to detect interest points in order to implement a local description, which enables precise regions to be underlined and eliminates other possible uninformative parts of the object. An interest point is a well-defined point which is robust to various perturbations so that it can be repeatedly detected. There are many possible representations for such a point.

One of the corner stones of interest point detection is the Harris detector [42]. The goal is to refine the detection of image features to restrict the search to corners, excluding edges and flat regions. The Harris detector tries to infer region types based on the eigenvalues of the structure tensor of the image. It is then possible to infer corners via non-maximum suppression over a function of the eigenvalues. This function should have high values when both of the eigenvalues are high, which is where the corners can be found. The algorithm for the Harris detection follows these steps: 1) calculation of the derivatives along both axes of a gray scale image. 2) Build the structure tensor out of the image derivatives I_x and I_y, which is defined as:

$$\mathcal{T} = \sum_{(x,y)\in W} \begin{pmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{pmatrix} \tag{4.1}$$

where W is a local window in the 2D plane. 3) Find the eigenvalues of the structure tensor, for each pixel. 4) Calculate the Harris response as $\det(\mathcal{T}) - 0.05 \cdot \mathrm{Tr}(\mathcal{T})$. 5) Perform non-maxima suppression so that only the responses for corners are retained. This detector features a satisfactory repeatability under rotation and changing illumination and is often used in image database retrieval. It is however computationally expensive and changes in the parametrization can lead to an overwhelming number of detected points.

A different approach is to perform blob detection, which considers interest points as small regions having distinctive features with respect to their surrounding. The most common blob detector is the Laplacian of Gaussian (LoG). First, a Gaussian function is convolved with the image, and then the Laplacian of the result is computed. This can be done for several standard deviations of the Gaussian and therefore the result can be obtained at different scales. In order to provide a multi-scale detector, the strategy is to find the extrema with respect to space (the 2D lattice) and scale (the standard deviation).

A similar detection method is the Difference of Gaussian (DoG), which has been shown to be mathematically close to the Laplacian of Gaussian. The method computes

the difference between two consecutive scales (for a given sampling), and finds the extrema in the same fashion as for the Laplacian of Gaussian technique. This approach has been used for the SIFT detector [66]. Namely, the keypoints are extrema in space and within an image pyramid. Thus, each keypoint is detected at a specific scale, which lead to a more accurate description. The LoG as well as the DoG are invariant to scale, rotation and translation. They tend to produce a lot of feature points, which is statistically interesting, but entails a need for many comparisons during the matching process.

Yet another technique, the Determinant of Hessian (DoH) calculates the Hessian matrix of the image for a given scale and proceeds to find the extrema the same way. This approach has been employed in the SURF technique for detection and description [13]. The determinant of the Hessian matrix is indeed considered as a metric for local change, large values subsequently leading to valid interest point candidates.

Based on this literature, there are various possibility for the construction of our recognition pipeline. We will prefer a method that guarantees a reasonable amount of points. Blob detection methods like LoG or DoG tend to produce a lot of keypoints, which is not what we will favor. The Harris detector is one of the most basic technique, yet it is still used in many domains and yields satisfactory results both in terms of quantity and quality. There are improvements of the Harris detector which, for instance, adapt the method to scale-dependent scenarios. It will be however shown later that scale invariance is not needed in our task.

4.1.1.2 Descriptors

In order to characterize an interest point, it is necessary to compute a description, usually of its surrounding area. Texture descriptors are numerous and have been used in many applications.

Texture description can be performed via the use of wavelets to decompose the frequency information within the texture [34]. A common filter used to this end is the Gabor filter, which uses Gabor wavelet to decompose the signal. The texture is actually convolved with several of such wavelets with varying rotation and scale parameters.

The GIST descriptor [79] explicitely uses the Gabor wavelet to summarize the gradient information in the image by convolving with 32 Gabor filters at 4 scales, 8 orientations which creates 32 feature maps of the same size. Each feature is split into 16 square regions, and then average the feature values within each region. Concatenate the 16 averaged values of all 32 feature maps results in a 512-dimension descriptor.

As stated in Section 1.3, the LIDRIC [111] is a descriptor specially designed for objects like ancient coins. It is indeed built to be resistant to illumination condition, and also uses combinations of Gabor wavelets to achieve that goal. The LIDRIC is computed over a dense grid of keypoints, making the description quite exhaustive and time-consuming. The use of Gabor wavelet at various orientations and scale is said to make the descriptor illumination invariant. Yet, using real coin examples with various light azimuths reveals that this is not the case.

On of the most famous texture descriptor is the SIFT descriptor proposed by Lowe

[66]. The SIFT descriptor is a window of 4×4 cells containing an 8-bin histogram of the local orientation. Since it is inferred from a keypoint detection which provide scale and orientation, the window is oriented and adapted to the scale of the keypoint. The resulting feature descriptor is a 128-dimension vector.

In 2005, Dalal et Triggs introduced HOG features [30]. HOG decomposes the image into cells, within which a histogram of gradient directions is built. To account for illumination variations, the authors normalize locally the histograms by grouping cells into blocks which can overlap. The resulting feature vector, therefore representing a dense cell grid in the image, is not rotation-invariant and computed at a specific scale.

The description via classic descriptors like the SIFT is efficacious mostly when it is paired with its detection step, which allows to target a specific scale. Although SIFT remains a good descriptor, we will prefer HOG to it, since the former covers a large area around the keypoint, which is enough and relevant for our task. We will also comment on the necessity to adapt the algorithms – which for the most part acts on the derivative of the image to process – so that they extract more meaningful data from our objects. HOG is particularly easy to adjust this way.

4.1.1.3 Comparing descriptors

Matching descriptors can be done by several ways. The most usual one is the pairwise comparison, which is the easiest to implement. However, because it requires to compare each potential candidates, the matching is done in $O(n_c)$, where n_c is the size of the database. This can take a long time when considering large databases.

The fastest way to perform the matching can be obtained by using K-d trees [36], which usually consists in splitting the hyperspace for each of the K dimensions of the descriptor. Each component goes through a binary tree, restricting iteratively the search to smaller half-spaces. Supposing the tree is well-balanced, we can expect a complexity of $O(\log n_c)$. However, this form of structure is not really suitable for high dimensions, since the number of branches in the tree thus becomes very high and can make the available data quite sparse within the tree. In other words, the K-d tree is prone to the curse of dimensionality.

4.1.1.4 Global description

Global description takes into account the whole available image or object to provide a description. It is possible to do so, for example, via the use of cross-correlation via the Fourier-Mellin transform (see the technique discussed in the introduction by Marchand [71]). Cross-correlation between images of two coins taken under different illumination conditions does not perform well, but one can imagine correlating the energy maps instead. In the scope of the work of Marchand [71], this works well for registration. However, one issue is that the inability of global comparison to take specific details of the objects into account would produce incorrect matchings between very similar objects; the correlation product drops significantly in the case of small dissimilarities. Another issue is that this method has a significant complexity whereas, in this thesis, we wish

to focus on low-complexity techniques for the recognition. For these reasons, global matching will not be tackled in this work.

4.2 Interest points

The state-of-art proposes methods to deal with images. However, in coins, the available material is much more than what one image can offer. The object itself is at our disposal and so are its features. The goal of the following work is to determine adequate and robust feature points with respect to the nature of the object and the task at hand. In the following, we present five more interest point detectors which take advantage of the energy map.

4.2.1 Locating interest points

First, we propose some detectors which allow for the use of the energy map to discern informative regions. The following lists five detectors and their characteristics.

4.2.1.1 Gaussian maxima

The energy map highlights the steepest parts of the relief of the object. In a sense, it already features interest points defined as the maxima of a function of $\mathcal{E}$. In the following, the energy maps have a size of 1024×1024.

The simplest way to extract interest points is to simply find the local maxima directly in the energy map. However, by considering the width of edges featured in the energy map, it is reasonable to assume that it needs a slight smoothing to fuse close local maxima and retain the most robust ones. The energy map is thus convolved with a Gaussian kernel g of size 8 px and the result is normalized between 0 and 1. The feature points are the local maxima (Equation (4.2)) such that they are separated by at least 32 pixels to each other.

$$X = \underset{(x,y)\in U}{\text{local argmax}} \ (\mathcal{E} * \mathcal{G})(x, y) \tag{4.2}$$

We will refer to those feature points as Gaussian maxima. Gaussian maxima appear to be the centers of small blobs located on ridges in the energy map (Figure 4.1). While this makes sense, their repeatability can be heavily compromised given that they may be positioned differently along the ridge; such a part of the coin features does not have an intrinsic center. Round blobs and corners are reasonable candidates for stable feature points using this methods. Moreover, this detection is very simple, easy to compute and therefore quick to run.

4.2.1.2 Multi-scale Gaussian

In order to build an even more robust version of the Gaussian maxima, one could propose to check their position at various scales. Through several scales, the most robust feature

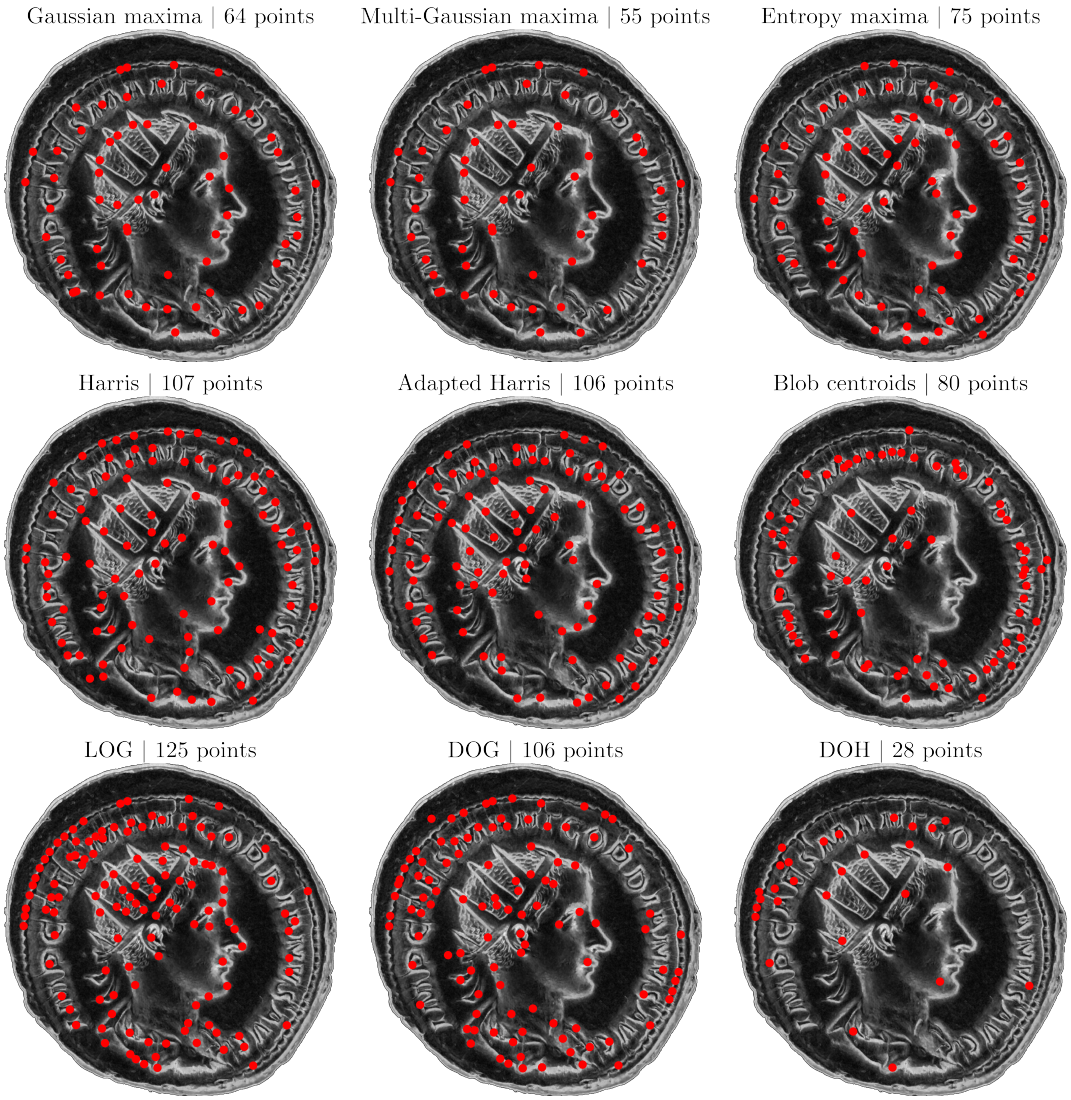


Figure 4.1: Resulting feature points for nine methods of detection. The number of detected points is written next to the method's name. Every point is computed on the same coin, of which we show the energy map.

points will be those who belong to many of them. The energy map is used to create four scales via a pyramid representation. The first scale is the original image and each consecutive scales is obtained via downscaling by 2. For each scales, the gaussian maxima are computed with Equation (4.2). The feature points will be selected as those who are present in at least three scales within a tolerance radius of 32 pixels.

We call these feature points multi-scale Gaussian maxima. There will be significantly less of them than there are simple Gaussian maxima, which can alter the recognition phase given that there are already very few points detected (see for example Figure 4.1, less than 100 points are detected).

4.2.1.3 Local entropy

A third idea is to expect the interest points to be where the energy map is the most informative. A valuable lead is to calculate the local entropy of the energy map. A local entropy filter of size 16 is applied to the energy map, which results in a entropy map $\mathrm{H}(\mathcal{E})$ making highlight the highly informative areas. The entropy H is normalized between 0 and 1 and feature points are detected according to Equation (4.3), ensuring that they are above a threshold of 0.8, which was found empirically.

$$X = \underset{(x,y)\in U}{\text{local argmax}}\ \mathrm{H}(x,y) \quad | \quad \mathrm{H}(x,y) \geq 0.8 \tag{4.3}$$

Entropy maxima can be quite close to Gaussian maxima. They are points located at highly informative parts of the energy map, which is mostly where large variations occur. In a way, they put an emphasis on the energy map derivatives amplitude rather than the energy map itself.

4.2.1.4 Adapting the Harris detector

Contrarily to blob detection or extrema detection, corner detection allows for more precise selection of features in an image. In the case of the energy map, it seems judicious to focus on angular parts of the ridges since they should stay robust to variations of intensity, which can disturb blob-detected points lying on edges. Applying this detector to an image is relevant, because it highlights the intensity variations within the image. Yet, when the algorithm is applied directly to the energy map, the features it reveals belong to the level of the second order derivatives of the object, which limits its relevancy.

Since the computation of the energy map also enables the retrieval of the orientation map, it is judicious to use this information into the Harris corner computation. Namely, as stated previously, applying the Harris algorithm to the energy map treated as an image requires the use of its derivatives. However, it is quite possible to use the object's gradient information formalized as the energy map for the magnitude and the orientation map for the angle (see Section 2.2.5.1).

Using Equations (2.19) an (4.1), we formulate the structure tensor directly from $\mathcal{E}_x$ and $\mathcal{E}_y$, as presented in Equation (4.4).

$$\mathcal{T} = \sum_{(x,y)\in W} \begin{pmatrix} \mathcal{E}_x^2 & \mathcal{E}_x\mathcal{E}_y \\ \mathcal{E}_y\mathcal{E}_x & \mathcal{E}_y^2 \end{pmatrix} \tag{4.4}$$

The rest of the algorithm is exactly the same as for the traditional Harris. Therefore the corners detected will satisfy a more intuitive definition since they will fit closely the corner patterns visually distinguishable within the energy map's ridges. The Harris detector is not scale-invariant. However, since the coins are all resized and framed equally, one can make the assumption that scaling is almost non-existent.

4.2.1.5 Blob ridges

The energy map can be thresholded so as to end up with a blob map. This is the approach previously explained in Chapter 3. Because those filiform regions fit the locations of ridges, they characterize the relief of the object. In order to produce feature points, we extract the centroids of those blobs, as defined by the mean position of all coordinates of the region within the lattice U.

Such a method is simple and easy to implement. We predict that it can behave really well for various transformations, including translation, rotation and scaling. It is however prone to be completely irrelevant if too much occlusion occurs (for instance due to erosion or oxidation) because blobs will easily split in several parts, or their shape will change, leading to different centroids positions.

4.2.2 Formatting interest points

Since each detection technique behaves uniquely and requires specific parameters, they will possibly yield different amounts of points arranged in various distributions inherent of the technique itself. In order to homogenize the amount of points, we empirically determined the parameters of each technique so that they generate a sufficient amount of points.

In order to guarantee that detected points are indeed within a informative region – for instance points detected on the coin field – a selection based on thresholding is performed. The energy map being normalized between 0 and 1, any element of a set of interest points X, for which the energy value is less than 0.2 is removed:

$$(x,y) \in X \implies \mathcal{E}(x,y) \geq 0.2 \tag{4.5}$$

Moreover, the points have to be within the mask defined by the coin. Still, this may not be sufficient because any coin possesses a border, which translates in the energy map as a circular ridge around the coin. There absolutely may be some keypoints along this border and, while they certainly are informative *stricto sensu*, they do not bring relevant material for recognition. Therefore, a stricter condition is necessary. In order to anticipate the next section (on describing the area around keypoints), we impose that

each point $(x,y) \in X$ be positioned so that a square patch of side c and centered around (x,y) has a sufficient fraction $\beta \in [0,1]$ of its area in the mask.[1]

$$(x,y) \in X \implies \sum_{-\frac{c}{2} \leq i,j \leq \frac{c}{2}} \mathcal{M}(x+i, y+j) \geq \beta \cdot c^2 \tag{4.6}$$

Conditions (4.5) and (4.6) are thus necessary conditions for a point to belong to the set of interest points X, and it is sufficient for any detected keypoint to belong to X if it respects those two conditions. Usually, we choose $\beta \geq 0.8$ in our experiments.

The final results of all interest point detection methods are illustrated in Figure 4.1. The points are formatted the way described above, with $\beta = 0.9$. The amount of points detected is heavily dependent on the method and its parameters. Here for example we see that the number of points span from 28 to 125. Obviously, finding too few points impairs the recognition while finding too many may impact the comparison run time. Besides the pertinence of the definition, it is important to consider how well do the feature points adapt to certain standard perturbations. A detector which is not resistant to phenomena that are likely to occur should not be kept. This is what the study in the next section is about.

4.2.3 Repeatability and robustness

Given the variety of detectors, we perform an evaluation of their repeatability for various perturbations. The goal is to trim the possible detectors so that only the most robust ones are retained. The repeatability measures the independence of the detector with respect to some perturbation. The idea is to provide an quantitative idea of the robustness of the detectors. The definition and the use of repeatability is the one given by Schmid [86]. More precisely, it refers to the ϵ-repeatability. We define in Equation (4.7) the set of pairs of points X_1 and $X_2 = T[X_1]$ which are found within an ϵ-neighborhood define by a distance metric $dist$ (L_2 norm), and where T is a perturbation, as:

$$R(\epsilon) = \{(X_1, X_2) \quad | \quad dist(X_1, X_2) < \epsilon\} \tag{4.7}$$

From this and the number of points $|X_1|$ in the first object, we calculate the repeatability according to Gauglitz [38]:

$$\hat{R}(\epsilon) = \frac{|R(\epsilon)|}{|X_1|} \tag{4.8}$$

The choice was made to use this definition to calculate the repeatability instead of the one formulated by Schmid et al. in [86]. The latter employs the minimum of the number of points in both sets. Therefore it provides a more optimistic measurement and treats both objects as equal. While this also works fine in most cases, this likely leads to symmetrical results, for example in the case of scaling, which is explained below. Instead,

[1] It can be argued that a circular area would be a better choice since it is rotation invariant. Even though this remark is perfectly accurate, we actually found no significant difference between the two versions.

Figure 4.2: Coins used for the measurements of repeatability.

we consider the objects as query and target and examine the results in an asymmetrical way.

The perturbations T analyzed are scaling, rotation, additive noise and erosion of the object. The general protocol is always the same: calculate interest points of the original energy map, then detect them for various parameters of perturbation. For every experiments, we choose $\epsilon = 8$. The objects investigated are four ancient coins, shown in Figure 4.2 and from which we compute average values of repeatability. Note that the following results are there for comparison between detectors in this context only and do not reflect an absolute tendency.

Given the results obtained with (4.8) for each parameter values, we also propose a more synthetic indicator. We use the well-known metric area under curve (AUC) to represent the overall repeatability of the range of parameters. More specifically, we calculate a normalized version, defined by Equation (4.9), where p is the set of parameters. This provides a metric to easily compare repeatabilities.[2]

$$\mathrm{AUC} = \frac{1}{|p|} \sum_{p} r_p(\epsilon) \tag{4.9}$$

The details of the experiments are available in Appendix B. In order to get a more condensed view of these results, we propose to average the AUC results into one bar chart. Even though it is possible to weigh the significance of the various perturbations in the calculation, we decide to value them equally. The averaged results are presented in Table 4.1. The compiled versions of repeatability AUC translates quite well the aforementioned evaluations. The first five best ranked detectors are the Gaussian maxima (79.5%), the multi-Gaussian maxima (72.8%), the adapted Harris detector (72.6%), the classic Harris detector (69.1%) and the Laplacian of Gaussian blob detector (68.5%).

[2]It is clear that the definition of repeatability creates a bias in this formula, since the scaling impacts directly and inevitably the number of points detected. However, since the comparison is made under the same assumptions for each detectors, it seems fair to admit that it correctly enables results to be compared.

Method	Gaussian	Multi-Gaussian	Entropy	Harris	Adapted Harris	Centroids	LOG	DOG	DOH
AUC (%)	83.6	75.8	58.8	72.9	75.8	71.2	72.2	71.2	56.6

Table 4.1: Average area under curve of repeatability.

For our application, perturbation like rotation, noise and erosion are prevalent. Erosion is undoubtedly the most frequent deterioration and encompasses any form of occlusion that noise cannot simulate. We therefore reckon that the adapted Harris is the best choice to find feature points.

4.2.4 Number of keypoints

From now on, the focus will be on the list of five best detectors according to the previous section. Those are the Gaussian maxima, multi-Gaussian maxima, Harris, adapted Harris and LoG. Even though it has not been clearly underlined yet, the various detectors do not produce the same amount of keypoints. It is, however, almost certain to be rather small (no more than a few hundreds) compared to those detected in natural images. This quantity is however sufficient for the future matching algorithms presented further in Section 4.4.

For the five remaining methods, Figure 4.3 shows the distribution of the number of detected keypoints on the artificial dataset described in Section 2.4. Four hundred artificial coin were minted (16 types, 5 dies and 5 coins) and detection was applied on each of them. The results show that large differences occur between detectors. On the one hand, the LoG detection, for instance, gathers twice as many points as there are in the Gaussian maxima and four times as many as the Harris detector. Detecting too many points can be a drawback since it enhances comparison run time, which leads us to think that the LoG is not the best choice we have. On the other hand, detecting too few points is risky because it requires that the description be particularly discriminative or that the matching algorithm be especially precise in order to make use of such little batches of interest points.

4.2.5 Informative regions

Even if the study on repeatability enables us to wisely pick the detector for our goal, it is not enough for us to unequivocally chose one once and for all; repeatability is necessary, but not sufficient. For instance, it says nothing about information, that is the amount of relevant data that can be extracted around the interest points and that ensures its distinctiveness. Schmid et al. [86] propose the use of the statistical entropy applied on a distribution of descriptors to infer a measurement of information. This is a relevant lead, but we choose not to make this comparison for several reasons. Firstly, it is mentioned that there needs to be a lot of interest points so that the results are statistically correct. Often the detectors create a few hundreds of them on the energy map, while commonly used detectors yield thousands on natural images. Secondly, we deem the current set of detectors to be sufficiently small and relevant in order to test the description. Finally,

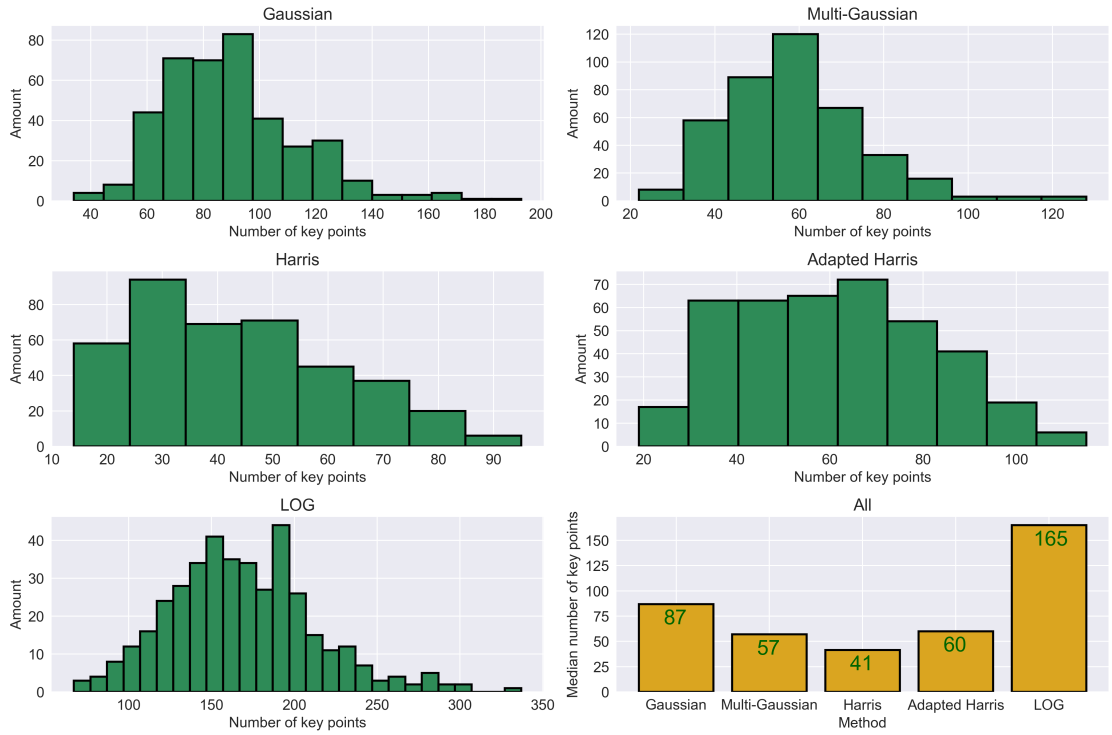


Figure 4.3: Number of keypoints detected per method. The number of bins varies with the range of values so that every bin covers the same range. Five charts show the distribution of the number of keypoints for each method. The sixth chart summarized the distributions via the median value.

the following step of description will by itself provide a measurement of information. Indeed, detectors providing very low information content will cause many points to be matched together, leading to false matches and therefore low accuracy of recognition.

The arrangement of the area around the keypoints is also crucial. We posit that the adapted Harris detector is the best choice to detect relevant interest points. It features a high repeatability and, contrarily to the Gaussian maxima detectors, the areas in points to are designed to be informative, because they are corners. Furthermore, it is interpretable in terms of variations of the object's relief. In the next section we tackle the definition of such areas around these interest points and we will establish the tools to describe them.

4.3 Local description

4.3.1 From interest points to patches

So far, one is able to locate interest points in the energy map, which gives spatial information. To provide a description of the interest points, it is common to use the surrounding neighborhood to collect statistical information about the area where the point is located. In SIFT, for example, a 16×16 region is used for description. The actual size of the region is dependent on the scale found by the detection algorithm, thus the area of description has in reality a variable dimension.

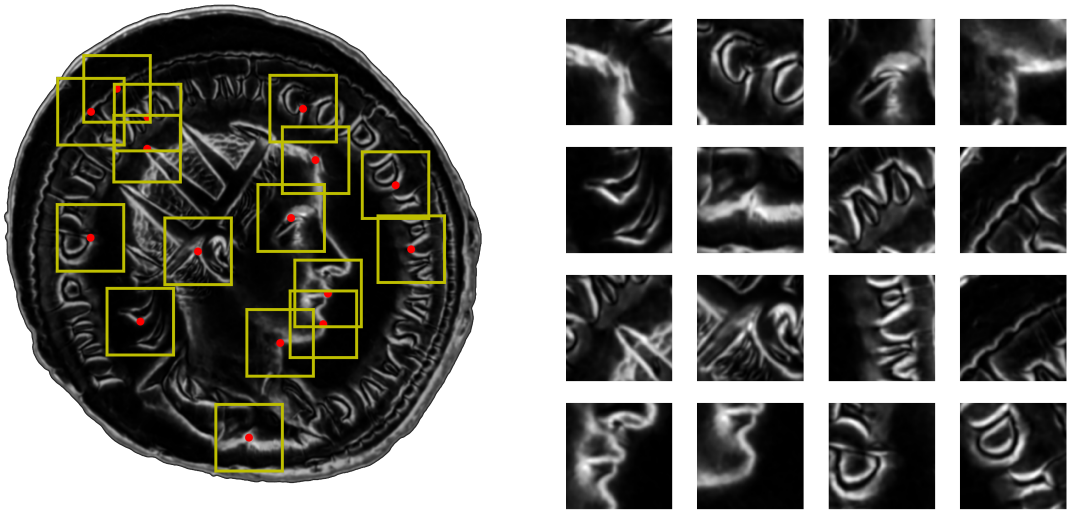
In our case, we do not possess information about scale; only keypoints were found with no additional information about the region. One can argue that there is no need for a scale, since the energy map is always of the same size and the elements constituting the relief of the object have no reason to vary tremendously in size. The choice we make is to select a fixed size of 128×128 to create a patch around the keypoint. The choice of the size is empirical as well as visually determined. Indeed, experiments run on description and matching tend to show that this is a good choice. Besides, the aspect of the energy map – also of fixed size – features large ridges and no fine details, which makes sense given that the die was handcrafted. Therefore, a too small size would miss the information altogether while a too large size may obviously gather too much, which may alter the distinctiveness of the description. This choice is also the reason why some points were excluded near the mask's border in Section 4.2.2.

To be specific, a patch of length l and centered on (x, y) is defined as a region $\mathcal{P}$ of the 2D lattice U such that:

$$\mathcal{P} = \left\{ (x', y') \mid x - \frac{l}{2} \leq x' \leq x + \frac{l}{2} \text{ and } y - \frac{l}{2} \leq y' \leq y + \frac{l}{2} \right\} \tag{4.10}$$

The values extracted from an energy map $\mathcal{E}$ at the location of patch $\mathcal{P}$ is noted $\mathcal{E}_\mathcal{P}$ (Equation (4.11)). Such a set of values will be called an energy patch (respectively an orientation patch with $\mathcal{A}$).

$$\mathcal{E}_\mathcal{P} = \{ \mathcal{E}(x', y') \mid (x', y') \in \mathcal{P} \} \tag{4.11}$$



(a) Patches detected in the energy map.

(b) Details of the energy patches.

Figure 4.4: Example of 16 patches found in an energy map. There are 50 interest points in total, which were detected using the adapted Harris method.

In Figure 4.4 are displayed some examples of patches detected across an energy map using the adapted Harris method (Section 4.2.1.4). The patches are quite sizeable, albeit relevant in comparison with the size of the whole energy map. Again, this is important: it is not desirable to gather too much details in one patch while avoiding to be limited to only flat zones.

4.3.2 Texture description via HOG features

In order to describe the patches, the approach formulated by [30] is employed. The Histogram Of Gradient (HOG) is a concatenation of histograms condensing the information of the local gradient in the image. This formulation is really close to the concept of SIFT descriptors, but does not limit itself to small neighborhoods. Instead, the entire region is taken into account and described. HOG features were used at first to describe and recognize pedestrians.

The choice of HOG to describe our patch comes from various observations. Firstly, it seems relevant to consider the local gradient in the patch to inquire about the local structure and it offers a comfortable amount of information to work with. Secondly, since scale is not determined, using a SIFT-like approach with a small window resolution would blur the informative content. On the contrary, extending this to a HOG has the advantage of preserving the integrity of the local information. Finally, this descriptor respects the spatiality of the patch since histograms are built within cells. This is pertinent given that the patch may present signs of erosion and thus lose part, but not all of its information.

Our energy patches are initially of size 128×128. They are then resized to 64×64. This is essentially for computational purposes since it is smaller while maintaining a sufficient resolution. In order to calculate this descriptor, the gradient components of the energy patch needs to be found. This is important because this means that HOG works on the derivatives of the energy patch, not the energy patch itself. The variations within the energy patch indeed carry convenient data for recognition. These derivatives are used to found the local orientation. Basically, the patch is divided into 8×8 cells, and an 8 bins histogram is computed within each cells. Therefore each histogram is built out of 64 orientation values. In the original idea, a block normalization is performed in order to compensate for the changes in illumination. In our case, this should not be an issue. However, it may still equalize the variations in erosion or smoothing of the coin's surface, since it will behave on the energy map as intensity variations. Block normalization is performed with blocks of 2×2 cells, which overlap by 50%. In each block, each cell is normalized with respect to the entire block. Therefore, the final descriptor has 7×7 blocks of 2×2 cells containing an 8 bins histograms, which amounts to 1568 dimensions. A visual summary is presented in Figure 4.5.

The final descriptor is a 1D vector, a flattened version of the representation presented in Figure 4.5. Namely, it can be written as in Equation (4.12), provided it is understood that they are a total of 49 (blocks) $\times$ 4 cells in the descriptor. Those cells are referred to by exponents and indices represent the orientation bins.

$$d = \left(\underbrace{d_0^0, ..., d_7^0}_{d^0}, \quad \underbrace{d_0^1, ..., d_7^1}_{d^1}, \quad ... \quad \underbrace{d_0^{195}, ..., d_7^{195}}_{d^{195}} \right) \tag{4.12}$$

This description indeed suits the profile of the patches we defined earlier. However, in the same way the mathematical relevancy of the Harris detectors have been questioned in Section 4.2.1.4, the nature of the HOG descriptor is not as relevant as it could be in the case of quasi-flat object. This is discussed in the next section.

4.3.3 Histogram of Oriented Maps of Energy

As mentioned in the previous section, HOG features are a testimony of the variations in the patch, which is calculated via the patch gradient. For the Harris detector, we investigated the possibility of using a lower level of derivative via the energy map (Section 4.2.1.4). In the same fashion, we can use the information provided by both the energy patch and the orientation patch to compute the gradient components $\mathcal{E}_{\mathcal{P}}^x$ and $\mathcal{E}_{\mathcal{P}}^y$ and use them in the HOG algorithm. To clearly separate the two ways of computing these HOG features, we name the latter Histograms of Oriented Map of Energy, or HOME. The formulation of HOME is the same as the one written in Equations (4.12). The only change is the nature of the derivatives.

We postulate that those two "levels of description" are similar but differ in their ability to capture details. Because HOG works with the derivative, it should be able to reveal finer details in the patch whereas working directly with the energy map provides

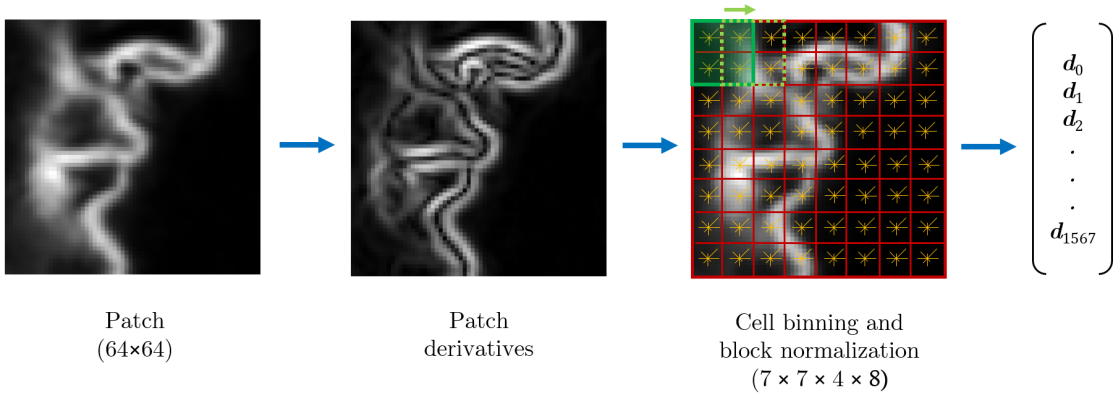


Figure 4.5: Principle of HOG. There are 8×8 cells (red) containing an 8 bin histogram (yellow). Each cell is normalized through block normalization. The blocks (green) contains 2×2 cells and are positioned so as to overlap by 50%. There can be 7×7 blocks within the frame, therefore the final descriptor has a length of $7 \times 7 \times 2 \times 2 \times 8 = 1568$.

a coarser description, underlining only large variations. In Figure 4.6, both levels of description are depicted via their gradient magnitudes (the energy map and the norm of the gradient of the energy map) as well as a visual representation of both HOME and HOG, showing the local orientations of their histograms.

The general observation is that both descriptors are very similar, but a closer look reveals that they seem to highlight different scales of description. HOME coarsely describe the patch, following softly the ridges of the energy map (in Figure 4.6, the chin and mouth). It does not care about small details, which makes it robust to noise. This also means that it will lose relevant information contained in those small details (see the pointy part of the crown in Figure 4.6). HOG is more precise and provides a clearer depiction of the details. It is able to capture the chin and mouth junctions whereas HOME bypasses those parts. This is often at the cost of losing discrimination as some histograms get more homogeneous, pointing at various orientations instead of a clear one.

4.3.4 Rotation invariance

The description of patches through histograms of gradient, should it be the actual HOG or HOME, both assume that the patches are oriented the same way for any coin they represent. However, this type of descriptor is not rotation-invariant, and this may be an issue in our case. Indeed, as mentioned in Section B.2, there is no universal manner to orient all coins the same way.

Even if we do not know the coin orientation, it remains possible to extract some information about it locally. This information is then used to orient the patches, and each patch will therefore have its own orientation. We propose two methods in order to

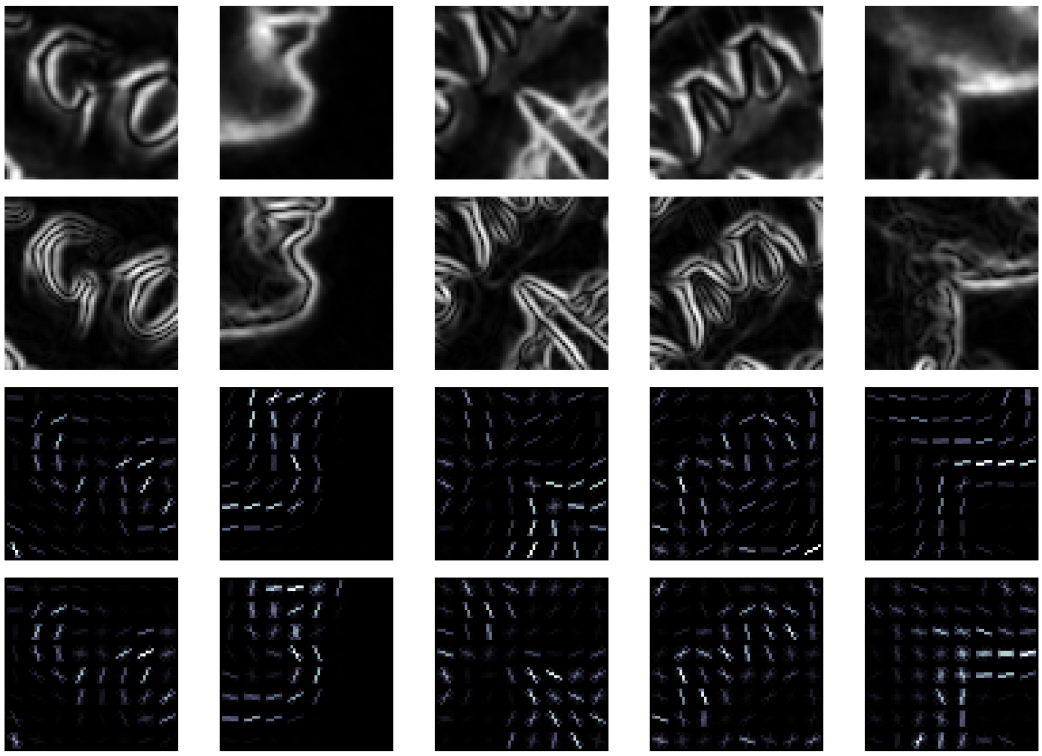


Figure 4.6: Various patches and their description. Top row: energy map patches. Top middle row: gradient magnitude of energy patches. Middle bottom row: HOME descriptors. Bottom: HOG descriptors.

achieve this. The first method relies on the coin center and the second uses only local orientation information inside the patch.

4.3.4.1 Orientation via the center of the coin

The idea is fairly simple: for a given patch of coordinates $(x, y) \in X$, the orientation will be the angle formed by this point and the center of the coin (x_C, y_C), which is given by the simple formula in Equation (4.13).

$$\theta_{\mathcal{P}}(x, y) = \arctan\left(\frac{y - y_C}{x - x_C}\right) \tag{4.13}$$

The question remains as to know what definition to give to the center of the coin. Several methods can be used: average of contour points, average of the entire mask or average of the coin's mask weighted by the energy map. However, given the fragility of each of these definitions (the coin may miss a part, the energy map may be dimmer in some parts due to erosion, ...), we chose to use the center of the image, that is $(512, 512)$ in this case. Recall that the energy map is framed so as to keep a fixed distance from the image borders; this naturally centers the energy map in a convenient way.

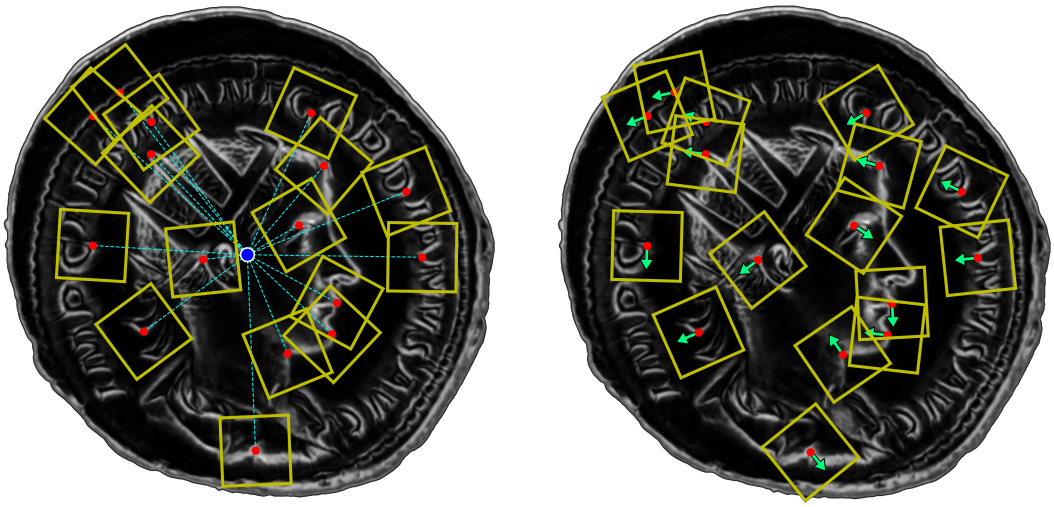
Some results are presented in Figure 4.7a, which takes the same patches as in Figure 4.4a. This naturally ensures a rotation invariance provided the definition of the center is accurate for any coin. Of course, this is not entirely true, but we may assume that the relative displacement of the center from one coin to another is sufficiently small to provoke negligible effects on the patches.

To further investigate this issue, it is useful to recall the content of Section 2.5.2, which exposed an analysis of the impact of decentering the coin with respect to the light positions. The situation here is quite analogous and we can rely on Equation (2.32) to estimate the impact on decentering on the orientation. Replacing in Figure 2.29b the abscissa labels by "patch orientations", it appears that the angle perturbation remains small, and, even if we considered large displacement (for example 50% of the coin radius), the largest differences would amount to 30°. Since HOG and HOME descriptors sample the orientations with 8 bins, it corresponds to a 45° admissible angle error. Therefore, the orientation error should be largely acceptable. However, the patch itself moves and so do the cells. For a cell at the border of the patch of side $c = 128$ px, the displacement δ for an angle error of $\delta\phi = 10°$ [3] is given by

$$\begin{aligned}\delta &= c \cdot \sqrt{2} \cdot \sin\frac{\delta\phi}{2} \\ &= 128 \times \sqrt{2} \times \sin 5° \\ &\approx 16 \text{ px}\end{aligned}$$

A cell is 8×8 px after resizing from 128×128 to 64×64. So a cell side is 16 px long in the original patch, which means cells at the border will move by one cell of distance

[3] This corresponds to a maximum error for a decentering of around 30% the radius of the coin.



(a) Coin centered. (b) Local pooling.

Figure 4.7: Oriented patches on the coin with two methods. In (a), distances (cyan) connecting the center of the coin (blue) to the centers of the patches were added to better show the orientations. In (b), the green arrows show the local orientations.

because of the error. It seems that, in this scenario, the correct alignments of patches is uncertain, depending on the coin center position.

The patch must be extracted from $\mathcal{E}$ as well as from $\mathcal{A}$ if HOME is used as a descriptor. The last step is therefore to re-index the values of the patches $\mathcal{A}_\mathcal{P}$ extracted from $\mathcal{A}$. Indeed, the angles corresponding to the newly rotated patches are based on the orientations from the unrotated patch. The coin pictures are acquired in a certain order, and the zero angle corresponds to the orientation of the first image. Since the rotation of the patches aims at calibrating the patches of all coins on the same scale, their angles need to be calibrated as well using Equation (4.14).

$$\forall (x',y') \in \mathcal{P}, \quad \mathcal{A}_\mathcal{P}(x',y') \leftarrow \mathcal{A}_\mathcal{P}(x',y') - \theta_\mathcal{P} \tag{4.14}$$

4.3.4.2 Orientation via local pooling

Another method is to consider each patch content regardless of the other features of the coin. The core idea in this method is to extract a small patch $\delta\mathcal{P}$, centered on each keypoints $(x,y) \in X$, of the orientation map (Section 2.2.4), and to compute the average orientation (Equation (4.15)). In reality, we should extract a disk, since there is *a priori* no reason to favor some directions. Therefore the pooled values are limited to a disk of center the keypoint and of radius 20.

Within the region $\delta\mathcal{P}$, the average orientation is calculated according to Equation (4.15). Note that this equation is built like Equation (2.18), since it is a circular mean.

The orientation is then assigned to the patch and the final patches $\mathcal{E}_\mathcal{P}$ and $\mathcal{A}_\mathcal{P}$ are rotated accordingly.

$$\forall (x,y) \in X, \quad \theta_\mathcal{P} = \arctan\left(\frac{\sum_{(x',y')\in\delta\mathcal{P}(x,y)} \mathcal{E}(x',y') \cdot \sin \mathcal{A}(x',y')}{\sum_{(x',y')\in\delta\mathcal{P}(x,y)} \mathcal{E}(x',y') \cdot \cos \mathcal{A}(x',y)}\right) \tag{4.15}$$

Additionally, the requirement of calibration is needed for $\mathcal{A}_\mathcal{P}$ (Equation (4.14)). A summary of the steps is provided in Algorithm 7.

To visually evaluate the quality of this method, Figure 4.8 displays the orientations found for two coins made from the same die. Two coins with very close topographies should feature the same set of patches. In this example, 26 arrows are displayed in light green, showing close orientations.

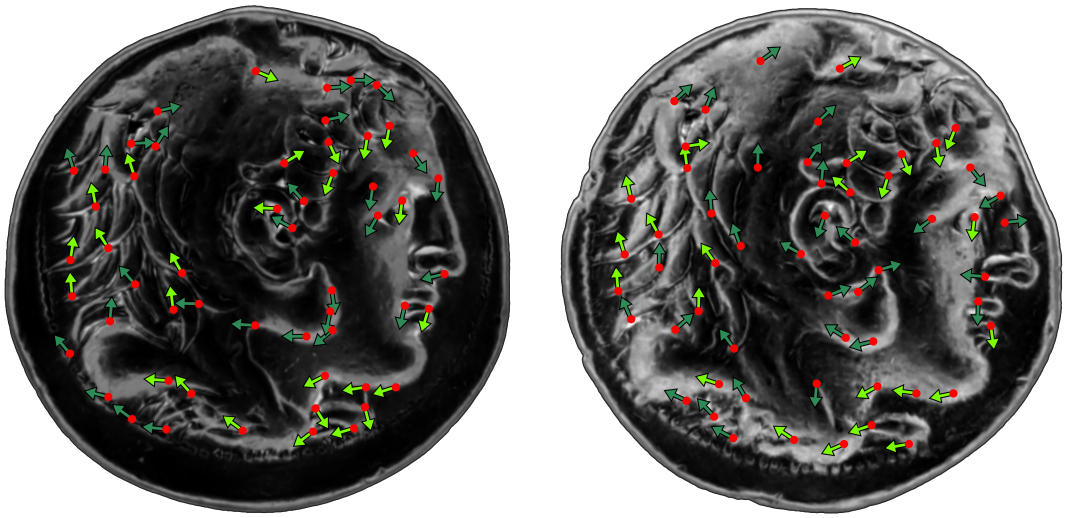
Algorithm 7: Patches extraction with local orientation pooling

Input : X: keypoints
$\mathcal{E}$: energy map
$\mathcal{A}$: orientation map

Output : lists of energy patches and orientation patches

$L_\mathcal{E} \leftarrow$ empty list
$L_\mathcal{A} \leftarrow$ empty list
for $(x,y) \in X$ **do**
  Calculate the patches $\delta\mathcal{P}(x,y)$ and $P(x,y)$
  Extract the small patches $\mathcal{E}_{\delta\mathcal{P}}$ and $\mathcal{A}_{\delta\mathcal{P}}$
  Calculate $\overline{\theta}$ with Equation (4.15)
  Rotate $\mathcal{P}$ according to $\overline{\theta}$ and the center of the image
  Extract an energy patch $\mathcal{E}_\mathcal{P}$ and orientation patch $\mathcal{A}_\mathcal{P}$
  Calibrate $\mathcal{A}_\mathcal{P}$ with $\theta_\mathcal{P}$ and Equation (4.14)
  $L_\mathcal{E} \xleftarrow[\text{append}]{} \mathcal{E}_\mathcal{P}$
  $L_\mathcal{A} \xleftarrow[\text{append}]{} \mathcal{A}_\mathcal{P}$
return $L_\mathcal{E}$, $L_\mathcal{A}$

In summary, the description of the coins are based on three major steps. Firstly, the large variations of the energy map make it preferable to use a vast region, or patch, around each keypoint. It is essential that we can find informative content inside the patches, which we decide to limit to a size of 128×128 px^2, or around 1.6% of the image surface. Secondly, the description is made via the histogram of gradient principle (HOG), which we also adapt to the quasi-flat object properties (HOME). Finally, since coins do not have a fixed orientation, two techniques to make the description invariant to rotation are proposed. The next step is to propose an efficient way to compare and match similar patches in order to recognize and retrieve coins from the same die.



(a) (b)

Figure 4.8: Orientation of patches for two coins from the same die. Light green arrows are the ones that actually match from one coin to the other.

4.4 Coin matching

So far, the task of recognizing coins takes the same approach as most feature matching algorithms, which encompasses capturing some keypoints in the image and describing them in a way that makes it possible to compare them. The next step is usually a matching algorithm which tries to compare a query element with some already known examples in a database. The standard pairwise comparison has a linear complexity, which is unacceptable when the amount of elements in the database is large. Learning is a paradigm that we did not chose to take. One reason is that it often requires large quantities of training examples per class, which is a condition rarely fulfilled in the case of recognition of coins struck from the same die.

4.4.1 General strategy

The strategy taken unfolds in the following manner. Each coins having been described is now a collection of keypoints and descriptors associated with a label gathering the type index and the die index.[4] Alternatively, one could only use a die index without the use of a type to discriminate between clusters of coins. All the coins are stored into a database whose structure is based on LACS model (see Section 3.3.4.2). The retrieval is performed in a similar manner as well. As long as collisions are well-handled, the

[4]There is not really such thing as a "type", but it does not matter here. We just say that it is depends on a clustering performed by experts, which allows certain coins to be grouped together. It can also simply refer to coins that resemble each other.

complexity of such a retrieval is therefore close to $O(1)$. Because our goal this time is the matching of patches instead of contours, we call the data structure and its associated retrieval system LAPS for Low-complexity Arrays of Patch Signatures.

In the following, we begin by describing the general structure of the LAPS system. The outline of the whole process of storage and retrieval is synthesized in Figure 4.9 and it will be referred to at various points in the section.

4.4.2 Structure

In chapter 2, LACS were introduced as a system of associative arrays. We exploited the signature of the contour, which really was just a fragment of a descriptor. By enabling the use of several signatures, that is several descriptor fragments, we made a first step towards a more general form of LACS. We intend to use HOG/HOME descriptors with the LACS paradigm, using associative arrays. In this case, the shapes are equivalent to the patches, the sets of signatures are equivalent to the descriptors and the pairs inside the signatures are equivalent to the bins of the orientation histograms. One major difference here is that the algorithm works at a deeper level than the coin itself since it focuses on patches, although a similar strategy has been exploited in Chapter 3, Section 3.3.7. For the shapes there was one set of signatures per contour whereas there are several patches/descriptors per coin. The other difference is that the geometric alignment is performed at the level of the whole coin instead of the patch itself.

In the case of contour signatures, we described a parallel system of tables which enables the processing of various frequency levels of the shape simultaneously; it was therefore explained that the table itself performs a logical *and* operation and that the final retrieval score corresponds to the sum of all the contributions of the value retrieved in each table in parallel. The same phenomenon occurs for LAPS. In order to build the structure, a first option would to consider using the whole descriptor for one table. However this implies that the table is 1568 nested arrays long, which corresponds to only one set of *and* operator. This is too discriminative and is more than likely to return nothing. A second option could be to consider each individual component of the descriptor and thus build 1568 tables, with only one array per table. This is a more viable option, but it is risky since it may lead to lots of different results with few or no coherence whatsoever. Instead, the general structure is built around the geometrical arrangement of the descriptors components. The system is built with respect to the patch so that there is a relationship between each sub-descriptor (histogram of HOG/HOME) and its position on the patch. Therefore, the system is conceived so that one table corresponds to one histogram. Because there are 1568 components in total, the complete system is composed of $n_u = 196$ tables of $n_v = 8$ nested arrays each. This arrangement is represented in Figure 4.9, where the system of associative arrays are composed of 196 tables of 8 nested arrays.[5] Furthermore, since each table corresponds to a histogram

[5]The nested arrays are presented sequentially in Figure 4.9 for simplicity. However, one as to keep in mind that there are multiple arrays stored at various location in the previous array. The reader is invited to examine Figure 3.14 in Chapter 3 for a graphical example.

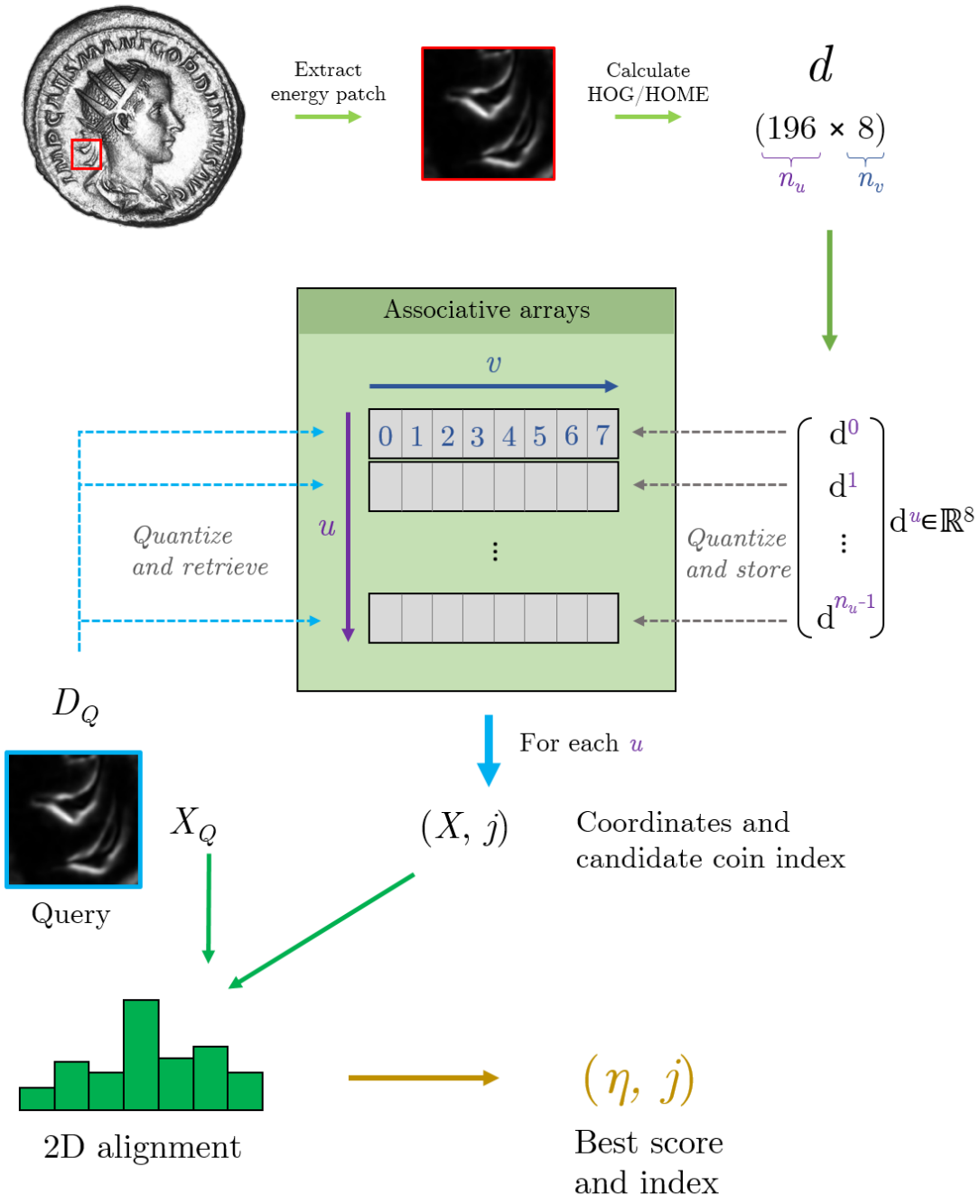


Figure 4.9: General structure of the LAPS system. The storage is presented in the first half of the schema. Every patch is converted into its descriptor, each component of which is quantized and stored in each of the n_u=196 tables, containing n_v=8 nested arrays. The information stored in the last array is the coordinate of the patch X and the coin index j. The retrieval is shown in the second half of the figure. In the same manner, the query descriptor is quantized and matched against the 196 tables. For each table, a tuple (X, j) may be retrieved by reaching the last nested array. This operation is repeated for each patch of the query coin and a geometric alignment is performed with a 2D histogram (one per index j). The score is given by the largest bin of all the candidate histograms.

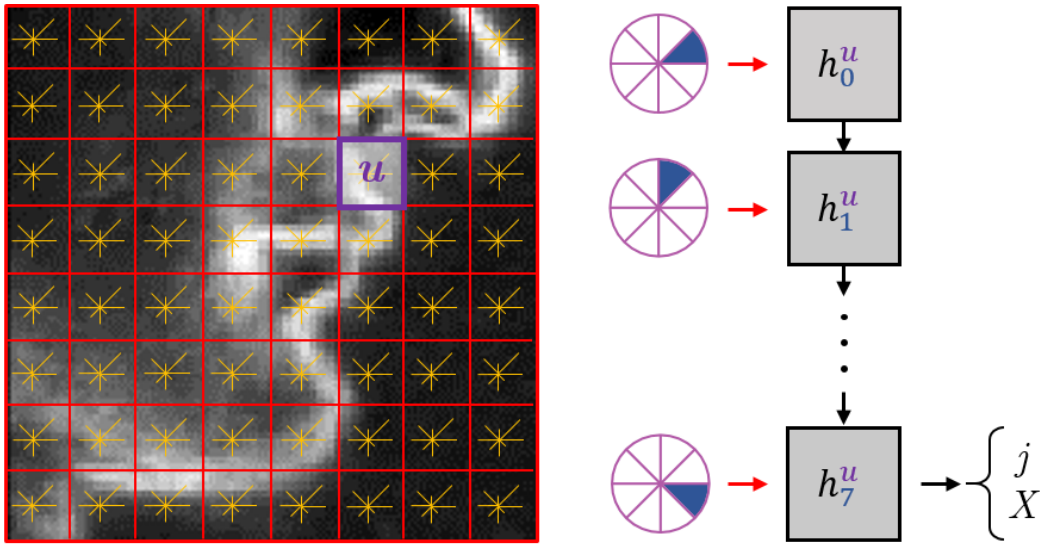


Figure 4.10: Details of one nested array for a sub-descriptor of index u. Each sub-descriptor component, represented as an eighth of a disk, leads to a key via h_v^u to determine the location of the next array. During the storing phase, every component is used to determine the final result's position. During the retrieval phase, if one calculated index points to nothing, the retrieval stops and *None* is returned.

of orientations in HOG or HOME, the structure takes into account the locality of each patch sub-descriptor.[6] The locality can be appreciated in Figure 4.10.

In the same way we defined LACS, we denote by h_v^u the key function of the u^{th} sub-descriptor for its v^{th} component (see Equation (4.12)). The definition for these functions is similar to the one formulated for LACS (Equation (4.16)).

$$\forall u \in \{0,...,n_u-1\},\ \forall v \in \{0,...,n_v-1\},\ h_v^u(d_v^u) = \lfloor q \cdot d_v^u \rfloor \tag{4.16}$$

It takes into account a quantization parameter q, which controls the precision at which we want to evaluate the descriptors. A low q means more collisions, which enables more candidates to be retrieved; a high q increases the precision and the arrays will filter out many candidates or even all the candidates. Contrarily to LACS though, all parameters are the same since there is no quantitative distinction to be made between each component of the sub-descriptors.

Now that the system is built, we will see in the next section how the values can be stored in it and how the retrieval is done.

[6] Note that this is not exactly how the descriptor geometry really is, since there are 7×7 blocks of 2×2 cells. The principle remains the same and the system preserves the locality of each cell, even though they are represented multiple times because of the block normalization.

4.4.3 Storage and retrieval

When a patch is stored, each of its sub-descriptors is stored in its respective table, and each of the 8 components of the sub-descriptor goes through each nested array. For example, if we denote the sub-descriptor $d^u = \{d_0^u, ..., d_{n_v-1}^u\}$, $u \in \{0, ..., n_u - 1\}$, we begin by calculating $h_0^u(d_0^u)$ which serves as a key for the first array. In this array, the location of $h_0^u(d_0^u)$ is the next sub-array for which we calculate $h_1^u(d_1^u)$, and so on. The complete algorithm is presented in Algorithm 8 and Figures 4.9 and 4.10 provide an illustration of how to go from a patch and its descriptor to the storage of the components. The final array ($v = 7$ in Figure 4.9) contains the relevant information about this patch, which is the coin index j and the coordinates X of the patch within the coin. Note that there is a thresholding condition in Algorithm 8, line 4. It has been found experimentally that recognition performs better when one excludes sub-descriptors having a low Euclidian norm (denoted by $\|.\|$ in Algorithm 8). The logical interpretation is that there are many parts of the energy patches that are of low values. These parts will then overload the tables because many stored values will reach the zeroth bin in each array.

Algorithm 8: *store*: storing one coin and its keypoints in a system of tables using its descriptors

Input : $\mathcal{H}$: system of tables
$\{d_0, ..., d_{n_X-1}\}$: list of descriptors
$\{X_0, ..., X_{n_X-1}\}$: list of keypoints
j: coin index

```
1 for i from 0 to n_X − 1 do
2     d = d_i        // i^th descriptor, d ∈ ℝ^{n_u} × ℝ^{n_v}
3     for u from 0 to n_u − 1 do
4         if ‖d^u‖ > 0.2 then
5             add(H^u, d^u, X_i, j)      // add X_i and j in the u^th table using
                  the u^th sub-descriptor
```

When a query patch is submitted, each of its sub-descriptors is tested against a specific table (indexed by u) of nested arrays. That way, passing through the table means matching the aforementioned histogram, bin per bin, in the same order. If, at any moment, the location determined by the key has no further array, the retrieval stops. Also, in the same manner, if nothing is retrieved then the method returns *None*. Of course, since all the tables are independent, some may retrieve a value and others may not, which enables us to overcome issues such as local data missing due to erosion, noise or occlusion in general.

Since the algorithms for adding and getting were already made general for LACS (Algorithms 4 and 5), they are used as they are for 196 different parallel tables, using

only one quantization parameter q. The whole retrieval routine is presented in Algorithm 9. Below are some commentaries about the retrieval algorithm.

- The order of the sub-descriptor components going through the nested arrays is always the same but arbitrary. However, it mathematically does not change anything since the retrieval returns an informative result if and only if there is an array/result located at each component key (logical *and*).

- One aspect of the Algorithm 9 is that it specifically calls a sorting method in order to rank the final retrieved items (line 11). In a way, this is a poor choice since a main asset of such a system is its rapidity (one can expect, at best and depending on the collisions, a constant complexity). Adding such a method would therefore corrupt the purpose since this part of the code runs, at best, in a $O(n_X \log n_X)$ complexity. There are various phenomena at play since it depends on a lot of factors such as the hardware, programming language, etc. However, the computation here is always limited by m, the maximum number of retrieved items (Algorithm 9, line 12). If this values is set to infinity, then, of course, the algorithm ends up exceeding $O(1)$. So, in this case, m acts as a speed factor which implicitly assumes that most of the values retrieved are relevant for the next parts of the algorithm (see Section 4.4.5 and Algorithm 11). The alternative is to successively update the maxima (this approach will be used in Algorithm 10 in the next sections). Yet, this implementation is not optimized (contrarily to the sorting method that we used) and takes more time to compute if m is too large (100 is already too much). The conclusion on this issue is that, on paper, the successive updating is faster but in practice, with the tools we used, it is slower.

- The value m is critical for speed, as pointed out above, but is detrimental to the quality of the retrieval. Indeed, there is a selection of items within the tables which is restricted to a certain number. Therefore, we may miss relevant coin indices and patch locations which would have been useful for the final decision. The smaller m, the more potential candidates are likely missing. Moreover, the way we stored data in the tables, and especially in the final array, is deterministic: items are successively appended and the first m items end up cut out. One way to circumvent this problem is to shuffle the entire table once the storage phase is done. This means that for every final array of each table, the list of stored item is randomly reorganized, which gives no deterministic privilege to any candidate. This is not optimal but one can rely on the fact that, if a coin is a decent candidate for matching, it will statistically prevail over the others.

The way the structure, the storage and the retrieval behave is certainly different from the how a classic Euclidean brute force matching would. The next section provides an mathematical interpretation to the retrieval and describes how to perceive the final score.

Algorithm 9: *retrieve*: find the matches to a descriptor in all tables

Input : $\mathcal{H}$: system of associative arrays
$d \in \mathbb{R}^{n_u} \times \mathbb{R}^{n_v}$: descriptor
m: maximum number of retrieved values
q: quantization parameters

Output : J: best indices
X: best corresponding keypoints

$\eta \leftarrow$ associative array (key, value) initialized at 0
$X \leftarrow$ dictionary of tuples
for $u = 0$ *to* $n_u - 1$ **do**
    $L \leftarrow get(\mathcal{H}^u, q, d^u)$ (Algorithm 5) `// request with` d^u `in table` u
    if L *is None* **then**
        return
    for (x, y), j *in* L **do**
        $\eta[j] \leftarrow \eta[j] + 1$
        $X[j] = (x, y)$
    J $\leftarrow \eta.keys()$ `// the keys of the dictionary`
    $sort(\eta, \mathrm{J}, X)$ `// rank according to scores`
    $truncate((\eta, \mathrm{J}, X),\ m)$ `// keep the` m `best indices`
return J, X

4.4.4 Matching distance equivalent

When a query patch with descriptor d is processed through the system, each of its sub-descriptors d^u will be used to seek the values in the deepest array of each table via each of its components. Thus, if another patch represented by its descriptor $\hat{d}$ is found in one table, it means that one part of the patch is matched with the other. In Figure 4.11, both descriptors are graphically represented and their matched parts are in blue.

Following this logic, for a given candidate stored in the system, each of its sub-descriptors can only possibly be either equal or not equal. In Algorithm 9, the score for a given candidate is incremented each time a sub-descriptor is matched. In essence, if we consider the patch as a grid of sub-descriptors, the algorithm calculates the similarity between a query patch and a candidate. In other words, a Hamming distance is calculated, or rather a "Hamming similarity". In Equation (4.17), each contribution to the score η is calculated via the sum of matched sub-descriptors, where $\delta(a = b)$ is 1 if $a = b$, else 0.

$$\eta = \sum_{u=0}^{n_u} \delta(\hat{d}^u = d^u) \tag{4.17}$$

$$= n_u - \text{Hamming}(\hat{\mathcal{D}}, \mathcal{D}) \tag{4.18}$$

At this point, the system behaves as if it was comparing each patch of the query with potential candidates via a Hamming comparison. However, those comparisons were hopefully done sparsely since not every patch, or at least not every sub-descriptor, is retrieved. Consequently, the whole calculation is performed quickly and in theory depends only on the number of query patches.

So far the retrieval is done independently of the position of the patches. We will examine in the next section what can be done to improve the retrieval using geometric attributes.

4.4.5 Geometric alignment

During the retrieval, we would like to assess the plausibility of each pair of patches of two coins (query and retrieved) using a geometric plausibility measurement. Namely, we want to know if the geometric transformation from one set of points to the other is the same for a significant amount of points; if there is no consensus, then the probability of a match between the two coins should be low. As stated in Section 1.3, in the context of coin matching, the work of Zambanini [113] used a pairwise geometric consistency cost to assess this plausibility. However, any sort of pairwise computations like this is extremely expensive because it requires testing all matches.

Furthermore, we want to be able to perform this assessment during the retrieval. Whenever a query patch is put through the system, the values retrieved per table, if they exist, contain the coordinates of patches in possible candidate coins. The goal, then, is the same as in LACS algorithm: we want to align the query with the candidates

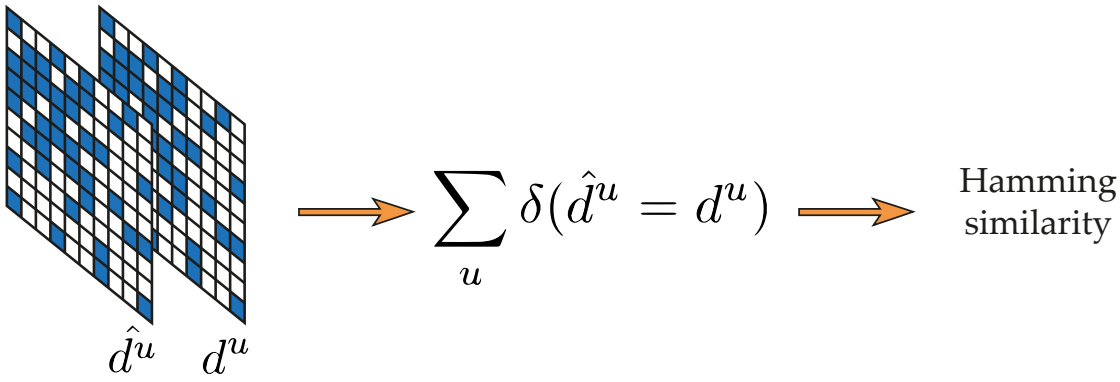


Figure 4.11: Two patches are represented by a grid of sub-descriptors. The retrieval system acts as if, for each candidate, a Hamming similarity was computed between it and the query since each matched sub-descriptor (blue) adds to the final score.

to evaluate its plausibility. The geometric alignment calculates and finds the candidate coin for which there is the maximum amount of patches having the same shift with the query.

In LACS, the aligned features were the coordinates of points along the curvilinear abscissa of the shape and the alignment was therefore unidimensional. In the case of coin matching, two dimensions have to be used, since the keypoints X belong to the 2D lattice U. This poses no problem if we assume that the coin has not undergone a rotation. Allowing the coins to rotate makes us change the geometry to a polar one. In this section, the index $\mathcal{Q}$ denotes one query patch in the query coin.

4.4.5.1 Cartesian alignment

The first step to perform geometric alignment is to consider the case of upright coins. In this case, the alignment consists simply in building an accumulator which tracks the shift between each query patch and each candidate patch. To do so, during the routine examining each query patch, each retrieved coordinates are compared to the query coordinates according to Equation (4.19), where x is the abscissa of the retrieved patch. This definition is identical along the y axis.

$$\Delta x = \left\lfloor \frac{x - x_{\mathcal{Q}}}{\varepsilon} \right\rfloor \tag{4.19}$$

The quantity ε is a quantization divisor which simply represents the step, in pixels, between each bin in the histogram (see Figure 4.12). In essence, ε is merely a tolerance in the 2D lattice. For example, if one chooses $\varepsilon = 10$ px, they allow for an error of 10 px of shifts.[7]

[7] This is not entirely true since this is a quantization, and therefore a shift of 9 and a shift of 11 would be quantized differently (0 and 1) even though they are 2 pixels apart.

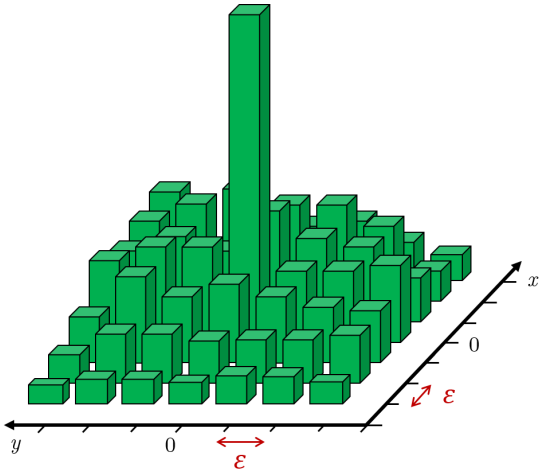


Figure 4.12: 3D visualization of the alignment histogram for one candidate coin. The x and y axes are quantized the same way via the quantity ε. The maximum value in the histogram is kept and eventually compared to the values obtained for other candidates.

The histogram is built for a coin in particular, so that each candidate coin has its own histogram. For instance, one candidate may have the histogram presented in Figure 4.12. The peak here indicates that a lot of patches in the query match the positions of the patches in the candidate coin for a shift between 0 and ε. This alone indicates a good probability of the coin being an accurate match with the query, since many parts of it directly match.

4.4.5.2 Polar alignment

In the case of rotation, the simple use of a 2D shift histogram cannot be recommended. Any rotation is assumed to be performed with respect to the coin center, whatever its definition may be. The usual way to perform a rotation is to capture the coin in a random orientation, which any non-expert individual may do. Since the coin is placed and centered within a 1024×1024 frame, we may once again assume that the center is the center of the frame ($x_C = 512,\ y_C = 512$).

The idea is thus to use the polar coordinates with respect to this center, such that for any point $(x, y) \in X$:

$$\begin{cases} \rho = \sqrt{(x - x_C)^2 + (y - y_C)^2} \\ \theta = \arctan\left(\frac{y - y_C}{x - x_C}\right) \end{cases} \tag{4.20}$$

With Equations (4.20) in mind, we may now build a histogram similar to the one shown in Figure 4.12 and use a similar expression for the quantization (Equation (4.21)),

where $\delta\theta$ is defined by the minimum angle difference in Equation (4.22).

$$\begin{cases} \Delta\rho = \left\lfloor \frac{\rho - \rho_{\mathcal{Q}}}{\varepsilon_\rho} \right\rfloor \\ \Delta\theta = \left\lfloor \frac{\delta\theta}{\varepsilon_\theta} \right\rfloor \end{cases} \tag{4.21}$$

$$\delta\theta = (\theta - \theta_{\mathcal{Q}} + \pi \pmod{2\pi}) - \pi \tag{4.22}$$

This time though, Equation (4.21) features two quantization parameters ε_ρ and ε_θ because it is important to differentiate between the two quantities; one being a distance (which can be negative) and the other an angle.

The critical point to make about the properties of both geometric alignments is that the Cartesian one takes shift into account while being vulnerable to rotation; and the polar one takes rotation into account while being vulnerable to shift. The shift between two coins is hard to define, given that coins do not have intrinsic positional properties (for example, there is no intrinsic and absolute center). However, it is largely compensated by the initial framing of the coin images, which enables us to arbitrary locate the center in the middle of the frame. The coin does not have an intrinsic orientation either, but there is no easy way to create an arbitrary one for them. It ensues that it is preferable to focus on a polar alignment to deal with a generalized case rather than orienting the coin for a Cartesian alignment.

4.4.5.3 Result and final score

The final step to select the best candidates for our query is to rank them. The geometric alignment builds histograms in which the largest bin contains the most patches that are correctly aligned by the query patches. This quantity is then taken as a score for the candidate. The ordered sequence of candidate is thus given by ordering the scores from the highest to the lowest. In Figure 4.13, the general pipeline of LAPS retrieval is represented. The four candidates are sorted with respect to their best alignment (red bins) and give the final result. Note that the number of final retrieved candidates can be chosen so that it is possible to just obtain the best score or a sequence of several coins.

The whole computation is explained in Algorithm 11. As for LACS, the alignment and the scoring are computed on the fly. That way, the calculation stays independent of the size of the database. This successive update of the best matches is performed by Algorithm 10. The principle is a bit more complicated than just updating one maxima since it is updating a ordered list of matches.

Besides the core of the computation provided by Algorithm 11, we propose some comments and modifications to further improve or adjust the algorithm:

- Taking the score as the largest amount of aligned patches have one obvious drawback that may alter the quality of the result: it depends on the initial number of patches for a given candidate. Indeed, if one coin possesses 100 patches but another has only 50, there is a chance that, even if the latter is a correct match, the other

Algorithm 10: update_matches: updates a sorted list of best matches

Input : J: sorted list of indices (with respect to η)
η: sorted list of scores
j: index
b : score value
K: number of final candidates

```
1  l ← 0
2  while l < K and b > η_l do
3      l ← l + 1 // find where to insert the value
4  if l = 0 then
5      return // nothing to do since it is less than the best matches
6  else
       // if j is already in the list, update it
7      if j ∈ J then
8          k ← position of j in J
9          if η_k < b then
10             η_k ← b
11             J_k ← j
12             j ← 0
               // Correction of inversions in the list which happen
                  because each index appears several times
13             while k + j < K − 1 and η_{k+j} > η_{k+j+1} do
14                 temp_j ← J_{k+j}
15                 temp_η ← η_{k+j}
16                 J_{k+j} ← J_{k+j+1}
17                 η_{k+j} ← η_{k+j+1}
18                 J_{k+j+1} ← temp_j
19                 η_{k+j+1} ← temp_η
20                 j ← j + 1
       // if j is not in the list, insert it
21     else
22         Insert b in η at location k
23         Insert j in J at location k
24         Remove the first element of η
25         Remove the first element of J
```

Algorithm 11: *match*: Finding matches in the database from a coin query

Input : $\mathcal{H}$: system of tables
X: query keypoints
d: query descriptors
m: maximum number of retrieved values
K: number of final candidates

Output : J: sorted best coin indices
η: sorted scores

$hist \leftarrow$ empty dictionary or dictionaries
$\mathrm{J} \leftarrow \underbrace{\{-1,-1,...,-1\}}_{K}$
$hist \leftarrow \underbrace{\{0,0,...,0\}}_{K}$
$\eta \leftarrow \underbrace{\{0,0,...,0\}}_{K}$
$n_X \leftarrow |X|$
for *i from 0 to* $n_X - 1$ **do**
    $d \leftarrow d_i$ `// ` i^{th} `descriptor`
    $R \leftarrow retrieve(\mathcal{H}, d, m)$
    if *R is not None* **then**
        for *all* $r \in R$ **do**
            $j = r(0)$
            $(x, y) = r(1)$
            $\Delta x,\ \Delta y = quantize((x, y), X_i)$ (Equation (4.19) or (4.21))
            $hist[j][\Delta x][\Delta y] \leftarrow hist[j][\Delta x][\Delta y] + 1$ `// ensure it exists first`
            $b = hist[j][\Delta x][\Delta y]$
            update_matches$(\mathrm{J}, \eta, j, b, n_X)$
return J, η

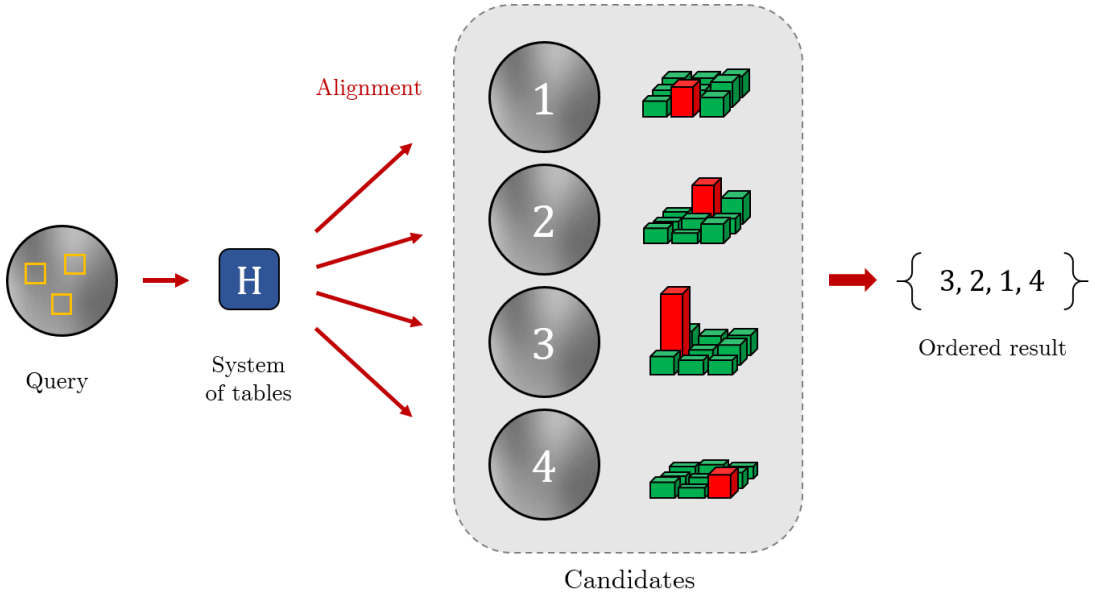


Figure 4.13: Pipeline of the LAPS system. Descriptors from a query coin are used to retrieve values in the system of tables. Retrieved keypoints are then compared to the query's during the geometric alignment phase. Histograms are built for each candidate coin and the final result is an ordered sequence of the highest scores.

is ranked above it. If we assume that all coins have roughly the same number of patches, then this calculation is fine; otherwise, it may cause an incorrect retrieval.

- One way to circumvent this problem is to normalize the score. This means that instead of merely taking the final score as the amount of aligned patches, one divides this score by the total amount of patches in the candidate coin. This provides an answer to the question "what percentage of patches have been retrieved from this candidate?". However, be aware that this percentage may exceed 100%. The reason is the same as for LACS: there may be several indices retrieved per patch. So, for example, if each of the 10 candidate patches have been found, since the retrieval can propose several candidates per query patch, it may end up with one query patch matching two of this candidate patches. The score is therefore 110% in this case.

- Another idea is to propose that instead of "adding 1" for each patch aligned in one bin (line 14 in Algorithm 11), we add the number of tables correctly matched in the previous step. Taking Figure 4.10 as an illustration, each table u returns potential candidates. At this step it is therefore possible, for each coin index j, to retrieve the number of tables returning this index (among others). Hence, one iteration in the histogram adds the number of tables having retrieved j (and not just 1). That way, the algorithm avoids ranking too high the cases where only a

few tables are retrieved at each iteration and fosters candidates coming from many tables.

- One problem that may occur in the retrieval so far is that some coins are wrongly matched with a high score because they have a lot of patches close to each other. In comparison with a coin for which matched patches are evenly spread over the coin, this has to be discarded. One way to make it happen is to calculate successively the total area covered by all the patches in the query for each candidate coin. The resulting score is therefore the total amount of pixels covered in the query, which can be normalized by taking into account the total amount of potential pixels available in the query (equivalent to the sum of the surfaces of all available patches). This can be done easily by updating a list of indices of keypoints for each bin, calculating successively the area via an image emulating the patches and taking its sum as the area. It is, however, rather slow since each area has to be updated each time a new patch is found. If a more efficient algorithm is found instead, this is deemed to be the best way to rank coins.

The tools described in this part of the dissertation composed a rich retrieval framework. Each coin can be quickly matched against a database of already known coins. In the next part, the possibility of not knowing the coin labels is discussed and a clustering system is proposed based on the previously defined tools.

4.5 Die clustering

The main purpose of this study so far has been to match a single query against a database. This assumes that the said database has already been built. An even stronger assumption could even be that all the coins are already known so that coins from the same die are labeled identically. However, what if none of this is known in advance? How should we proceed if an entire bag of coins is unearthed and one seeks to cluster them so as to group coins from the same die? In this section, we will establish some ideas to tackle this issue.

In order to group coins together, we must first establish a correspondence between each of them. In theory, if n_c is the number of objects, this requires ${n_c}^2$ operations. Yet, since we have the possibility to use the LAPS system, this amount can be considerably reduced. It is necessary to store the elements in the system of tables prior to any comparison, and this operation depends linearly on the number of coins to be stored.

A first intuitive method, which somewhat compensates for the consuming task of storing, consists in assessing the similarity of the query with every other coins, then store the query and repeat the operation for every coin left. However, this technique has a major drawback: the asymmetry of the scores ($\eta(i,j) \neq \eta(j,i)$) makes the final grouping appear random, since changing the order of storage significantly changes the score every time. It would be possible to shuffle the tables with each iteration, but that would also mean losing time on that task, which is not worth.

Instead, the grouping has to be kept simple: all the coins are stored in the tables, then each coin is retrieved one by one and excluded from the database simply by ignoring its occurrence during the retrieval. This procedure is presented in Algorithm 12. For each coin tested, the candidates retrieved and their score (percentage of covered area) are used to build a similarity matrix $A \in \mathcal{M}_{n_c}([0,1])$ such that for all i, index of query, and for all k in $[0, K-1]$, rank of candidate of index j, $A_{ij} = \eta_i(k)$.

One key parameter here is the number of final candidates K because it has a direct impact on the interconnections between all coins. For instance, setting $K = 1$ allows only one match and therefore may prevent a cluster of several coins from being grouped together. Alternatively, $K = 10$ allows a lot of connections to be made, but many of them may be irrelevant, even though the scores between incorrect matches should be low.

The similarity matrix A can be thought of as an adjacency matrix. However, the way the matrix is built makes it not symmetrical, precisely because of the asymmetry of the score attribution. This asymmetry is unfortunate since comparing a coin with another should not be an oriented measure. Therefore, we would like to make A symmetrical (or triangular, which is enough). Since pairs of coins that have high matching scores both ways should be favored, we chose to take the average of those scores (Equation (4.23)).[8]

$$\forall i, j \in [0, n_c - 1],\ A_{ij} \leftarrow \frac{A_{ij} + A_{ji}}{2} \tag{4.23}$$

Algorithm 12: *group*: match all coins against each other

Input : $\mathcal{H}$: system of tables
$\{d_0, ..., d_{n_c-1}\}$: list of sets of descriptors
$\{X_0, ..., X_{n_c-1}\}$: list of keypoints
K: number of final candidate for retrieval

$labels \leftarrow \{0, ..., n_c - 1\}$ `// each coin is uniquely labeled`
$A \leftarrow$ empty matrix of size $n_c \times n_c$
$store(\mathcal{H}, d, X, labels)$ `// Algorithm 8`
for *i from 0 to* $n_c - 1$ **do**
    J, $\eta = match(\mathcal{H}, d_i, X_i, K)$ `// Algorithm 11`
    for k *from 0 to* $K - 1$ **do**
        $A_{i,\mathrm{J}[k]} \leftarrow \eta[k]$
return A

[8] Another solution is to keep the maximum score, but this has been found to be less effective.

4.6 Experiments

Now that we have established the tools needed for recognition, we devise some tests whose aim is mainly to gauge the quality of the detectors, the descriptors and the matching system. Such tests will be performed under a same objective, that is to perform die recognition; it means retrieving, for a given query, coins which were made from the same die.

Since we have already pruned off most detectors because of their repeatability, comparative experiments for detectors will be performed on five of them: Gaussian and multi-Gaussian maxima, Harris and adapted Harris, and Laplacian of Gaussian. The descriptors are the one already mentioned previously: HOG and HOME.

The experiments begin with the artificial dataset presented in Chapter 2 to compute quantitative results. Then, tests are performed on the small dataset of real coins presented in Chapter 3 to qualitatively assess the recognition pipeline.

4.6.1 Retrieval of artificial coins

The coins generated artificially are useful to assess the ability of the system to recognize coins from the same die given various parameters and tools. It is sufficiently large and precise – coins from the same die share many exact resemblances with each other – to validate the tools built so far. To be specific, the following experiments use 16 types of 5 dies of 5 coins. Each die deteriorates with each coin struck, as mentioned in Chapter 2.

In order to keep the experiments simple, the final score of the retrieval will be calculated as presented in Algorithm 11, that is the amount of elements in the largest bin of the histogram of alignment.

4.6.1.1 Non-rotated coins at fixed quantization

First, we consider the coins to be set upright, without any angle of rotation. We wish to evaluate the performances of the algorithm of retrieval given various detectors and descriptors. All the parameters are set and listed in Table 4.2. One may notice that the parameter m is quite low. Considering the amount of data, it has been clear from the start that keeping all retrieved values in Algorithm 9 leads to a very long run time. This was an expected risk considering that it directly impacts the complexity. Therefore, the value m was chosen to be low and is set to the average number of collisions γ. Also, the number of final candidates K was limited to 1, making it clear that a result is correct if and only if the best index retrieved is the correct one. The geometry chosen is Cartesian and the geometric parameter ε is 32 px, which gives some comfortable leeway to the geometric alignment. The final score is simply calculated as the number of elements in the largest bin of the histogram of alignment.

There are several aspects of the retrieval that we wish to analyze. First, the detectors that were previously selected and the descriptors that were chosen can be compared through the LAPS system. The system is fixed with the aforementioned set of parameters and the quantization parameter is set to $q = 10$. The choice of this value comes from

Parameter	Value
β	0.8
n_u	196
n_v	8
m	5
ε	32
$(\varepsilon_\rho, \varepsilon_\theta)$	(50,15°)
K	1

Table 4.2: Parameters of the experiments for the retrieval of artificial coins.

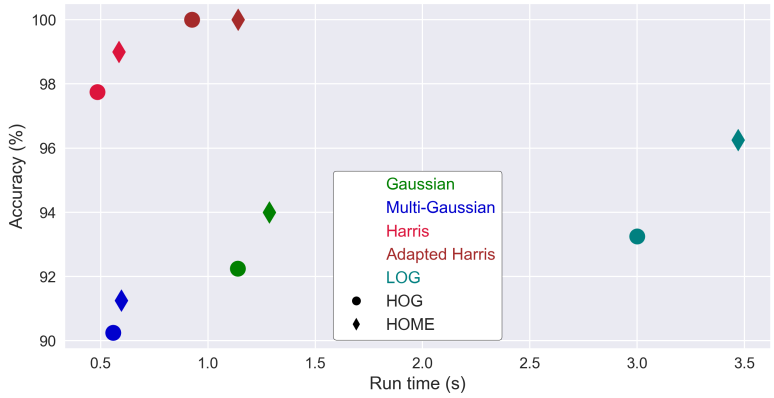


Figure 4.14: Accuracy of retrieval against the average run time per coin. There are 5 colors for the detectors and 2 shapes for the descriptors. The results show a high accuracy and a quite low computational time for each method. Note that the ordinate axis begins at 90%.

initial tests with the algorithms to verify its behavior, its speed and its accuracy. Such a value is satisfactory in both run time and accuracy to pursue more experiments.

The protocol is the following. For each detector, keypoints are calculated and patches are extracted; descriptors are calculated with both HOG and HOME methods; and the database is stored in the LAPS system which contains all the coins (meaning all the descriptors of all patches). Whenever a coin is chosen as a query, every instance of its index is ignored so that the system behaves just as if this coin did not belong to the database. This way, for each method, one storage is required and the complete test requires 10 runs for 400 queries. The final results are the accuracy and the average run time per coin. The accuracy is calculated as the number of correct matches (the retrieved die is the same) over the number of queries. The run time corresponds to how much time it takes for the system to retrieve a candidate from an already built database and for already computed query keypoints and descriptors.

The results are presented in Figure 4.14. In terms of accuracy, all the results are very satisfactory. It does not fall below 90% and reaches up to 100%. It appears that both

m	1	5	10	20	50	100
Accuracy (%)	100	100	100	100	99.75	99.5
Run time (s)	1.81	1.71	1.74	1.69	1.88	1.93

Table 4.3: Accuracy and retrieval time with HOME against the number of retrieved values m.

Harris detectors perform particularly well, ahead of LoG, Gaussian and multi-Gaussian. Multi-Gaussian has the lowest accuracy. This is probably coming from the fact that there is a low amount of interest points, which does not enable a rich description.

In terms of retrieval time per coin, one can say that results are also satisfactory: run times are below 4 seconds and most results revolve around 1 second of run time. It is quite noticeable that the LoG stands apart from the other detectors. The main reason such variations in run time occur at this level is that there are variations in the number of keypoints, depending on the method, which has been discussed in Section 4.2.4. Since there is no reason for the relationship between the number of keypoints and the run time not to be linear, it is also likely that the points detected via the LoG are provoking more collisions than the others (for instance, there are twice as many points than there are Gaussian maxima, yet the run time has more than tripled). Note that the Harris detector is very efficient in pinpointing highly informative feature points since it detects a median amount of about 40 points per coin (Figure 4.3) while still achieving very high accuracy of retrieval. It therefore enables an accurate and very fast matching. In general, one wishes for the smallest run time, but wants at the same time a sufficient accuracy. Given the results in Figure 4.14, it seems reasonable, in this context, to choose either the simple Harris detector for speed over accuracy or the adapted Harris for accuracy over speed, while keeping both of them high.

As stated before, the number of retrieved values should be determined by the number of collisions in the database. In Figure 4.15, the distribution of collisions in the tables for HOME descriptor is shown. It was mentioned before that the number of collisions cannot be lower than 1 since a bin is created if and only if there is a need for it. The histogram reveals that there is a majority of bins (65.2%) composed of only 1 value. In general 88.5% of the bins contain at most 5 values. The maximum number of collisions obtained here is 697. Collisions superior to 10 are so infrequent that it is likely that deleting those bins altogether would not affect the retrieval. The average value of collisions is $\gamma = 5.06$ for HOME and $\gamma = 4.60$ for HOG.

In Table 4.3, we present the results of the retrieval accuracy and run time obtained with HOME with respect to the number of values m retrieved in each bin. It is clear from these data that there is no significant change in terms of accuracy and that it is even worse to increase m too much. The run time also proves to get slightly longer as m increases, which is to be expected.

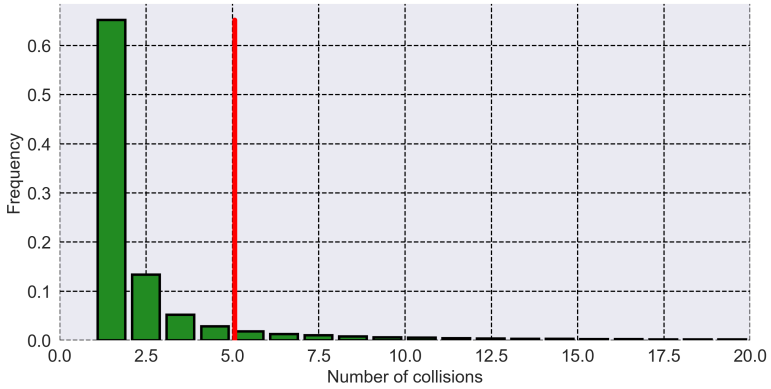


Figure 4.15: Distribution of collisions in the database for HOME ($q = 10$). The red vertical line is the average number of collisions $\gamma = 5.06$. There are a total of 663887 values stored.

4.6.1.2 Accuracy and run time against quantization

The trade-off between accuracy and run time is an important component of the retrieval with the LAPS system. Because it uses associative arrays for storage and because of the collisions inside those arrays, there is a logical influence of the availability of data on the retrieval time.

The parameter responsible for such a trade-off is the quantization q, which determines the width of cells of the associative arrays. As mentioned in the previous paragraph, q has been set to 10 to compare the various detectors and descriptors. Here we shall see how fast and how accurate this is meant to be in theory, and by how much this value impacts the retrieval. For this purpose, we choose only one detector, the adapted Harris; the accuracy ranking between detectors should not change since it does not depend on q itself. However, we choose to compare the descriptors because the values they compute may by themselves have an influence on the collisions, which does not have a linear relationship with q.

In Figure 4.16, the accuracy of retrieval against q is presented. As one can expect, the accuracy is dependent on q and decreases when q increases. This is fairly logical: the cells contains less and less values (collisions are fewer and fewer), which requires for the descriptor values to be more and more precise. Of course, the accuracy benefits from a lot of information, since the geometric alignment prunes off noisy data. HOME scores a higher accuracy on average. Note that the amplitude of the slope is larger for HOG, increasing the gap between it and HOME. Overall, one can expect a very satisfactory retrieval ($> 90\%$) up until around $q = 30$.

Run time against q is analyzed in Figure 4.17. High values of q (> 50) produce a very fast retrieval, with under 0.5 s per coin (around 3 min for all the coins). The behavior of the curve is not at all linear. There is a seemingly hyperbolic increase of the run time as q decreases under 50. A quantization of 10, used previously, gives a run time superior

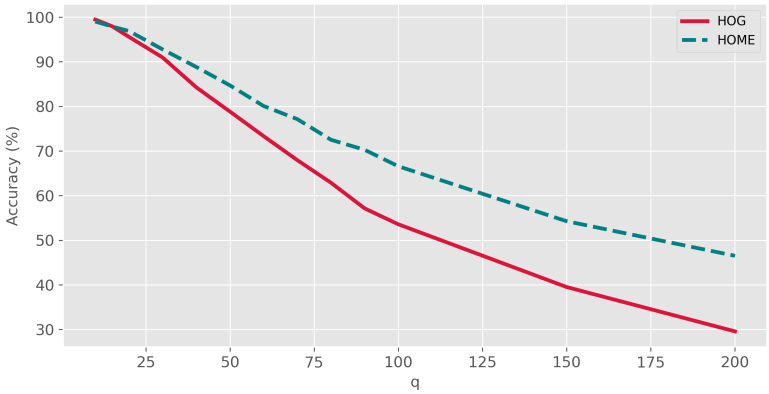


Figure 4.16: Accuracy against quantization from $q = 10$ to $q = 200$. The results for both descriptors are shown.

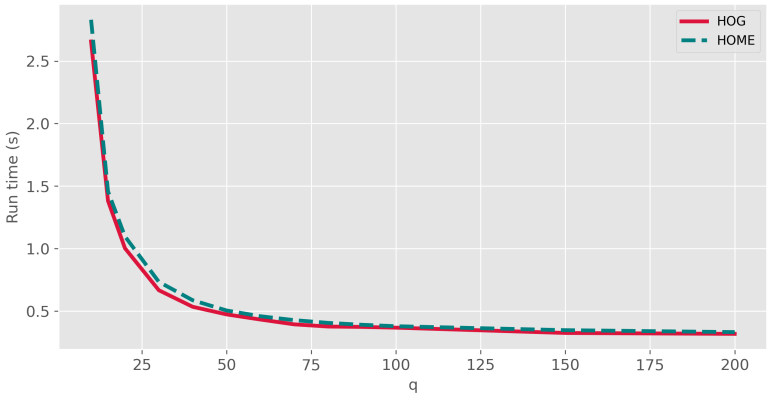
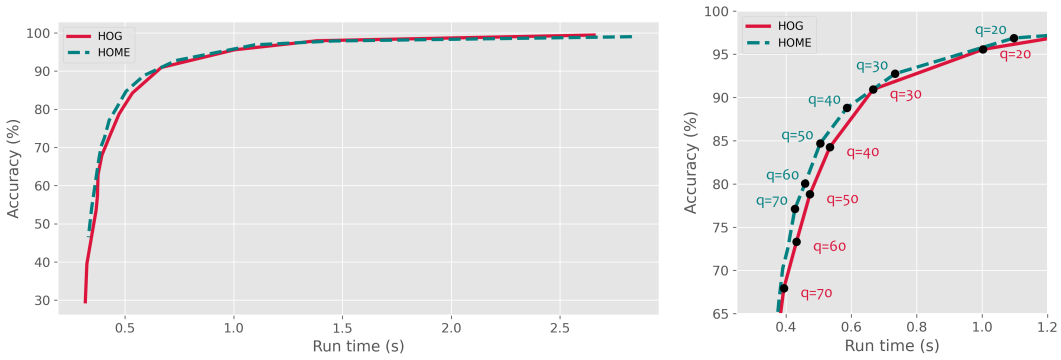


Figure 4.17: Run time against quantization from $q = 10$ to $q = 200$. The results for both descriptors are shown.

to 2.5 s per coin (around 17 min for all the coins). This is particularly long for such a small batch, and would be prohibitory for very large databases. It is advisable in this case to study the trade-off between accuracy and speed and select q according to the requirements of the task at hand.

Note that LAPS takes slightly more time with the HOME descriptor, especially for low values of q. As already explained, this phenomenon is likely due to HOME producing more collisions due to its values; it all depends on the difference of distributions of such values between HOG and HOME.

To summarize those two graphs, it is interesting to plot the graph of accuracy against run time, which is displayed in Figure 4.18. The dynamic of the curve is the same as for the run time, which is normal given that the accuracy is approximately a linear function of q. Thus there is an abrupt diminution of accuracy with very low run times and an abrupt increase of run time at high accuracy. There are two asymptotes visible



(a) Full view of the graph. Two asymptotes are visible at both ends of the graph.

(b) Zoom on the elbow part of the graph.

Figure 4.18: Accuracy against run time trade-off.

in Figure 4.18a, which essentially mean that it is not worth investing too much time in the retrieval and that accuracy falls really fast if one aims at very low retrieval times. The most relevant part of this trade-off is the elbow region, which has been foregrounded in Figure 4.18b. One can notice that, in general, HOME requires slightly less time than HOG for the same accuracy. Furthermore, a critical point here can be said to be around $q = 30$. At that location, one begins to lose more accuracy than one gains run time by decreasing q. This assertion is symmetrical: one loses more run time than one gains accuracy by increasing q. The choice is again dependent on the user, but we see a fair amount of leeway in the range $q \in [20, 70]$ to adjust the system at will.

Finally, in Figure 4.19 is represented the run time per coin as a function of the number of collisions. As a reminder, the collisions are defined as the number of elements per cell of the associative arrays (last layer). The number of collisions here is the average number of elements per cell in all the system. This number cannot be lower than 1.

Of course, the quantization has an impact on the collisions since it defines the width of each cell. The relationship between collisions and run time is linear. However, just as for the run time, the number of collisions increases inversely to q. This graph provides an answer to our question about the difference in run times between HOG and HOME. It is clear here that the number of collisions is always higher for HOME than for HOG, and this difference decreases with q.

4.6.1.3 Slightly rotated coins

In this section, we would like to assess the robustness of the current implementation with respect to rotation. Of course, since the geometry is Cartesian and the patches are upright (neither the center of the coin nor the local gradient are taken into account), the results are expected to be poor. However, it is still intriguing to see how fast the quality of the results falls if the rotation stays reasonably slight. One can imagine the inherent constraints of acquisition: the coin has to be placed below the camera, and by

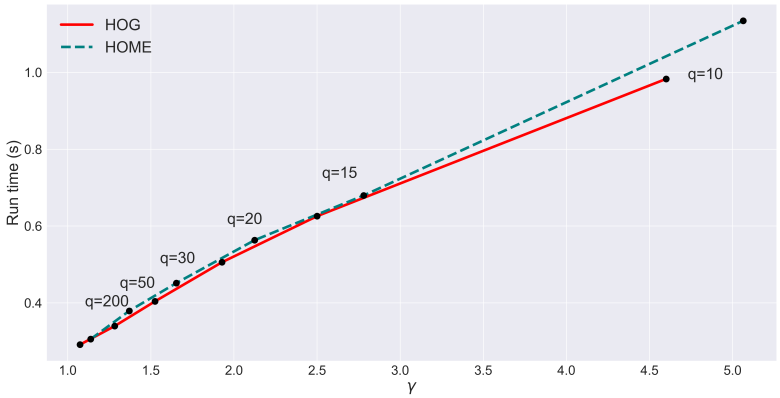


Figure 4.19: Run time per coins against the average number of collisions γ in the system database.

doing so is given an arbitrary rotation based on human perception. It would therefore be relevant to know if this simple error has a significant impact on the retrieval.

The protocol is straightforward: for each angle of rotation, a duplicate of the database is created with all its elements rotated by that amount. The original database is stored in the LAPS system and each rotated element is tested against it.[9] The angle range has to be short, therefore we select positive angles inferior to 10°. This is sufficient to show a significant decrease in accuracy, as can be observed in Figure 4.20. In these data, accuracy starts to plummet after 3° and tends towards 0. The results are not subject to an all-or-nothing function; the diminution is gradual, although rapid.

There are, as it has been observed before, very few differences between HOG and HOME. However, there is a clear division between the various detectors. The adapted Harris method is always ahead, staying above 80% even for a 5° rotation. There is also a clear gap between the Harris methods and the three others, which seems to widen after 4°.

In general, if the coin has a clear mark with respect to which it can be aligned, the Cartesian method may suffice. Yet, it seems not advisable to bet on the non-polar geometry and non rotation-invariant patch extraction to deal with small rotations.

4.6.1.4 Rotated coins

Following this analysis, we shall treat the case of fully unknown rotations of the coins. In this case, the patches are extracted via either the center of the coin or via local gradient pooling and the retrieval algorithm uses the polar coordinates and the parameters $(\varepsilon_\rho, \varepsilon_\theta) = (50, 15)$. Each coin of the database is randomly rotated around the center of the frame (512,512).[10] The experiment follows the same protocol as for the other tests,

[9] The index of the query coin is not taken into account in the retrieval

[10] The choice of the center hardly matters in reality because the coins are randomly rotated. Again, the center does not have any intrinsic definition.

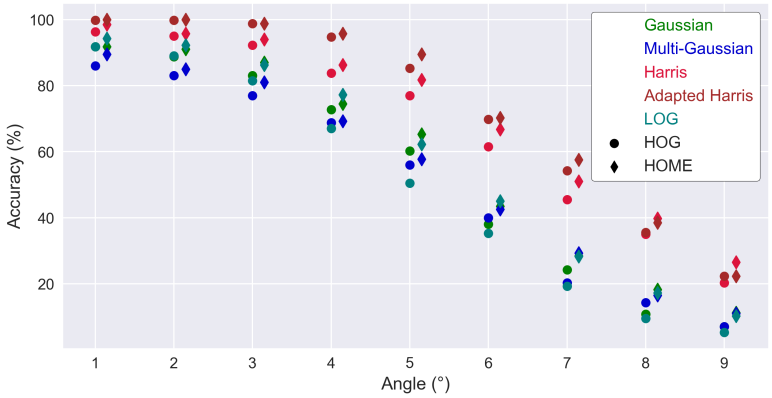


Figure 4.20: Accuracy of retrieval with a slight rotation. HOG and HOME results are displayed next to each other for each angle.

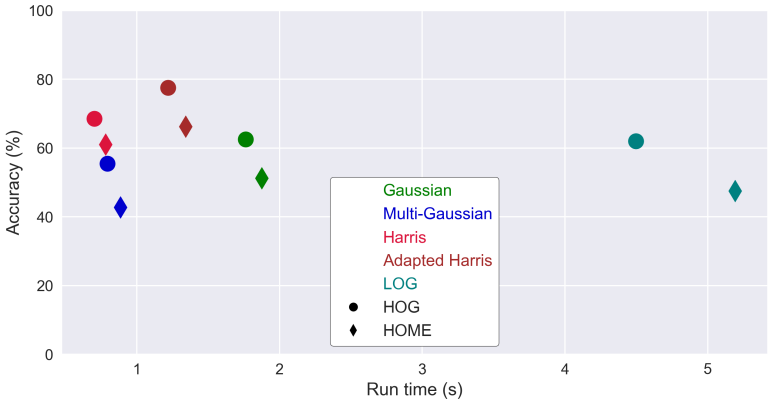
except this time there are two methods for extracting the patches. The same detectors are employed as well as the same descriptors.

The results are presented in Figure 4.21. As a general observation, these results are significantly worse than for unrotated coins. Both methods reach similar results overall, with a slight advantage for local pooling. The detectors follow the same order as already established, with the adapted Harris ahead and reaching up to 80.5% using local gradient and HOG. However, the order of the descriptors has changed: HOG seems to prevail over HOME in the majority of cases. We may infer from this observation that taking the derivative component of the energy map is more robust to rotation than is the direct use of the energy map and orientation map for description. Also note that this order is systematic in the case of the extraction via the coin's center (Figure 4.21a) whereas it happens only for both Harris detectors when using the local gradient (Figure 4.21b).

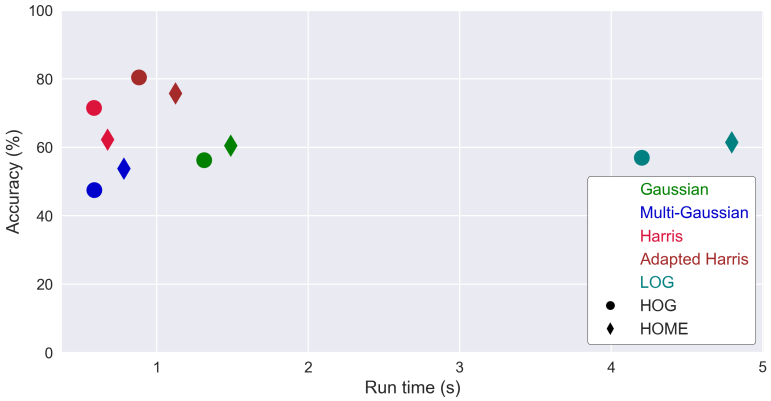
Finally, in the same way we did in Chapter 3, we modify the number of coins per die to see how it impacts the retrieval results for both straight and rotated coins. The results can be seen in Table 4.4. The accuracy of the retrieval performed on straight coin is remarkably high, even with only 1 coin, for both HOG and HOME. Rotated coins are, as we just discussed, much more difficult to retrieve. Yet, better results are observed than with contours (Figure 3.7), starting at 77.7 % for only 1 coin using HOG and quickly improving by adding more coins. As always, it is advisable to use as many coins as available to obtain better results, but this system can yield satisfactory results even with very low amounts of examples.

4.6.1.5 SIFT and LIDRIC

Finally, here is a brief summary of the performance of other general methods of detection, description and comparison for upright coins. Because SIFT is an well-established solution programmed within code libraries, it is easy to test. It was used on the artificial



(a) Coin centered.



(b) Local pooling.

Figure 4.21: Accuracy against run time trade-off for two methods of patch extraction with rotation.

	Straight		Rotated	
# coins	HOG	HOME	HOG	HOME
1	98.1	96.6	77.7	72.0
2	99.7	99.7	91.3	88.0
3	100	100	96.6	94.9
4	100	100	98.0	97.3

Table 4.4: Accuracy (%) of the retrieval against the number of coins per die.

Method	SIFT on $\mathcal{E}$ (L_2)	SIFT on $\mathcal{E}$ (LAPS)	SIFT on images (L_2)	LIDRIC on images
Accuracy (%)	91.8	53.0	80.8	63.3
Total run time (h)	13.3	1.14	3.16	> 140

Table 4.5: Accuracy and retrieval time for various methods. The retrieval times displayed are just raw indicators of speed because not different programming languages were used.

dataset, but since it requires only one image, the matching has been done on two possible images: the energy map $\mathcal{E}$ and a single image taken with three light orientations, placed in the manner suggested by [40]. The overall angle is always the same, but varies slightly via a small random angle. The LIDRIC method has been only used with these images, since it aims at recognizing images of coins while being robust to lighting conditions. In these methods, both keypoints and descriptors have been used as they are.

The matching technique for SIFT is traditionally an L_2 distance which makes the total number of comparisons quadratic. The same remark holds for the LIDRIC, although the similarity score provided by the method is used; points are not detected in this case; they are simply found on a dense regular grid. Finally, SIFT was tested with our matching method. The number of tables and the number of nested arrays are set to 16 and 8, because the SIFT descriptor is built from the same spatial logic as HOG. The quantization parameter q was set to 20 for speed purpose.[11]

The results are shown in Table 4.5. The retrieval times should only be considered as an indication. The computation were not run with the same programming language for each of them (python for LAPS, python + fortran for L_2 comparisons and Matlab for LIDRIC); small differences are not interpretable.

It makes sense that, given the visual proximity of the coins, the results are rather good. One can appreciate that, all other things being equal, performing on the energy map improves the accuracy. Still, satisfactory accuracy values are obtained on images, while also diminishing the total run time. This is because of the number of interest points found in the energy map which largely exceeds the one found in simple images. Obviously, using LAPS with SIFT radically improves the computation time (3 times as fast as SIFT on images). However, the consequence of such a choice of parameter ($q = 20$) is that accuracy is really impacted. The main conclusions on this aspect is that SIFT points are too numerous and many of them are spurious. Descriptor values tend to collide with each other, which leads to an impractical number of collisions. Also, it is possible that the number of dimensions of SIFT does not favor this type of matching calculation. Finally, LIDRIC seems not sufficiently discriminatory and is very expensive in terms of computation time. LIDRIC is supposed to be very robust to illumination changes, yet it does not seem to take advantage of this property in this experiment.

[11] This leads to a average number of collisions of $\gamma \approx 3$ whereas using $q = 10$ produces $\gamma \approx 30$ collisions, which causes very long retrieval times.

4.6.2 Retrieval of real coins

So far, the experiments have been based on an artificial dataset proposed in Section 2.4. While this is useful to infer statistical results about the retrieval pipeline, it constrains the tuning of the parameters to this specific dataset, which may of course slightly differ for real acquisitions. The dataset used is the same as in Section 2.3.

Through the analysis of the retrieval on the dataset, one can aim at answering several questions. Are the parameters determined for the artificial dataset transferable to this dataset? Is it still sufficient for pairing coins struck with the same die? Is there a significant gap in scores in order to adequately discriminate the dies?

4.6.2.1 Dataset

In this section, we present some tests performed on real coins. For the experiments, we use the dataset presented in Chapter 3 in Figure 2.16. Note that we can group together coins which look alike. In this section as in the previous one, "types" of coins are not dealt with. The definition of a type remains vague, but it is still interesting for the algorithm to try to match coins that resemble each other for lack of anything better. Therefore, the interpretation of the tests will mostly be based on visual analysis, given that, except for dies, resemblance is not well-defined.

4.6.2.2 Parametrization

The parameters for this dataset are slightly different from the one chosen in the previous section on artificial coins. Those parameters (including those for the polar geometry) are listed in Table 4.6. One notices that β has been set to 1.0, which is a radical choice indicating that every patch should be entirely inside the mask of the coin. It has been observed that many points can be detected at the borders if β is not high enough, which greatly impairs the results (borders often look alike and are not a discriminative element of the coin).

The number of accepted collided elements per table m has been raised to 10, a value still close to the one previously chosen. The database is shuffled (the last array of each table) so that the order of storage does not statistically impact the retrieval. Moreover, the number of final best candidates should allow a visual selection by the user. Therefore, it has been set and limited to 5.

The tolerance for the geometric alignment has been increased to 64. This is an empirical decision; while the difference in ranking between 32 and 64 is not significantly large, allowing that much error for the alignment provides a richer description while still representing only a sixteenth of the frame length.

Based on the results of the previous section, the adapted Harris detector will be used and both descriptors (HOG and HOME) will be tested.

Parameter	Value
β	1.0
n_u	196
n_v	8
m	10
ε	64
$(\varepsilon_\rho, \varepsilon_\theta)$	(50,15°)
K	5

Table 4.6: Parameters of the experiments for the retrieval of real coins

4.6.2.3 Scoring methodology

It has been observed that the simple score employed in Section 4.6.1 is not as relevant here as it is for the artificial dataset. Firstly, it greatly limits the accuracy of the scoring; indeed, this method works statistically, providing large score if and only if the geometric alignment is respected for a large number of points. If only a few keypoints have been detected or if the parameter ε is too large, the scores may be low and close to each other. For instance, see in Figure 4.22a, the correct coins are retrieved, but the scores are not discriminative.

As listed in Section 4.4.5.3, there are various other way to calculate the score. In order to make an interpretable expression of the final result, we chose to explore the percentage of available query area retrieved. In essence, this is similar to the percentage of patches found, but it takes into account the proximity of such patches (overlap should not be counted twice). However, this score takes more time to compute than more basic ones and thus was not suited for the previous extensive calculations on run times.

The differences between the two scoring methods are presented in Figure 4.22b. There is not a significant difference in the candidates, but their order varies. In particular, it enables a discrimination between the two last coins. The results are more interpretable and they are less influenced by the number of keypoints in the candidates. Surjections are ignored because they cover the same area.

Of course, by no means does this prove to be the best way to rank the results; after all, the differences with other methods are not major. Nonetheless, we suggest that taking a measurement of the covered area might be the preferred way to infer meaningful conclusions about the retrieval.

4.6.2.4 Visual inspection

In order to analyze the results of the retrieval, we tested the algorithm with both HOME and HOG. The queries used are coins from dies for which more than one coin is available. Only five of them are displayed here. As stated in Section 4.6.2.2, the algorithm returns $K = 5$ candidates. The results are presented in Figures 4.23 and 4.24. Several comments can be made:

(a) Amount in the largest bin of alignment.

(b) Percentage of area covered in the query.

Figure 4.22: Difference between two types of scoring method with HOME descriptor. Red squares indicate the correct (desired) matches. The amount in the largest bin of the histogram of alignment may be very small and scores may be close to each other whereas taking the percentage of the available area in the query gives finer results and is more interpretable.

- Firstly, in both HOG and HOME results, the dies are quite correctly recognized and paired with their query. The only exception is coin (c), for which the correct candidate (red square) is found at the second place. Coin (d) is the only one whose die is represented by five different coins. Not all exemplars are displayed here, but, for every single one of them, the algorithm correctly ranks the others as the first four best matches. In other words, those coins are implicitly gathered together by the LAPS system.

- Secondly, in terms of scores, the algorithm neatly ranks the coins so that coins from the same die have on average greater scores than coins from different dies. See for example coin (f), the first coin is the correct candidate (red square) and the second and third are from different dies; they are highly ranked but their score is lower than the correct candidate. This difference in scoring is larger for HOG descriptors, for which it amounts to 39.8 points. Also note that the four (correct) best candidates for query (d) all feature high scores. Again, this is especially true for HOG. On average, the trend seems to be that HOG discriminates better correct candidates from incorrect candidates when looking only at the scores. This is not true though for coins (a), (b) and (c), for which HOME provides a larger distinction between correct and incorrect matches.

- Moreover, the algorithm still tries to group lookalike coins, even though they are not from the same die. This is particularly clear for coins (a), (f) for HOG, and

(g). Scores are in general not as high as for correct candidates, as they should not, but nonetheless are ranked in top positions.

- In addition, there are some mistakes that the algorithm makes which are not merely explainable by simple resemblances. Such mistakes are for example the second place in coin (e) (for HOG and for HOME, although the latter is even more mistaken for (e) in general) or the third place in (c) (HOME), for which the score is quite higher than the next candidates even if they are visually closer to the query.

- A curious phenomenon can be noticed on every pair query/candidate: the score attributed to one of them when the other is the query is not the same when the roles are reversed. In other words, there is an asymmetry in the retrieval. As a first clue, the number of retrieved values per table m seems to make a perfect culprit since it is the fundamental limiter in the system. This has been empirically shown to be wrong, as setting $m = \infty$ also manifests an asymmetry. Nor it is due to the number of final candidates K.

  In general though, the difference is not large. It could indicate that the mere quantization process in both the tables and the geometric alignment are involved in the phenomenon. This could be just a slight detail if it did not change the order of the candidates in some way, yet it does significantly so. Further investigation could be to identify the roots of this asymmetry and lessen its effects.

- Finally, these analyses unravel some intricacies in the retrieval, especially between HOG and HOME. Both descriptors perform roughly the same job in pinpointing correct candidates. There is however, at least visually, a seemingly better ranking when using HOG because lookalike coins are also matched with some queries, although with a lesser score. Actually, this tendency is not completely true and systematic. For instance, coins (b) and (f) (which could be gathered together in the same cluster by resemblance) behave somewhat differently when using HOG and when using HOME. With HOG, coin (b) returns three inappropriate candidates, while HOME gathers visually correct candidates. The opposite is true for coin (f). Also, scores are differently attributed. Scores of coins (a), (b) and (c), for instance, are higher with HOME than with HOG. The opposite occurs for coins (e), (f) and (g), while for coin (d) scores are roughly the same.

Accordingly to this analysis, there may be some coin features which differentiate the results between HOG and HOME descriptors, and those features may well have to do with the thickness of the ridges within the energy maps. Indeed, it appears that, for coins (a), (b) and (c), ridges are thicker than those in coins (e), (f), (g) and (h). Said another way, the former present large, thick details while the latter are made of fine, thin details. This would imply that HOME best detects large features while HOG spots fine features. Note that HOME does not perform badly on coins (f) and (g); it just performs worse than HOG. A middle ground seems to be found in coin (d), which may be uniquely interpreted both in terms of large and fine gradient variations.

Based on these remarks, it is tempting to combine both descriptors in order to enjoy both their strengths in their ability to highlight meaningful features. It is, given the modularity of the LAPS system, rather easy to perform this combination. The descriptors can be concatenated such that the final vector comprises 392 dimensions. Therefore, it is sufficient to change only the number of tables required in the system to process this new descriptor. No other change in terms of parameters has been made. As an aggregation of both HOG and HOME, we simply named this descriptor HO2. The results are presented in Figure 4.25.

There is no major difference in terms of correct retrieval of candidates. However, slight improvements have been made in terms of retrieval of lookalike coins, as well as in the attribution of scores. The results are not perfect, but nonetheless improved the coherence of the retrieval. Coins (a), (b) and (c) are clustered nicely together while coins (e) and (f) are grouped with better candidates than for HOME, albeit worse than for HOG. No obvious change has been observed for coin (d), while coins (g) and (h) are relevantly matched only with lookalike coins.

In Figure 4.26 one can see which patches have been extracted and paired for several coins and their best candidate. A fairly logical behavior is observed; patches match their visual equivalent in the candidate, and the matched parts are visually relevant: for instance the hair in (a), left symbol and the legs in (c) and the legs in (d). There may be spurious patches or even surjections, but they are not prevalent and do not corrupt the score.

It is also interesting to look at matches which do not originate from the same die. Some are presented in Figure 4.27. On average, the matches are very similar to the query. In (a), for instance, face features and some parts of the hair are quite resembling. Sometimes, the resemblance is such that a very high score is reached. This is what happened for coin (c); both coins have the exact same features, but are not from the same die. This tendency may confuse an user as this is not *per se* a good match. However, it still indicates a valuable kinship between both coins, and a good eye can anyway eventually rule them out. Sometimes, very few matches are found (see coin (c)) but still manage to reach the status of best candidate. Finally, note in coin (d) how the face is relevantly highlighted. The score is low; this is not a good match at all. Yet, it indicates that the algorithm is still capable of pinpointing common features which share similar relative positions.

4.6.2.5 Numerical evaluation

Besides the visual analysis, we also would like to provide some measurements of the quality of the results. This is a necessary step to fine tune some parameters and give more insight about the behavior of the retrieval.

Collisions

First of all, the average number of collisions γ is 6.27 for HOG and 7.77 for HOME, suggesting that the former has an overall better value distribution in the database than

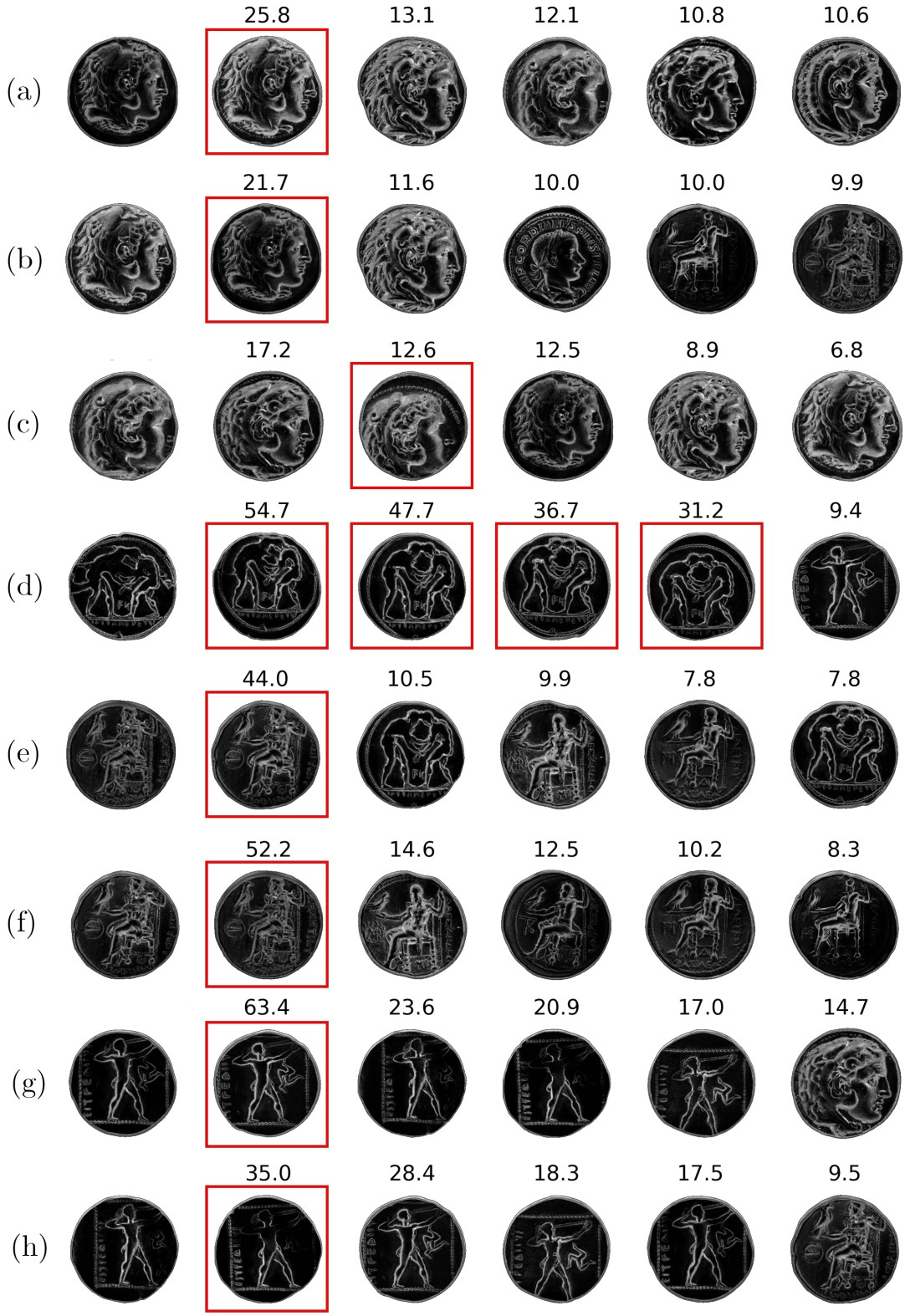


Figure 4.23: Retrieval results obtained using HOG descriptor. The query is the leftmost coin and the candidates are ordered from left to right after it. Red squares indicate the correct expected candidates.

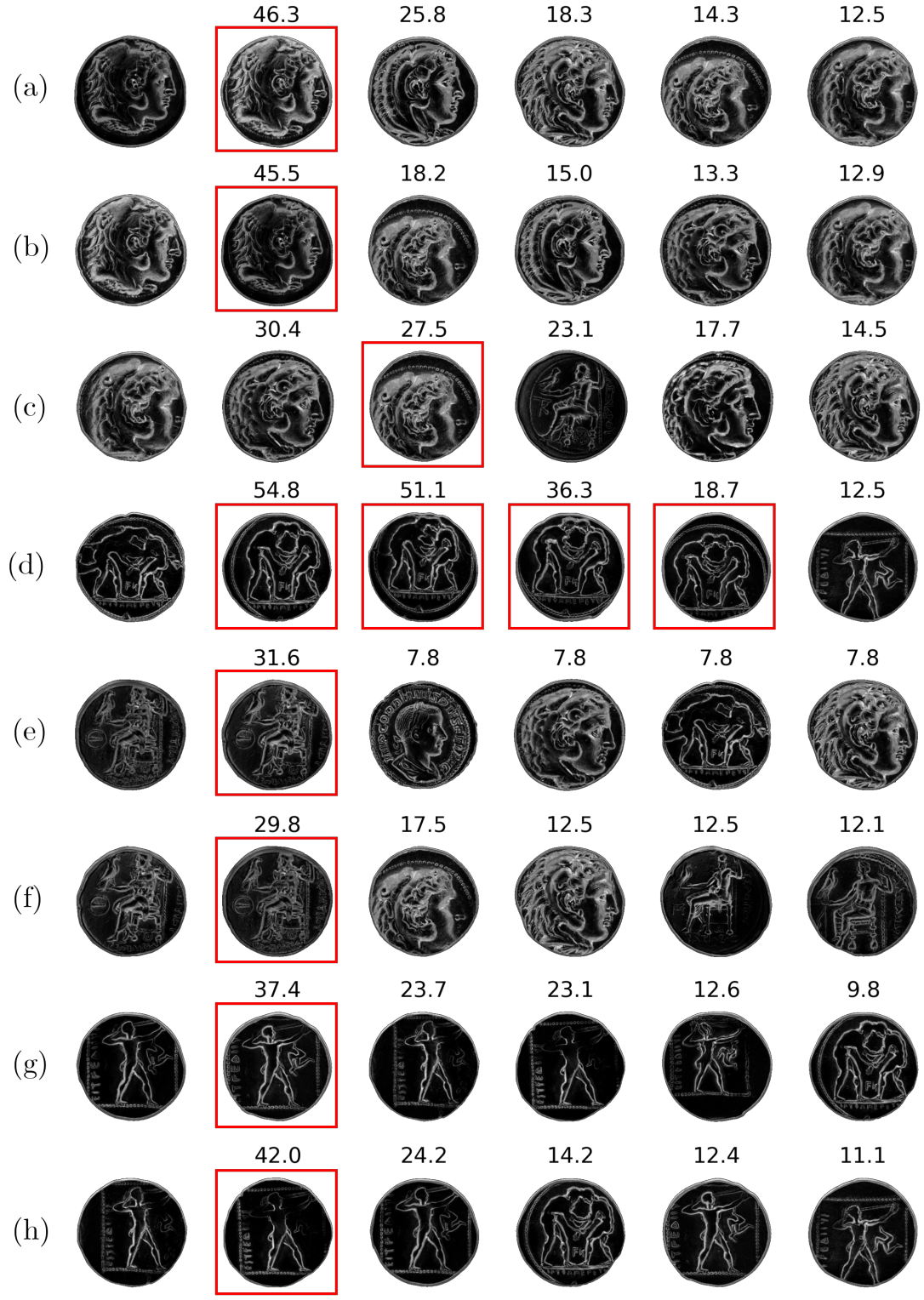


Figure 4.24: Retrieval results obtained using HOME descriptor. The query is the left-most coin and the candidates are ordered from left to right after it. Red squares indicate the correct expected candidates.

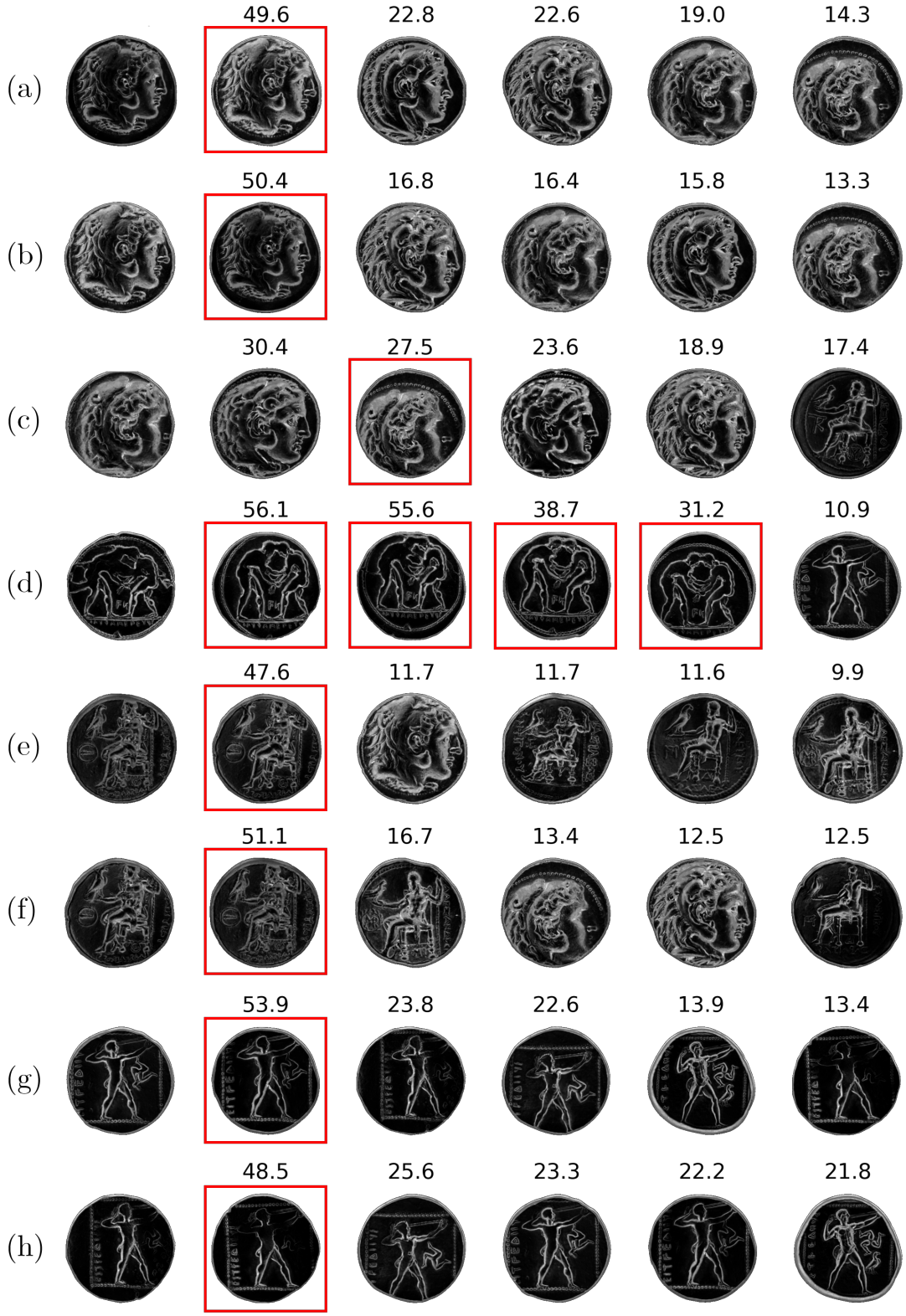
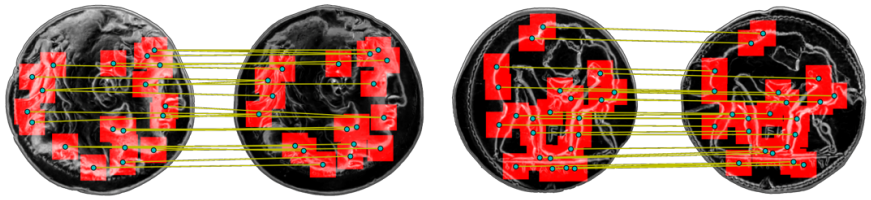
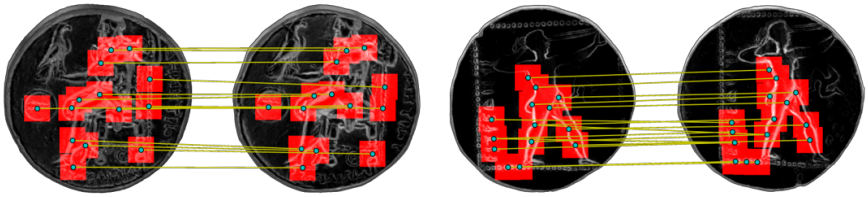


Figure 4.25: Retrieval results obtained using the HO2 descriptor. The query is the leftmost coin and the candidates are ordered from left to right after it. Red squares indicate the correct expected candidates.



(a) 50.7% (b) 63.0%



(c) 47.9% (d) 42.8%

Figure 4.26: Details of the matches between patches for targeted coins with the HO2 descriptor. The score (percentage of maximum area covered) is displayed for each match.

(a) 40.5% (b) 63.0%

(c) 15.2% (d) 9.7%

Figure 4.27: Details of the matches between patches for lookalike coins (not from the same die) with the HO2 descriptor. The score (percentage of maximum area covered) is displayed for each match.

Descriptor	Min	Max	Mean	*st.d.*
HOG	3.48	41.6	21.6	11.1
HOME	3.91	30.2	17.2	8.16
HO2	4.18	36.5	24.4	9.78

Table 4.7: Minimum, maximum, mean and standard deviation of score difference between the last correct match and the next candidate. The larger the difference, the more significant the choice of the correct candidate is, and the more discriminative the retrieval is.

HOME. Note that, curiously, this value is larger than that of the entire artificial coins database ($\gamma = 5.06$ for HOME). Yet, there are 30268 values stored in the database (the artificial dataset had 663887, around 20 times more). The number of patches extracted per coin does not seem to explain this observation, since there are even less keypoints (around 27) detected on this dataset than in the artificial one. However, there are around 60% of the cells containing values in the artificial database contained only 1 value. In the real dataset, this number is 40%. There are therefore more values in the tails of the distribution for the real coins than there are for the artificial coins. The dataset remains too small to derive conclusions, but it may be that the real coins feature on average more redundant values than the artificial coins, which could originate from more values being accepted in the tables by thresholding (Algorithm 8 line 4).

Discrimination behavior

The ability of the algorithm to discriminate is an important component of its performance. It has already been briefly explained in the aforementioned remarks that the descriptors do not seem to behave the same way in this regard. We chose to calculate the score difference between the last correct candidate and the next right after it to evaluate this behavior. In Table 4.7, these results are listed. As already noted, HOG discriminates better than HOME, albeit featuring a quite largely spread distribution. HOME is a bit more centered around its mean, which signifies that it can be less volatile in making distinctions between true candidates and false candidates. The HO2, finally, is prone to merge both behaviors and obtains a larger mean difference than both HOG and HOME while maintaining a lesser variance than that of HOG.

Impact of shuffling

Given the processing framework, one hopes that storing biases will be statistically corrected by shuffling the arrays (as suggested in Section 4.4.3, since a restricted number of values m are retrieved, it is reasonable to shuffle the last arrays of the system). In order to assess the impact of randomness on the retrieval, we perform 100 runs with new a random shuffle each time. The results are processed in the following manner. All ranks are separated and, for each of them, the number of differences (i.e. when the coin index is not the same) between each run is calculated. There are therefore 4950 comparisons per rank and per coin. The average percentage of difference is computed for all coins

Rank	1	2	3	4	5
All coins	14.3	32.2	42.1	49.6	59.7
Targeted coins	2.37	16.3	23.9	33.0	44.7

Table 4.8: Impact of shuffling the tables on the retrieval. The results are the proportions (%) of differences between 100 different runs of the algorithm, for each rank.

and we are left with five percentages, one by rank. Furthermore, the calculation has also been performed only for target coins, which are coins having at least one counterpart from the same die.

The differences are enumerated in Table 4.8. There is a clear impact of the shuffle on the various retrieved coins. It is logical that the percentage of error increases with the rank, since candidates are statistically less and less prone to be correct matches and therefore may have scores close to each other. Targeted coins from same dies are more robust to this phenomenon. This can be explained by the difference between true and false matches listed in Table 4.7. Given these results, it seems well advised not to rely entirely on the algorithm to attribute kinship between coin for rank superior to 1 since, even for targeted coins, no less than 16% variations in coin placement is expected.

Retrieval parameter m

Another characteristic of the retrieval is that it restrains the number of values retrieved per table m, which acts as a pre-selection parameter for the retrieval. In order to evaluate the behavior of the algorithm with respect to this parameter, we have to use some specific measurements. Given that the outcome of the algorithm is an ordered list of candidates, it is relevant to use the mean average precision. For any ordered list of results, one calculates the precision at rank k by counting the number of relevant elements within the first k candidates:

$$P(k) = \frac{\#\text{ relevant elements}}{k}$$

It follows that for K total elements in the list, the Average Precision at K is defined as :[12]

$$AP(K) = \frac{1}{K}\sum_{k=1}^{K} P(k)$$

Finally, for N queries tested against the database, the Mean Average Precision (MAP) is obtained via Equation (4.24), where the number K is set to be the maximum number of candidates retrieved by the algorithm.

$$MAP(N,K) = \frac{1}{N}\sum_{i=0}^{N} AP_i(K) \tag{4.24}$$

[12]In reality, one has to take into account that K may be superior to the total number of possible relevant documents. In this case, this latter number should be used to divide the sum.

With these definitions in mind, we can simply evaluate the Mean Average Precision over all our database for each desired values of m. Of course, not all coins are relevant; we only focus our attention on dies used for several coins in the database. Therefore, even though all the coins are stored as usual, only those targeted coins will account for the calculation.

In Figure 4.28, one can appreciate the impact of m on MAP and run time. There is roughly a diminution of MAP for all detectors up until $m = 20$. The behaviors of MAP for HOG and HOME tend to diverge; the former increases after $m = 75$ while the latter continues to diminish. HO2 balances those two tendencies and MAP tends towards a constant. Given the quite complicated overall variations of these curves, it seems reasonable to keep m low, ideally between 1 and 10.

The decrease for HOME is somewhat surprising; the intuitive assumption is that a larger number of retrieved values improves the retrieval, since no value is left behind. Yet, this is not what we observe. It is therefore possible that a too large m disrupts, on average, the performance by overflowing the geometric alignment with lots of spurious values. This would mean that the major issue comes from the descriptors, and imposing a larger cutoff value (see Algorithm 8) might be a solution. As for HOG, values greater than 200 would in theory provide no much better results than very low values. Figure 4.28b shows, as predicted, a linear evolution of the run time with m. A linear regression shows that the algorithm takes around 5 ms longer for each value added. Therefore there is little to no benefit in choosing high values of m.

Quantization q

Finally, results of Mean Average Precision and run time have been computed against the quantization factor q, and can be found in Figure 4.29 (MAP) and Table 4.9 (run time). Top results in MAP are obtained across the range from $q = 5$ to $q = 7$ for HOME and HO2, and decline starting from $q = 10$. The maximum MAP for HOG is obtained for around $q = 5$. Large values of q, authorizing less collisions, naturally alter the quality of the retrieval, and very low values, allowing more collisions, do not improve the results.

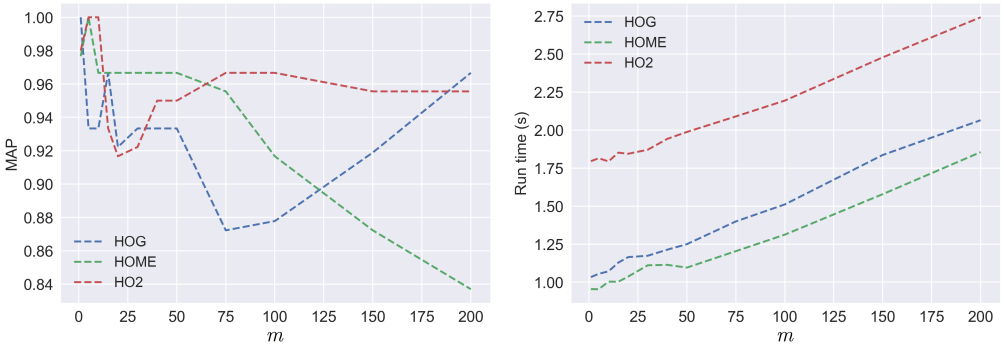
Furthermore, as previously noticed for artificial coins, the run time per coin explodes for very low values of q, exceeding 10 s for HOG and the HO2. The run time stabilizes after $q = 7$, which puts $q = 5$ on the sweet spot for a practical trade-off between run time and precision.

Overall, the differences between the descriptors seem to favor the HO2 in terms of MAP, since it remains the highest for any q. However, the trade-off that HOME provides is arguably the best because it leads to a high MAP and a low run time for $q = 5$.

4.6.2.6 Randomly rotated coins

In much the same way as was done previously in Section 4.6.1.4, the algorithm is used for rotated coins. The parameters are the same as for the upright coins, and the geometric parameters are set to ($\varepsilon_\rho = 50$, $\varepsilon_\theta = 15°$).

Dealing with rotated coins is a much more challenging task. Both methods for



(a) Mean Average Precision against m.

(b) Run time against m.

Figure 4.28: Impact of the maximum number of retrieved values per table m on the retrieval.

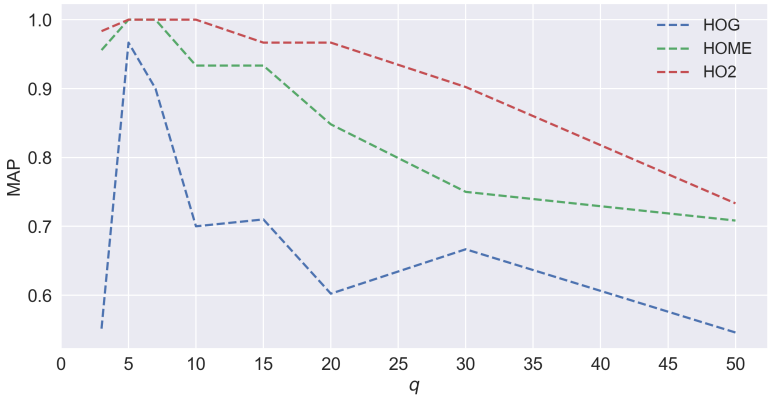


Figure 4.29: Mean Average precision against the quantization factor q for three descriptors.

q	3	5	7	10	15	20	30	50
HOG	10.4	0.99	0.56	0.52	0.38	0.42	0.45	0.38
HOME	3.46	1.00	0.64	0.49	0.46	0.45	0.44	0.46
HO2	13.2	1.85	1.21	0.87	0.83	0.88	0.85	0.79

Table 4.9: Run time (s) vs q per descriptor.

extracting oriented patches and for dealing with a polar geometry are also used as they are in this context. Each of these steps can have an impact on the quality of the retrieval since the first one fetches values in the tables, which here depends on the patch orientation, and the second one is a polar alignment of the patch position. The previous calculations on rotated coins (Section 4.6.1.4 and Figure 4.21) tend to favor local pooling, which is therefore employed for these experiments. Figure 4.8 shows the matches obtained between two real coins from the same die (the light green arrows).

Coins were randomly rotated around the middle of the frame (512,512). The protocol is then the same as usual and the Mean Average Precision is recorded. Since one obstacle of the retrieval is the correct alignment of the patches, a pre-processing step has been tested. Even if one cannot inherently pinpoint a coin center, the process of framing them within a 1024×1024 image implicitly does so. The underlying problem is that the engravings by themselves are not automatically centered this way. For example, in Figure 4.25 (g), one can certainly acknowledge that the query is very much similar to the third candidate (which does not originate from the same die, but absolutely could), but they were struck at different spots on the flan, which means the engravings are not positioned the same way; their implicit centers do not superimpose. This poses no problem whatsoever in the case of upright alignment because it precisely takes shift into account. Yet, in the case of the polar geometry, if the center is off then the retrieval is already mathematically impaired.

We propose one simple solution which creates a center based on this definition: the center of the coin is the outcome of the weighted sum of the positions in the energy map, which is written in Equation (4.25) where p designates a point in U.

$$p_{\text{center}} = \frac{\sum_p \mathcal{E}(p) \cdot p}{\sum_p \mathcal{E}(p)} \tag{4.25}$$

The results are gathered in Table 4.10. The Mean Average Precision is, on average, worse than for upright parametrization (with the same general parameters). As predicted by the study on artificial coins, HOME leads to slightly better results, but HOG performs on average slightly better than HOME when it comes to rotated coins. Both of them are outperformed by HO2. There is a slight gain in precision when the coins are centered beforehand and this gain is positive for every descriptors. The average increase is 0.04, while the average error is 0.09. Therefore, given the current precision, the gain is not sufficient to be significant and meaningful.[13] It also shows that the drop of MAP with respect to upright coins is not sufficiently explained by the (likely) positioning offset. However, the coherence between each descriptor (positive gain for each) seems to indicate nonetheless that centering is favorable, although very slightly so. The standard deviation, in general, is rather high. This implies that the outcomes are notably dependent on the rotation angle which has a twofold impact: one on the patches extraction and one on the alignment.

[13]The size of the dataset is at least partly responsible here. Having more data would enable us to avoid making wrong statistical conclusions.

(a) Original coin. (b) Coin after centering.

Figure 4.30: Centering of a coin. The frame is delineated by the black dotted line. The detected coin center is represented by a yellow dot and the frame center by a red dot. They are superimposed in (b).

	HOG	HOME	HO2
Original	0.63 (± 0.11)	0.52 (± 0.08)	0.71 (± 0.09)
Centered	0.66 (± 0.07)	0.58 (± 0.08)	0.73 (± 0.07)

Table 4.10: Mean Average Precision of the retrieval for rotated coins. These values are the average of 10 random sets of rotation angles. Standard deviations are shown in parentheses.

4.6.2.7 Conclusions

The current results on the real dataset are promising. The various scores obtained for upright and rotated coins, the behavior of the tables with respect to collisions and the in-depth mechanisms of matching single patches after geometric alignment all seem to agree with the previous tests on artificial coins.

The parameters were, for some, modified so as to foster better results on this dataset. Yet, those modifications are not major in that they do not differ radically from previously chosen parameters. The major changes and adaptations are the choice of the area score and the HO2 descriptors, which participated in the improvement of the results.

A quite good retrieval can be performed with the current parametrization. It is more than likely that those parameters are transferable to other, larger datasets. The transition from artificial to real coins went, overall, rather smoothly, which both indicates the relevancy of the carefully constructed artificial dataset as well as a satisfactory robustness of the algorithms. Rotated coins are, just like before, quite harder to retrieve correctly.

Coins struck with the same die are for the most part distinctly found together. The attributed score is even able to discriminate between these and mere lookalikes. However, all is not perfect and the algorithm can make several mistakes with various degrees of severity. Irrelevant coins are sometimes found in high ranks; correct coins are sometimes found at low ranks; coins with lots of features but which are not correct matches are sometimes paired with a very high score.

4.6.3 Real coin clustering

In this last part, we evaluate the clustering of coins presented in Section 4.5. For the next examples, we use the same parameters as those listed in Table 4.6, except that $n_u = 392$ since we use HO2. The coins are upright and the detector is still the adapted Harris. The quantization parameter is $q = 5$. K is the major variable which enables us to change the connectivity of the coin network.

The adjacency matrix has been computed for $K = 5$ and is presented in Figure 4.31. The coins have been ordered so that visually similar coins and coins from the same die are grouped together. As shown by some clearly distinguishable clusters of high scores on the diagonal, the algorithm has a nice proclivity to group coins which resemble each other, at least from a human perspective. However, no obvious distinction is brought out with respect to dies; just as it has been demonstrated in Section 4.6.2.4 and Figure 4.27, very similar coins may have a very high score even though they do not originate from the same die.

In Figure 4.32, various adjacency matrices and their corresponding graphs are displayed. The graph representation lets us observe how close the target coins are to each other. Note that, starting with $K = 1$, the connections between nodes of the same color are in general visibly stronger than any other. This tendency is clearly noticeable for the green label. Thus, using $K = 1$ already gives a fair idea of the major connections,

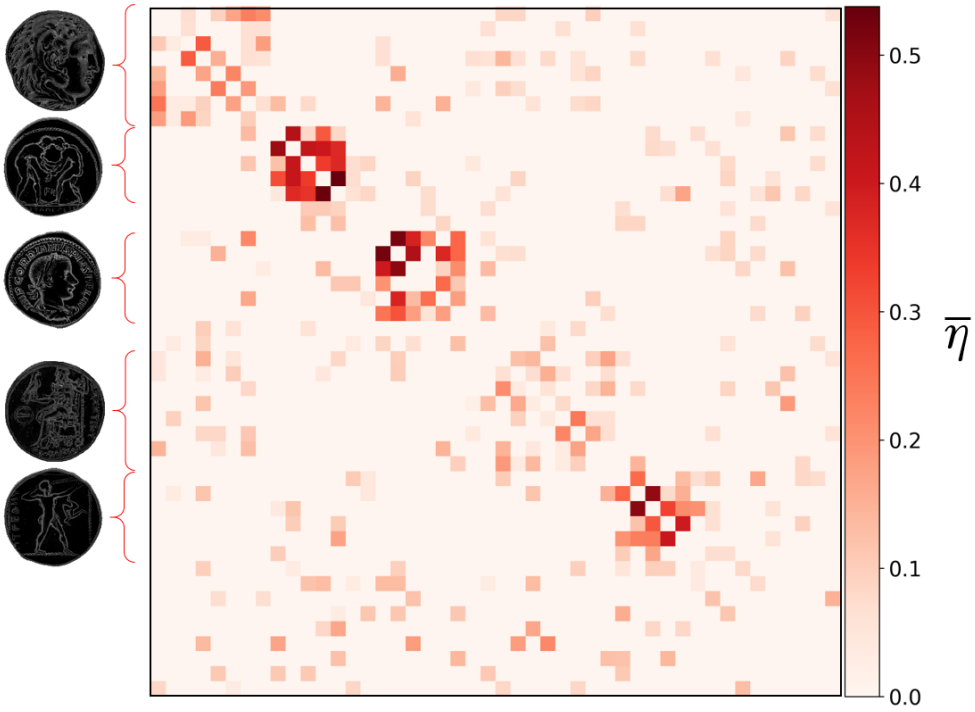


Figure 4.31: Adjacency matrix obtained with Algorithm 12 for $K = 5$. The color intensity varies with $\bar{\eta}$, which is the average score corresponding to Equation (4.23). The coins are arbitrarily ordered by appearance and by our prior knowledge of their label; this is for clarity purpose. Groups of coins emerge as being closely linked together. One exemplar of each group is displayed next to their location in the matrix.

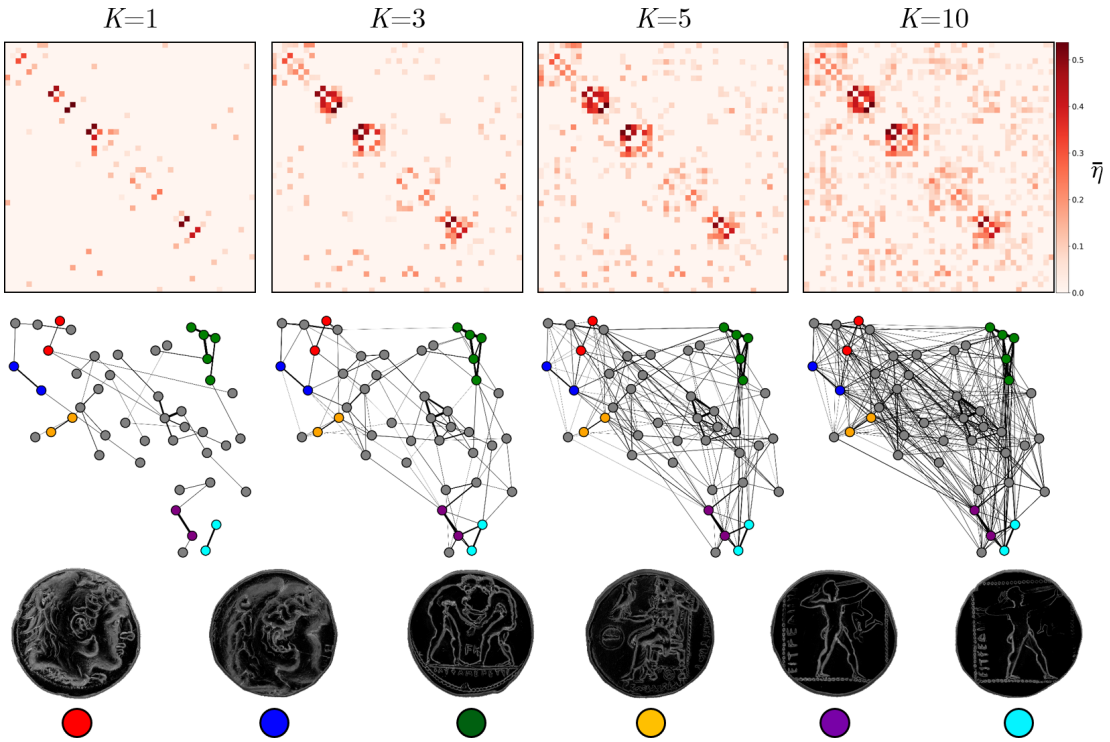


Figure 4.32: Adjacency matrices and corresponding graphs for various values of K. Colored nodes represent target coins from the same die. The edges are as thick as the associated scores are high. The proximity between nodes is relative to the strength of the edges.

even though it also lacks some obvious correct ones. Increasing K makes the network more connected, but at the cost of finding more and more irrelevant associations.

As a direct consequence of an aforementioned comment, there is also a cluster of grey nodes in the graphs, which stems from really high scores between those coins. They form a cluster of their own, which makes them indiscernible from correct clusters of target coins.

4.6.3.1 Clustering of coins

There is not a single way to extract meaning out of these data. Using graphs as a visualization tool like in Figure 4.32 is not a reasonable approach because large datasets would not be visually interpretable. However, from there, several options are available. In this section, we do not explore in details the possible solutions for clustering the graph. Nonetheless, some leads can be proposed.

It is possible to find graph clusters by using algorithms such as K-means or the Louvain algorithm [16]. The former requires to specify the number of clusters, which is likely to not be known. The latter is a type of community detection algorithm, which optimizes the local modularity (relative density of edges inside communities with respect to edges outside communities) in order to progressively cluster the graph; it is therefore unsupervised.

It has been observed that the Louvain clustering produces promising results by efficiently grouping coins from the same die in the same cluster. However, the drawback is that clustering assumes that all coins are to be labeled and that there are indeed real clusters to be highlighted. Yet, this is not true in our case; it would be if we were to consider a broader definition of clusters (types of coins, for example), but in our case there are a lot of isolated coins. Various propositions to solve this issue can be imagined. One could for instance apply a threshold to the adjacency matrix A. This has been found to be efficient, yet there is no absolute or calculable way to set the threshold. One could also simply look at each cluster to check the strength of the connections between its nodes. Moreover, the algorithm is quite greedy, with a $O(n_c \log n_c)$ complexity, which unfortunately defeats the purpose of LAPS.

A more practical approach is to perform a hierarchical clustering on the set of coins. Instead of simply finding clusters, one would like to organize the dataset such that similarity between coins or group of coins is found at various degrees. The resulting approach is presented in Figure 4.33. The method is the Paris algorithm [17], which is an agglomerative clustering algorithm that performs a greedy merge of nodes based on their similarity.

It has been found that using $K = 10$ yields better results than lower values, which is likely due to the high connectivity it enables. We visually clustered coins by type (by resemblance) so as to assess the relevancy of the similarities. The lower the connection point between two coins, the higher their proximity.

All target coins are correctly found closest to their matching counterpart (colored lines), which was a desirable result. However, the algorithm does not differentiate be-

tween target coins and similar coins (see types G and H), although one has to admit that the resemblance is high. This tendency stems directly from previous observations (retrieval, similarity matrix). Also, note that the algorithm has relevantly separated G from H in the hierarchy. This pattern is also noticeable for the A type. Still, there is no way to differentiate wrong matches (types and dies) from acceptable matches. This is likely because the method tries to fit every coin the best it can, which is not always the right action to perform (a more correct decision would be to discard the isolated coins altogether).

Overall, those simple results are promising. The general retrieval technique of LAPS can fit nicely into a clustering framework, although in these specific cases (Louvain and Paris), the gain in terms of retrieval time is lost. Perspectives for the future should include an investigation of more appropriate solutions in terms of complexity. The creation of a hierarchical clustering algorithm using the LAPS structure in its core would greatly improve run times and would represent an elegant solution for coin retrieval.

4.7 Conclusions

In this chapter, many themes about image recognition were tackled in order to establish a suitable pipeline for an efficient coin matching. The experiments were focused on the pairing of coins originating from the same die, which does make for a clear definition but remains nonetheless a non trivial exercise.

The first step is finding relevant locations to concentrate on so that meaningful data can be extracted. Through the paradigm of the energy map, we saw that the quantity and the quality of the interest points differ from those of natural images; there are much less locations, and therefore they have to be as robust and repeatable as possible. A total of five detectors were retained: the Gaussian, the multi-Gaussian, the Harris, the adapted Harris and the Laplacian of Gaussian. We deem that the adapted Harris is nonetheless the best choice for detection, as it detects a reasonable amount of points which were proved to be located in highly informative area.

The rationale for describing the coins is that variations inside of a coin are somewhat coarse if we compare it to natural images (and taking also into account the image size). Therefore, instead of small region descriptors, we choose to make use of the Histogram of Gradient paradigm across large scale patches. For rotation invariance, two techniques of patch extraction were implemented. The resulting descriptors are then ready for matching, with which measures and assessments can be made. The HOME provides an elegant solution to the task of description by combining the structure of a well-known, reliable descriptor, HOG, and the energy map model which gather meaningful data about the object.

As a matching strategy, the LACS system presented in Chapter 3 was upgraded so that it fits not only signatures but entire descriptors. This new matching system is called LAPS. The structural benefit of such matching is that it respects the locality in the comparison while allowing for gaps in the patches. Results of matching with this system are promising. It appears that the adapted version of the Harris detector stands out in

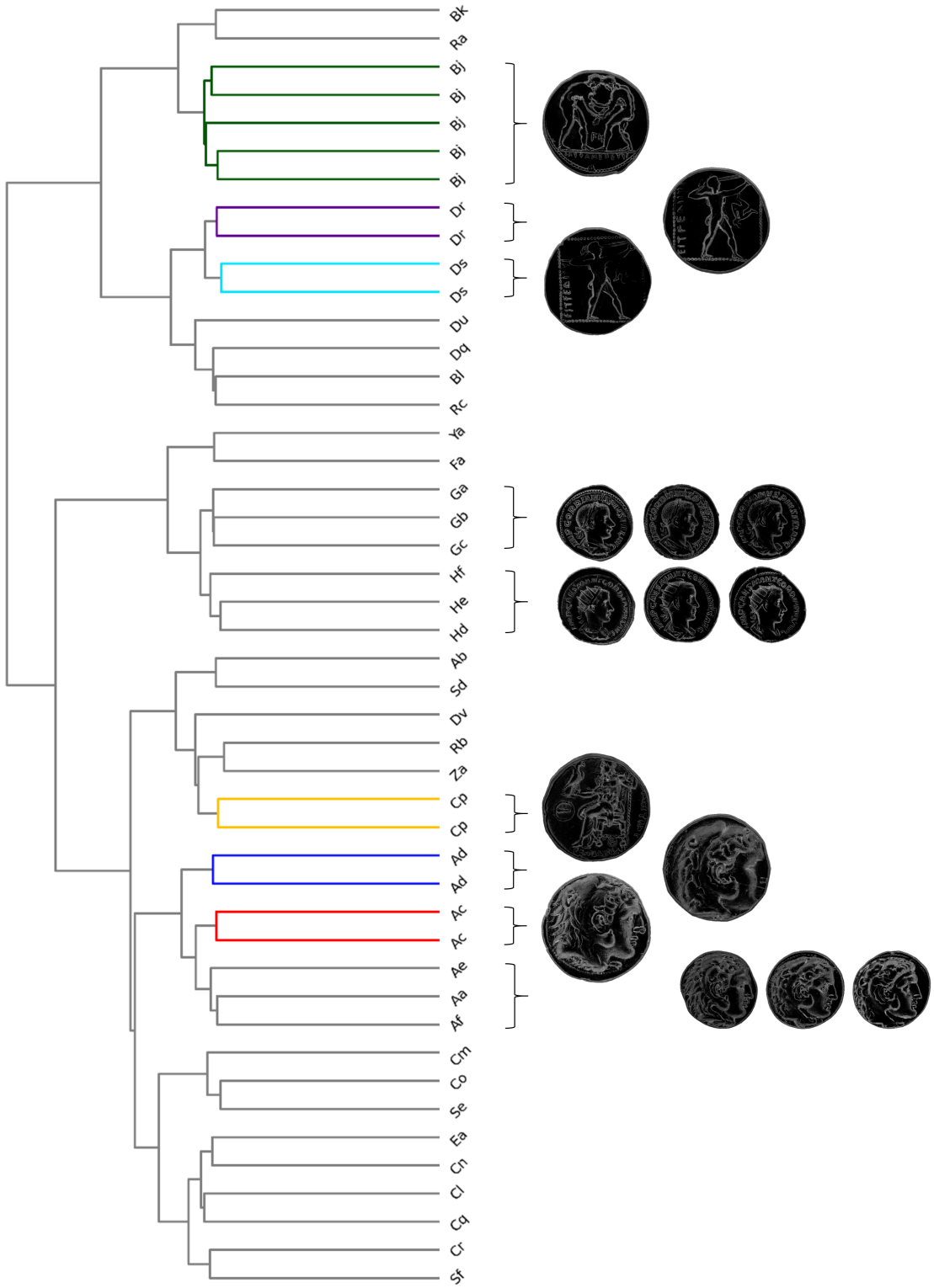


Figure 4.33: Dendrogram resulting from the hierarchical clustering. Colored lines represent target coins matched with their counterpart(s). Types and dies are respectively represented by a upper case type letter and a lower case die letter. Types were visually attributed. Various corresponding energy maps are displayed to the right.

terms of accuracy. The use of HOG or HOME barely make any difference in general, except when rotation is taken into account, in which case HOG performs better. General results of matching are excellent on the artificial dataset, both for the accuracy and the run time. Real coins were investigated as well and demonstrate the tranferability of the parameters and the framework in general. Satisfactory general results were obtained, although issues of inter-class variations (coins of the same type but not from the same die) became clearly visible.

Finally, clustering ideas were performed onto the real dataset. It appears to be a difficult task, but the use of LAPS embedded in an appropriate clustering strategy should be extremely efficient at categorizing ancient coins. Hierarchical clustering is maybe the best lead to conduct further analysis since it enables a human appreciation of the results in terms of progressive, structured similarities between objects.

Chapter 5

Conclusion

Quasi-flat objects are merely a slight relief of a flat surface obtained via striking a malleable piece with a matrix which have been usually handcrafted. The information they convey is not carried by any applied texture such as written glyphs or symbols but instead via the variations of their texture. As we have seen, ancient coins are perfect examples of such object: they are obtained by striking a metal flan with a die and thus feature a distinctive surface. We have observed that those objects can be apprehended as 2.5D objects, since the matrix itself can be modeled that way; to each point of the lattice U is associated one and only one height. This facilitates the mathematics regarding the description; in this configuration, the imprint on the surface and the light conditions are the only factors that intervene in the process of creating an image of the object. This further simplifies the way we understand it, but it also means that it is really dependent on the way the light is set, which can lead to very different images even from the same object. Illumination conditions can be controlled. It is partly for this purpose that the energy map has been introduced. However, in the context of recognition, it has been noted that other barriers are to be taken into account. Dirt, abrasions, wear and tear all play a role in the deterioration of the information of the object, which means that parts of it can be missing, altered or superimposed with noisy data. In the case of ancient coins, we have seen that erosion can alter the sharpness of the details by altering the degree of steepness of the relief variations. Additionally, the design of the imprint is conditioned by the matrix: a same matrix produces very similar items, although the strength and angle of the blow account for small differences. The matrix eventually wears out, which gradually deteriorates the quality of the quasi-flat object. Moreover, when changing the matrix is necessary, while the symbolism stays the same, non-rigid deformations between objects from different matrices will occur. These sometimes subtle variations in the relief make it difficult for a computer to correctly pair objects from the same type. The perspectives taken throughout this manuscript aim at overcoming these obstacles while producing a reasonable and efficient pipeline for retrieving such objects, at least in the context of object struck by the same matrix. The energy map is a basic model from which pieces of information have to be extracted. For this, two axes of research have been tested. Firstly, the energy map naturally translates the fluctuation

of the irradiance across the surface of the object, which appear to follow continuous one-dimensional lines which we interpret as contours. Secondly, the energy map is also a translation of the variations of the object's height map, which is interpretable in terms of textures. Both of these approaches underline promising results both in terms of handling the difficulty of analysing such objects and providing a fine accuracy of recognition.

The various contributions presented in this dissertation can be summed up following three axes. The first axis has to do with the representation of the object. To this end, we introduced the energy map, while expliciting some of its mathematical properties. The energy map is a synthetic, light and relevant model of the object, which takes into account the variations within its surface. In addition to the model, two coin datasets were shown; the first one is a set of 23 real coins, captured with various light positions so as to being able to compute the energy map. The second one is a set of 400 artificial coins which were designed computationally in the same fashion as artist engravers did in the antiquity.

The second axis refers to the choice of the decomposition of the object in terms of features which can then be compared. Two types of features were introduced: the contours and the textures. On the one hand, we explained two ways to extract the contours, both of which use the energy map model. Namely, we explicited an adaptation of the Canny algorithm in combination with the energy map. We showed how to connect pieces of contours together and how to choose the best of them, relying on their complexity. On the other hand, ways of extracting texture patches were described: the adapted Harris detector was introduced as well as the HOME which describes the object at the level of the surface gradient; both techniques use the energy map at their core.

The third and final axis concerns the comparison of objects. Two main algorithms for retrieving objects via their features were built: the LACS which uses contour information, and the LAPS which uses texture information. The LAPS is presented as an extension of the LACS, adaptable to numerous situations of higher dimensionality than the 1D signature model used with the LACS. Both algorithms are extremely fast, even in the presence of a large database. Additionally, they are essentially built for exact retrieval, which means that they highly discriminate small differences between features. For that reason, they are exceptionnally appropriate for matching objects from the same matrix. In addition to the retrieval algorithms, we suggested a way of clustering the objects that groups together very similar objects - the ones created with the same matrix assurely cluster together.

5.1 Energy map

The energy map [62] has been introduced as a simple yet efficient model for quasi-flat objects. It is computed from multiple images of the object taken with varying grazing light azimuths. The successive differences between such images actually enables the calculation of the variations of the image with respect to the light azimuth; and this quantity, obtained per pixel, corresponds precisely to the magnitude of the gradient of the object's relief. This representation therefore carries a significant amount of information

regarding the object – the very information which pinpoints relevant features of quasi-flat surfaces. The energy map not only is a basic model to represent the object, it also encompasses, by definition, various light conditions and therefore will always be invariant to any external illumination. The model is fast to compute, light, and it is thus interpretable in terms of the object's surface variations. It is easy to accompany this magnitude information with the corresponding orientation map just by considering the way pixels lighten as a function of the light azimuth. This additional information is useful to derive describing tools such as a feature detector – the adapted Harris – or a descriptor – the HOME. Experiments showed that this model is robust to the variations of the light slant – which does not vary in the calculation – and to very slight deviations from the center, which are interpretable in terms of an unequal azimuth distribution. The energy map is a repeatable model: provided the acquisition parameters stay the same, one is sure to obtain the exact same model.

It has to be noted that the energy map model still takes into account various assumptions. It presupposes that the object is a perfect diffuser. This hypothesis is quite strong, but it simplifies a lot the mathematics behind the model. This first approximation however yields fine results. The main concern is that coins, for instance, are not necessarily diffuse objects, and can be indeed quite specular. Specular reflections do not behave the same way and could therefore corrupt the result. Yet, a correct behavior of the model is observed even for specular coins. The grazing light is partly responsible. Besides, an analysis of the specular consequences have been studied, showing that a regular pattern emerges from the specularity and is akin to a second order gradient of the surface – it is however a slight effect. Using the diffuse approximation is therefore sufficient to explore the object information via the energy map model.

We also supposed that the object is not tilted. This has not been mentioned, but the tilt of the object – which could be in theory extracted from a statistical analysis – is a main factor in the resulting energy map. Tilting the object contracts images of the object, and the height map is wrongly oriented with respect to the z axis. In general this effect is small, but some ancient coins for instance are not really flat at all since they feature very wide surface variations. In these case, the coin angle should be corrected prior to any acquisition.

Finally, the energy map has to be built from a series of images which are acquired via a specific device presenting a camera on top, directed towards the object, and a set of n lights, usually 8. This device has not been built yet. The current version of the device is not ergonomic since it requires ambient light to be turned off, a smartphone camera with an arm to hold the smartphone and a portable LEDs torch light. The current version works well and all results are conform to the mathematical calculations. However, a small and portable device would be a great progress towards a more accessible and public version of the acquisition. We can at least list various ideas for future improvements:

- The LEDs can be positioned on a ring which could surround the camera (of a smartphone for instance). The whole device could be easily handled without the need of having a complete access to the object; it could be, for instance, acquired through glass in a museum). This would however change the light slant, which

may result in loss of contrast.

- Instead of positioning the object below the device, one could imagine the contrary, with the object put on a thin glass; and the camera below, pointing upwards.
- If the lights are sufficiently strong with respect to the ambient light, one can neglect the ambient illumination because it will be canceled out in the energy map calculation.

5.2 Recognition with contours

The energy map model by itself is not sufficient to compare objects. In Chapter 3, it has been noted that one can sum up the object as a collection of contours, which represent general shapes emerging from the relief imprinted by the matrix. Indeed, steep slopes, which represent large variations of the object's height map, visually appear as elongated parts from which contours can be inferred. There is unfortunately no general way to extract these contours. We chose two different ways originating from different paradigms. The first one is to create a version of the Canny filter adapted to the energy map; the filter thus acts directly at the level of the height map. This edge extraction technique requires the use of both the energy map and the orientation map. The final edge map results in a collection of localized thin sequences of edgels which we call segments. The second approach is to use an adaptive thresholding to extract elongated blobs from the energy map and define the segments as the contiguous sets of pixels of the skeleton of the blobs. Organizing the segments is critical because some can be too shorts and do not provide sufficient information. As noted, the density of the segment map will impact both the quality of the recognition and the computational run time – the more, the better, the slower. We define contours as a concatenation of possible consecutive segments, up to a maximum number of segments. The resulting contours are finally selected depending on their complexity, which enables only truly informative ones to be kept.

To recognize an object, the main idea is to use the collection of contours characterizing it. By comparing contours, we subdivide the deed into small retrieval tasks. First, we need to describe the contours: they are represented with the Triangle Centroid Distance (TCD) descriptor. This model captures the variations within the contours with a great stability; it is robust to Euclidean transformation as well as noise. The classical way of comparing two objects would then be to find matching contours in both objects and calculate a distance measurement. However, the retrieval pipeline we chose to build is quite different. We thereby introduced the Low-Complexity Arrays of Contour Signatures (LACS) [63], which is a system of nested associative arrays that enables a retrieval in a near constant complexity. The LACS system uses a geometrical alignment strategy to correctly select the candidates. The LACS especially deals with exact retrieval and is therefore appropriate for die recognition. We demonstrated the reliability of this retrieval system via well-known shape datasets as well as by using it to perform coin identification, in which, once again, the borders are exact. In the case of die recognition,

the retrieval results are different if we use the adapted Canny or the skeleton. As predicted, the density of the skeleton segment map provides a good accuracy which greatly improves as more TCD signatures are added, but the consequence is a increased run time. The Canny segment map is very sparse and not dense, which results in very short retrieval time; however, the accuracy is lower and barely impacted by the addition of more signatures. Using real objects, it appears that it remains preferable to use a denser segment map – the skeleton – if one is hoping to find some relevant matching contour parts. Overall, the retrieval is very disappointing in the case of real object; very few contours are retrieved and this is generally not sufficient to obtain correct results. In particular, this method of exact retrieval is poorly suited for non-rigid deformation within the relief of the object, which is why it is preferable to focus on recognizing matrices. Yet, even so, the deformation brought just by the impreciseness of contour localization seem sufficient to eliminate good potential candidates for the retrieval.

The developed system is extremely helpful for exact contour recognition. However, the algorithm cannot account for occlusions; parts of contours cannot be efficiently recognized. This is, in part, inherent to the nature of contour themselves: the information of a contour is always more than the sum of the information of its parts. In general, it should be possible to adapt the TCD to a localized version; it would still imply losing precision in the signatures. A lack of density leads to a similar problem, where too little data are gathered for recognition. The obvious issue underlying our algorithm is the trade-off between information and speed. To solve that issue, it is necessary to strive for a perfect contour selection: they have to be complex – at least in the sense of the definition provided in Chapter 3 – and very stable between objects. In theory, it would be better to find only a few contours with a high matching precision than lots of them and lose speed. At this level, this is only a speculation to say that such contours exist. The very first assumption with holds this chapter is the fact that there are indeed contours to be extracted and that these contours describe the object. Since no intrinsic definition has been found regarding contours, their modeling is arbitrary. Some examples of the real coins have for instance very thick edges; the elongated blobs are also wide, which makes pinpointing the segment inside it more of an educated guess than a reasonable task. The result is that none of those coins are correctly matched; the correct results are mostly obtained with steep slopes in the energy map where segments are easy to find.

5.3 Recognition with textures

The final attempt to retrieve objects from the same die was built around the idea of matching texture patches instead of mere contours. Textures are better defined; whichever the description, they always take into account the image variations within a certain area. Previous works on this matter have shown interesting results using a mix of Gabor wavelets (LIDRIC) or just simple localized gradient histograms (SIFT). To find the localization, feature point detection can be performed within the image, or a dense grid can be imposed on the 2D lattice. Contrarily to what has already been done, we prioritize the choice of interest points found within the object's surface, and

not in a mere image of it. We have shown that the most relevant and robust way to derive feature points is with the adapted Harris detector, which is a version of the Harris detector computed with the energy map and the orientation map. Using a local Gaussian maxima detector in the energy map proved to be very robust as well. However, the Harris detection leads to more informative regions around the keypoints. The description was performed with fixed sized patches and the Histogram of Gradient paradigm (HOG). A version adapted to the energy map, the Histogram of Oriented Map of Energy (HOME) was tested as well. It has been shown that it is possible and, most of the time, desirable to orient the patches because the object cannot be perfectly put straight with an absolute certainty. Yet, it is possible to get a reasonable centering, and the fixed size lattice adjusted to the borders of the object makes it invariant to scaling effect.

The retrieval was performed along the lines of the LACS model. We call this system Low-complexity Arrays of Patch Signature (LAPS) which is really a generalization of the LACS system to multiple tables in a descriptor. The spatiality of the descriptor is preserved as each histogram is used in a separate table of nested arrays. The LAPS remains extremely fast, provided the descriptors do not produce too many collisions. We have shown that the performances of the retrieval can be greatly improved with the use of the parameter m which limits the number of elements found per table, thereby implicitely limiting the collisions and the time complexity. This parameter appears to be critical, even though it makes a random decision on the choice of the retrieved values. Overall though, the algorithm seems to present a satisfying stability. It is remarkable that the algorithm performs a fast Hamming distance across all possible candidates. As we have done for LACS, the LAPS uses a geometric alignment which can be Cartesian (straight objects) or polar (rotated objects).

The resulting accuracy obtained on both the artificial and the real coins dataset is better than the one obtained with the contours on Chapter 3. Invariably, allowing the coins to be rotated impacts the retrieval quality. It has been also mentioned that the way the final score is computed is important to the final decision. It appears that incremental scores are sufficient in the case of artificial coins, but we found that calculating the final score via the percentage of area covered by the patches is more pragmatic and reasonable for real objects. Results on the real dataset proved to be convincing, although a confusion occurs when the randomized m values are changed. The algorithm has a fine tendency to discriminate between close objects and others by displaying significant score differences. Even though non-rigid deformations obviously impact the descriptor, the coarseness of each individual histogram (only eight bins) seems sufficient to account for this impreciseness. Also, occlusions are simply dealt with by two characteristics of this method: firstly, the local description makes the recognition sufficiently flexible, provided enough interest points are detected; and secondly, the locality of the matching (similarly to a L_1 norm) accounts for partly corrupted patches, which consider only part of it.

Finally, let us conclude this part with the attempt we made to regroup lookalike objects in the hope of finding clusters of coins struck with the same die. Clustering appears to be a whole domain of research to be examined, but we nevertheless tried to find some ways to get interesting results. The clustering can occur via the use of the

LAPS algorithm, from which a similarity matrix between each object can be calculated. There is not a unique way to cluster the set of objects. In order to demonstrate the feasability, we computed the Louvain algorithm as well as a hierarchial clustering to show that, at least in the latter case, it is possible to obtain a fine distinction between lookalikes and coins from the same die, for a relatively small dataset. This lead is a very promising one, because it would spare a considerable amount of time sorting through a set of objects and provide a first organization of the elements. However, in order to keep the process in the same vein as previous algorithms, it would be enthralling to adapt the LAPS retrieval system into a direct clustering algorithm, thereby avoiding to lose run time for the clustering.

5.4 Perspectives

The energy map is close to an ideal model for quasi-flat objects. The problem remains to extract its characteristics so that they can be exploited to perform local comparisons. Extracting contours is difficult and presupposes their existence and construction. Experiments tend to show that LACS is a good retrieval system using quasi-flat object contours provided the extraction of such contours yields very stable and complex contours, avoids noisy contours and ensures perfect matches with exact contour recognition. Extracting textures is a more stable solution but it invariably takes more retrieval time than with contours. There is little room for improvement regarding the nature of the descriptors. A particular focus should be put on improving the results for rotated coins, which is a more general case. Solutions may be found in rotation-invariant descriptors or better ways to orient the patches. One idea that has not been discussed so far is the combination of both contours and textures for the retrieval. It undoubtedly would improve the accuracy, but not by much, given the already high accuracies found with textures. However, it would take more time, which may not be worth. The clustering of quasi-flat object can also be bettered. Combining the LAPS principle with already existing hierachical clustering techniques may provide a new way to cluster them efficiently, providing one with the opportunity to easily sort a set of objects.

Quasi-flat objects analysis remains a challenging field of research. There are many tools that have been developed, and we just showed that many more can be possible to use. Using local description is indeed the way to go for these objects. We insist on the fact that the object should be considered, not just the image, which is projection of the light onto the surface and therefore cut out some information. However, it should be mentioned that, since the energy map is mathematically "at the level" of the image of the object, it may be possible to combine prior knowledge of the object (the energy map) with a mere photograph, although this has still to be worked out. Future developments should aim at building a suitable acquisition device; a simple, light and easy system can be harmoniously formulated. Such a device would enable lots of data to be captured regarding ancient coins or even other quasi-flat objects. The various pieces of algorithm presented throughout this dissertation are relevant to quasi-flat analysis, but we hope that some can be experimented on other domains. The elegant solution provided by

the LACS and the LAPS can be adapted to other fields of image recognition or even data retrieval in general. This will mostly prove to be efficient in the case of frameworks already using the Hamming similarity by considerably diminishing the retrieval time. Given the rapidity of the retrieval algorithm, one can also imagine using it in the context of real time tracking, especially if an exact pattern or symbol has to be recognized.

Bibliography

[1] Anwar, H., Anwar, S., Zambanini, S., and Porikli, F. CoinNet: Deep ancient Roman Republican coin classification via feature fusion and attention. *arXiv* (2019), 1–34.

See page 27.

[2] Anwar, H., Zambanini, S., and Kampel, M. A bag of visual words approach for symbols-based coarse-grained ancient coin classification. *ArXiv abs/1304.6192* (2013).

See pages 5, 25 et 26.

[3] Anwar, H., Zambanini, S., and Kampel, M. A Rotation-invariant Bag of Visual Words Model For Symbols Based Ancient Coin Classification. *International Conference on Image Processing(ICIP)* (2014), 5257–5261.

See page 25.

[4] Anwar, H., Zambanini, S., Kampel, M., and Vondrovec, K. Ancient coin classification using reverse motif recognition: Image-based classification of roman republican coins. *IEEE Signal Processing Magazine 32*, 4 (2015), 64–74.

See page 25.

[5] Arandjelović, O. Automatic Attribution of Ancient Roman Imperial Coins. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), 1728–1734.

See page 25.

[6] Asian, C., and Tari, S. An axis-based representation for recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV'05)* (2005), vol. 2, pp. 1339–1346.

See page 108.

[7] Aslan, S., Vascon, S., and Pelillo, M. Ancient Coin Classification Using Graph Transduction Games. *2018 IEEE International Conference on Metrology for Archaeology and Cultural Heritage, MetroArchaeo 2018 - Proceedings* (2018), 127–131.

See page 24.

[8] Aslan, S., Vascon, S., and Pelillo, M. Two sides of the same coin: Improved ancient coin classification using graph transduction games. *Pattern Recognition Letters 131* (12 2019), 158–165.

See pages 24 et 26.

[9] Attneave, F. Some informational aspects of visual perception. *Psychological Review 61*, 3 (May 1954), 183—193.

See pages 72 et 88.

[10] Badoud, N. Deciphering greek amphora stamps. *CHS Research Bulletin 5* (08 2017).

See page 18.

[11] Bai, X., Rao, C., and Wang, X. Shape vocabulary: A robust and efficient shape representation for shape matching. *IEEE Transactions on Image Processing 23* (september 2014).

See page 73.

[12] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. Speeded-up robust features (surf). *Computer Vision and Image Understanding 110*, 3 (2008), 346 – 359. Similarity Matching in Computer Vision and Multimedia.

See page 24.

[13] Bay, H., Tuytelaars, T., and Van Gool, L. Surf: Speeded up robust features. In *Computer Vision – ECCV 2006* (Berlin, Heidelberg, 2006), A. Leonardis, H. Bischof, and A. Pinz, Eds., Springer Berlin Heidelberg, pp. 404–417.

See page 125.

[14] Belongie, S., Malik, J., and Puzicha, J. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 4 (April 2002), 509–522.

See pages 72, 73, 105, 107, 108 et 110.

[15] Belongie, S. J., Malik, J., and Puzicha, J. Shape context: A new descriptor for shape matching and object recognition. In *NIPS* (2000), T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., MIT Press, pp. 831–837.

See page 24.

[16] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment 2008*, 10 (Oct 2008), P10008.

See page 187.

[17] Bonald, T., Charpentier, B., Galland, A., and Hollocou, A. Hierarchical graph clustering using node pair sampling. *CoRR abs/1806.01664* (2018).

See page 187.

[18] Bremananth, R., Balaji, B., Sankari, M., and Chitra, A. A new approach to coin recognition using neural pattern analysis. *Proceedings of INDICON 2005: An International Conference of IEEE India Council 2005* (2005), 366–370.

See page 26.

[19] Brenner, S., Zambanini, S., and Sablatnig, R. An investigation of optimal light source setups for photometric stereo reconstruction of historical coins. In *GCH 2018 - Eurographics Workshop on Graphics and Cultural Heritage, Vienna, Austria, November 12-15, 2018* (2018), R. Sablatnig and M. Wimmer, Eds., Eurographics Association, pp. 203–206.

See page 34.

[20] Brognara, C., Corsini, M., Dellepiane, M., and Giachetti, A. Edge detection on polynomial texture maps. vol. 8156, pp. 482–491. Image Analysis and Processing – ICIAP 2013".

See page 71.

[21] Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8* (1986), 679–698.

See page 70.

[22] Capece, N., Erra, U., and Ciliberto, A. V. Implementation of a Coin Recognition System for Mobile Devices with Deep Learning. *Proceedings - 12th International Conference on Signal Image Technology and Internet-Based Systems, SITIS 2016* (2016), 186–192.

See page 26.

[23] Chan, T., and Vese, L. Active contours without edges. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society 10 2* (2001), 266–77.

See page 71.

[24] Chang, L., Arias-Estrada, M., Hernández-Palancar, J., and Sucar, L. Partial shape matching and retrieval under occlusion and noise. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (CIARP)* (2014).

See page 72.

[25] Chen, Y.-S., and Hsu, W.-H. A modified fast parallel algorithm for thinning digital patterns. *Pattern Recognition Letters 7*, 2 (1988), 99–106.

See page 77.

[26] Coleman, T., and Moré, J. J. Estimation of sparse jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis 20* (1983), 187–209.

See page 80.

[27] Cooper, J., and Arandjelovic, O. Understanding ancient coin images. *arXiv* (2019).

See pages 5 et 27.

[28] Cooper, J., and Arandjelović, O. Learning to Describe: A New Approach to Computer Vision Based Ancient Coin Analysis. *Sci journal 2*, 1 (2020), 8.

See page 27.

[29] Cui, M., Femiani, J., Hu, J., Wonka, P., and Razdan, A. Curve matching for open 2D curves. *Pattern Recognition Letters 30*, 1 (2009), 1 – 10.

See pages 72, 88, 102, 105, 107, 108 et 110.

[30] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (2005), vol. 1, pp. 886–893 vol. 1.

See pages 126 et 136.

[31] Daliri, M. R., and Torre, V. Robust symbolic representation for shape recognition and retrieval. *Pattern Recognition 41* (May 2008), 1782–1798.

See pages 72 et 73.

[32] Davidsson, P. Characteristic Decision Trees By Controlling. *In Ninth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE-96)* (1996), 403–412.

See page 22.

[33] Donoser, M., Riemenschneider, H., and Bischof, H. Efficient partial shape matching of outer contours. In *Asian Conference on Computer Vision (ACCV)* (2009).

See pages 72 et 73.

[34] Fogel, I., and Sagi, D. Gabor filters as texture discriminator. *Biological Cybernetics 61* (2004), 103–113.

See page 125.

[35] Frankot, R. T., and Chellappa, R. A Method for Enforcing Integrability in Shape from Shading Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 4 (1988), 439–451.

See pages 5, 35 et 36.

[36] Friedman, J., Bentley, J., and Finkel, R. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions Mathematical Software 3* (09 1977), 209–226.

See page 126.

[37] Fukumi, M., Omatu, S., Takeda, F., and Kosaka, T. Rotation-Invariant Neural Pattern Recognition System with Application to Coin Recognition. *IEEE Transactions on Neural Networks 3*, 2 (1992), 272–279.

See page 26.

[38] Gauglitz, S., Höllerer, T., and Turk, M. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision 94* (2011), 335–360.

See page 131.

[39] Gionis, A., Indyk, P., and Motwani, R. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1999), VLDB '99, Morgan Kaufmann Publishers Inc., p. 518–529.

See page 73.

[40] Goodman, M. *Numismatic Photography*. Zyrus Press, 2008.

See pages 32, 33 et 168.

[41] Grauman, K., and Darrell, T. Fast contour matching using approximate earth mover's distance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2004), pp. 220–227.

See page 73.

[42] Harris, C. G., and Stephens, M. A combined corner and edge detector. In *Alvey Vision Conference* (1988).

See page 124.

[43] Hödlmoser, M., Zambanini, S., Schlapke, M., Kampel, M., and Schlapke, M. Evaluation of Historical Coin 3D Models. *Computer Applications and Quantitative Methods in Archeology - CAA'2010*, May 2014 (2010), 1–7.

See pages 5, 35 et 36.

[44] Horn, B., and Brooks, M. *Shape from Shading*, vol. 2. MIT Press, 01 1989.

See page 35.

[45] Hossfeld, M., Chu, W., Adameck, M., and Eich, M. Fast 3d-vision system to classify metallic coins by their embossed topography. *ELCVIA Electronic Letters on Computer Vision and Image Analysis 5*, 4 (2006), 47–63.

See page 35.

[46] Huber, R., Ramoser, H., Mayer, K., Penz, H., and Rubik, M. Classification of coins using an eigenspace approach. *Pattern Recognition Letters 26*, 1 (2005), 61–75.

See page 23.

[47] Huber-Mörk, R., Nölle, M., Rubik, M., Hödlmoser, M., Kampel, M., and Zambanini, S. Automatic Coin Classification and Identification. *Advances in Object Recognition Systems*, 2003 (2012).

See page 25.

[48] Huber-Mörk, R., Zambanini, S., Zaharieva, M., and Kampel, M. Identification of ancient coins based on fusion of shape and local features. *Machine Vision and Applications 22*, 6 (2011), 983–994.

See pages 25, 112 et 113.

[49] Jia, Q., Fan, X., Liu, Y., Li, H., Luo, Z., and Guo, H. Hierarchical projective invariant contexts for shape recognition. *Pattern Recognition 52* (2016), 358–374.

See page 72.

[50] Kampel, M., and Zaharieva, M. Recognizing ancient coins based on local features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5358 LNCS*, PART 1 (2008), 11–22.

See page 24.

[51] Kampel, M., and ZAMBANINI, S. Coin data acquisition for image recognition. *Proceedings of the 36th CAA Conference*, April (2008), 163–170.

See page 32.

[52] Kaothanthong, N., Chun, J., and Tokuyama, T. Distance interior ratio: A new shape signature for 2d shape retrieval. *Pattern Recognition Letters 78* (2016), 14–21.

See page 72.

[53] Kaplan, H., and Tenenbaum, J. Locality sensitive hashing for efficient similar polygon retrieval. *ArXiv* (2021).

See page 94.

[54] Kass, M., Witkin, A., and Terzopoulos, D. Snakes: Active contour models. *International Journal of Computer Vision 1*, 4 (1988), 321–331.

See page 71.

[55] Kavelar, A., Zambanini, S., Kampel, M., Vondrovec, K., and Siegl, K. the Ilac-Project: Supporting Ancient Coin Classification By Means of Image Analysis. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XL-5/W2*, September (2013), 373–378.

See page 25.

[56] Kim, J., and Pavlovic, V. Ancient coin recognition based on spatial coding. *Proceedings - International Conference on Pattern Recognition* (2014), 321–326.

See page 26.

[57] Kim, J., and Pavlovic, V. Improving Ancient Roman Coin Recognition with Alignment and Spatial Encoding. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8927* (2015), VI.

See page 26.

[58] Kim, J., and Pavlovic, V. Discovering characteristic landmarks on ancient coins using convolutional networks. *Proceedings - International Conference on Pattern Recognition 0* (2016), 1595–1600.

See page 27.

[59] Kun, Z., Xiao, M., and Xinguo, L. Shape matching based on multi-scale invariant features. *IEEE Access 7* (2019), 115637–115649.

See page 72.

[60] Lambert, J. H. *Photometria, Sive de Mensura Et Gradibus Luminis, Colorum Et Umbrae*. 1760.

See pages 22, 33 et 38.

[61] Lamdan, Y., and Wolfson, H. Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision* (1988), pp. 238–249.

See pages 73 et 94.

[62] Lardeux, F., Marchand, S., and Gomez-Krämer, P. Multi-Light Energy Map. In *EUROGRAPHICS Workshop on Graphics and Cultural Heritage (GCH)* (Vienna, Austria, Nov. 2018), pp. 199–202.

See pages 31, 68 et 192.

[63] Lardeux, F., Marchand, S., and Gomez-Krämer, P. Low-complexity arrays of contour signatures for exact shape retrieval. *Pattern Recognition 118* (2021), 108000.

See pages 88, 122 et 194.

[64] Lim, M.-J., Han, C.-H., Lee, S.-W., and Ko, Y.-H. Fast shape matching using statistical features of shape contexts. *IEICE Transactions on Information and Systems 94-D* (2011), 2056–2058.

See page 73.

[65] Lin, W. J., and Jhuo, S. S. Coin recognition based on texture classification on ring and fan areas of the coin image. *Conference Record - IEEE Instrumentation and Measurement Technology Conference 2016-July* (2016).

See page 25.

[66] Lowe, D. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60* (11 2004), 91–.

See pages 125 et 126.

[67] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (Nov. 2004), 91–110.

See page 24.

[68] Maaten, L. J. P. V. D., and Boon, P. J. COIN-O-MATIC : A fast system for reliable coin classification Muscle CIS benchmark. *Processings of the Muscle CIS Coin Competition Workshop* (2006).

See page 23.

[69] Maaten, L. V. D., and Postma, E. Towards automatic coin classification. *Proc. of the EVA-Vienna* (2006).

See pages 5, 23 et 24.

[70] Malzbender, T., Gelb, D., and Wolters, H. Polynomial texture maps. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01* (2001), 519–528.

See pages 34, 36 et 71.

[71] Marchand, S. IBISA : Making Image-Based Identification of Ancient Coins Robust to Lighting Conditions. *Proceedings of the 12th EUROGRAPHICS Workshop on Graphics and Cultural Heritage (GCH'14)* (2014), 13–16.

See pages 22, 34 et 126.

[72] Marchand, S., Desbarats, P., Vialard, A., Bechtel, F., Amara, B., Cicuttini, B., Bost, J.-P., Alain, B., Koray, K., and Beurivé, A. IBISA: Image-Based Identification / Search for Archaeology. *International Symposium on Virtual Reality*, October 2017 (2009), 57–60.

See page 22.

[73] Marr, D. Early processing of visual information. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences 275 942* (1976), 483–519.

See page 69.

[74] Mikolajczyk, K., and Schmid, C. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 27*, 10 (oct. 2005), 1615 –1630.

See page 24.

[75] Modi, S., and Bawa, S. Automated Coin Recognition System using ANN. *International Journal of Computer Applications 26*, 4 (2011), 13–18.

See page 26.

[76] Mudge, M., Voutaz, J. J.-p., Schroer, C., and Lum, M. Reflection Transformation Imaging and Virtual Representations of Coins from the Hospice of the Grand St . Bernard. *6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage* (2005), 29–39.

See pages 34 et 36.

[77] Nicodemus, F. E. Directional reflectance and emissivity of an opaque surface. *Applied Optics 4*, 7 (July 1965), 765–775.

See page 33.

[78] Nölle, M., Penz, H., Rubik, M., Mayer, K., Holländer, I., and Granec, R. Dagobert – A New Coin Recognition and Sorting System. *Proc. VIIth Digital Image Computing: Techniques and Applications* (2003), 329–338.

See page 23.

[79] Oliva, A., and Torralba, A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision 42* (05 2001), 145–175.

See page 125.

[80] Phong, B. T. Illumination for computer generated pictures. *Communications of the ACM 18*, 6 (June 1975), 311–317.

See page 51.

[81] Queau, Y., Durou, J.-d., and Aujol, J.-f. Normal Integration : A Survey. *Journal of Mathematical Imaging and Vision 60* (2017), 576–593.

See page 35.

[82] Raskar, R., han Tan, K., Feris, R., Yu, J., and Turk, M. Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging. *ACM Transactions on Graphics*.

See page 71.

[83] Reisert, M., Ronneberger, O., Burkhardt, H., Rösch, P., Harz, M., Schmitt, M., Peschke, K. D., Motzkus, H. W., Lankers, M., Hofer, S., and Others. An efficient gradient based registration technique for coin recognition. *Proceedings of the Muscle CIS Coin Competition Workshop 19* (2006), 31.

See page 24.

[84] Riemenschneider, H., Donoser, M., and Bischof, H. Using partial edge contour matches for efficient object category localization. In *European Conference on Computer Vision (ECCV)* (2010), pp. 29–42.

See page 72.

[85] Schlag, I., and Arandjelovic, O. Ancient Roman Coin Recognition in the Wild Using Deep Learning Based Recognition of Artistically Depicted Face Profiles. *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017 2018-Janua*, i (2018), 2898–2906.

See pages 27 et 33.

[86] Schmid, C., Mohr, R., and Bauckhage, C. Evaluation of Interest Point Detectors. *International Journal of Computer Vision 37*, 2 (2000), 151–172.

See pages 131 et 133.

[87] Sebastian, T., Klein, P., and Kimia, B. Recognition of shapes by editing shock graphs. *Transactions on the IEEE Pattern Analysis and Machine Intelligence 26* (June 2004), 550 – 571.

See pages 90 et 107.

[88] Shen, W., Du, C., Jiang, Y., Zeng, D., and Zhang, Z. Bag of shape features with a learned pooling function for shape recognition. *Pattern Recognition Letters 106* (02 2018).

See page 73.

[89] Shen, W., Jiang, Y., Gao, W., Zeng, D., and Wang, X. Shape recognition by bag of skeleton-associated contour parts. *Pattern Recognition Letters 83* (2016), 321–329.

See page 72.

[90] Shu, X., and Wu, X. A novel contour descriptor for 2D shape matching and its application to image retrieval. *Image and Vision Computing 29* (March 2011), 286–294.

See page 72.

[91] Sobel, I., and Feldman, G. An isotropic 3×3 image gradient operator. *Pattern Classification and Scene Analysis* (1968), 271–272.

See page 70.

[92] Song, R., Zhang, Z., and Liu, H. Edge connection based canny edge detection algorithm. *Journal of Information Hiding and Multimedia Signal Processing 8* (11 2017), 1228–1236.

See page 70.

[93] Stern, Carl M. Pester, A. L., and Shah, G. Low power coin discrimination apparatus. *Google Patents CA1336782C* (1986).

See page 22.

[94] Sun, K. B., and Super, B. J. Classification of contour shapes using class segment sets. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2* (2005), 727–733.

See page 73.

[95] The Moving Picture Experts Group. MPEG-7 dataset.

See page 108.

[96] Tompa, V., Dragomir, M., Hurgoiu, D., and Neamţu, C. Image Processing Used For The Recognition and Classification of Coin-type Ancient Artifacts.

See page 33.

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

See page 71.

[98] Waechter, M., Everhart, D., Gerrity, D., Phillips, A. C., and Neubarth, S. K. Coin discrimination apparatus and method. *Google Patents CA2426293A1* (1997).

See page 22.

[99] Wang, A. An industrial strength audio search algorithm. In *International Conference on Music Information Retrieval (ISMIR)* (January 2003).

See pages 88, 94 et 102.

[100] Wang, J., Bai, X., You, X., Liu, W., and Latecki, L. J. Shape matching and classification using height functions. *Pattern Recognition Letters 33* (2012), 134–143.

See page 72.

[101] Wang, X., Feng, B., Bai, X., Liu, W., and Latecki, L. J. Bag of contour fragments for robust shape classification. *Pattern Recognition 47* (2014), 2116–2125.

See page 73.

[102] Wei, L., Keogh, E. J., Xi, X., and Yoder, M. Efficiently finding unusual shapes in large image databases. *Data Mining and Knowledge Discovery 17*, 3 (2008), 343–376.

See page 94.

[103] Wolfson, H. J., and Rigoutsos, I. Geometric hashing: an overview. *IEEE Computational Science and Engineering 4*, 4 (1997), 10–21.

See page 88.

[104] Woodham, R. J. Photometric Method For Determining Surface Orientation From Multiple Images. *Optical Engineering 19*, 1 (1980).

See page 33.

[105] Yang, C., Wei, H., and Yu, Q. A novel method for 2D nonrigid partial shape matching. *Neurocomputing 275* (January 2018), 1160–1176.

See pages 72, 73, 88, 90, 96, 105, 107, 108 et 110.

[106] Yang, J., Wang, H., Yuan, J., Li, Y., and Liu, J. Invariant multi-scale descriptor for shape representation, matching and retrieval. *Computer Vision and Image Understanding 145* (2016), 43–58.

See page 72.

[107] Zaharieva, M., Huber, R., Nölle, M., and Kampel, M. On ancient coin classification. *8th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST'07)*, April 2014 (2007), 55–62.

See pages 23 et 24.

[108] Zaharieva, M., Kampel, M., and Zambanini, S. Image based recognition of ancient coins. *Lecture Notes in Computer Science (including subseries Lecture*

Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4673 LNCS (2007), 547–554.

See page 23.

[109] Zambanini, S., and Kampel, M. Robust Automatic Segmentation of Ancient Coins. *4th International Conference on Computer Vision Theory and Applications* (2009), 273–276.

See pages 47 et 112.

[110] Zambanini, S., and Kampel, M. Automatic Coin Classification by Image Matching. *VAST11: The 12th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*, January (2011), 65–72.

See page 24.

[111] Zambanini, S., and Kampel, M. A Local Image Descriptor Robust to Illumination Changes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7944 LNCS*, November 2015 (2013).

See pages 24, 33 et 125.

[112] Zambanini, S., Kavelar, A., and Kampel, M. Improving ancient Roman coin classification by fusing exemplar-based classification and legend recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8158 LNCS* (2013), 149–158.

See page 25.

[113] Zambanini, S., Kavelar, A., and Kampel, M. Classifying ancient coins by local feature matching and pairwise geometric consistency evaluation. *Proceedings - International Conference on Pattern Recognition*, August (2014), 3032–3037.

See pages 25, 26 et 150.

[114] Zambanini, S., M Kampel, and M Schlapke. On the Use of Computer Vision for Numismatic Research. *9th International Symposium on Virtual Reality, Archaeology and Cultural Heritage 45*, 7 (2008), 17–24.

See page 32.

[115] Zambanini, S., Schlapke, M., Hödlmoser, M., and Kampel, M. 3D acquisition of historical coins and its application area in numismatics. *Analysis 7531*, February (2010), 753108–753108–8.

See pages 35 et 36.

[116] Zeng, J., Liu, M., Fu, X., Gu, R., and Leng, L. Curvature bag of words model for shape recognition. *IEEE Access 7* (2019), 57163–57171.

See page 73.

[117] Zhang, D., and Lu, G. A comparative study of Fourier descriptors for shape representation and retrieval. In *Asian Conference on Computer Vision (ACCV)* (2002).

See page 88.

[118] Zhang, Z., Pati, D., and Srivastava, A. Bayesian clustering of shapes of curves. *Journal of Statistical Planning and Inference 166* (2015), 171–186.

See page 73.

[119] Zheng, Y., Meng, F., Liu, J., Guo, B., Song, Y., Zhang, X., and Wang, L. Fourier transform to group feature on generated coarser contours for fast 2d shape matching. *IEEE Access 8* (2020), 90141–90152.

See page 72.

Appendix A

Real coin dataset details

Coin references	**Details**
6 roman imperial coins of Gordian III	3 denarii and 3 antoninianii
8 greek coins of Alexander III the Great and his successors	1 tetradrachm of Seleukos I Nikator in the name of Alexander, 1 tetradrachm of Alexander (lifetime issue), 1 tetradrachm of Antigonos II Gonatas in the name of Alexander (same obverse die as the next coin), 1 tetradrachm of Antigonos (same obverse die as the previous coin), 2 tetradrachms of Demetrios I Poliorketes (from the same pair of dies) [*cf. coin number 28 (die pair XXIII/66) in Edward T. Newell, The Coinages of Demetrius Poliorcetes, Oxford University Press, London, 1927*], 2 tetradrachms of Seleukos (the second one formerly in the collection of the Metropolitan Museum of Art of New York City)
7 staters of Aspendos	5 staters from the same pair of dies (with a die break at various stages), 1 stater of different dies, 1 stater of different dies, but the same engraver as the first 5 staters [*cf. Olivier Masson, Onomastica Graeca Selecta, Tome III, p. 86, 2000*].
1 roman republic denarius, with incuse reverse	Strike error where the obverse of a previous coin was not removed from the reverse die and left a negative print on the next coin.
1 electrotype copy of the famous Demareteion decadrachm of Syracuse, the original being in the collection of the British Museum	https://www.britishmuseum.org/collection/object/C_1841-0726-287.

Table A.1: Details of the 23 coins of the dataset.

Appendix B

Feature point repeatability

The repeatability is a key measurement for feature points because it is necessary to enquire about the robustness to various perturbations. We analyze the repeatability for four transformations: scaling, rotation, noise and erosion (occlusion).

B.1 Scaling

The energy map can be formatted so that it respects a standardized dimension. For instance, one can decide that the frame is the boundary box of the mask of the energy map. In any case, two energy maps may not be scaled the same way, depending on their shape. We analyze here the influence of the scaling on the repeatability of the detectors. The range of parameters $\{1.0, 1.1, ..., 2.0\}$ is broad in order to distinguish general tendencies, but the real range of interest is assumed to be much smaller. Interest points of the scaled version of the energy map are remapped according to the scaling factor so that they match the original ones.

Results are displayed in Figure B.1. Most detectors show a satisfactory robustness. The centroid keypoints are the most robust, followed by the Gaussian maxima, multi-Gaussian maxima and Harris keypoints. Both DoG keypoints and entropy maxima seem to be sensitive to scaling. The entropy maxima need a scale parameter for the size of the entropy filter, which of course impacts the detection when the scale changes. Note that this is not the case for the Gaussian maxima, even though they do take into account a filter size. This may be explained by the original purpose of the Gaussian filter, which mainly reduces noise across the energy map. The area under curve scores are presented in Table B.1. It is clear with this representation that centroids of blobs, Gaussian maxima and Harris keypoints feature the best repeatabilities and that entropy is by far the least robust.

Method	Gaussian	Multi-Gaussian	Entropy	Harris	Adapted Harris	Centroids	LOG	DOG	DOH
Accuracy (%)	93.7	74.2	59.6	91.3	84.3	97.3	88.8	76.8	82.7

Table B.1: Area under curve of repeatability for scaling.

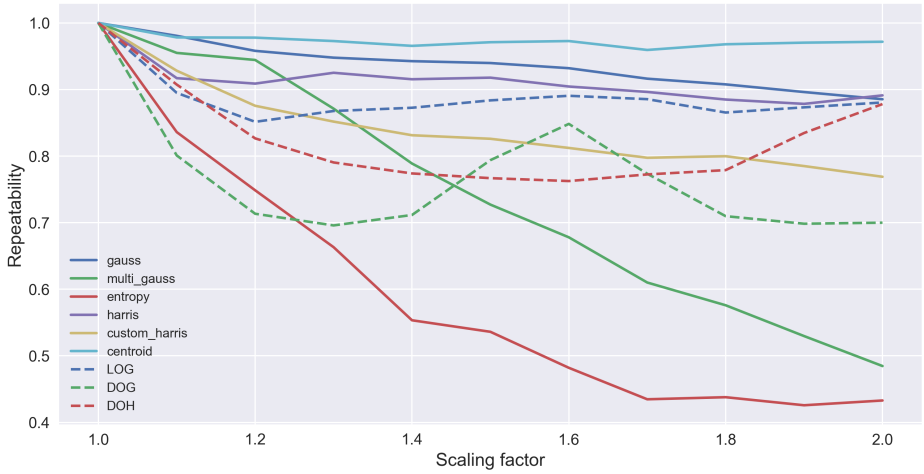


Figure B.1: Repeatability for scaling.

Method	Gaussian	Multi-Gaussian	Entropy	Harris	Adapted Harris	Centroids	LOG	DOG	DOH
Accuracy (%)	89.3	85.9	80.2	85.7	86.7	93.3	89.3	78.7	73.7

Table B.2: Area under curve of repeatability for rotation.

B.2 Rotation

Since there is no way to correctly orient a coin, it is logical that we take into account the possible rotation of the object. The detected interest points must therefore stay robust to any angle of rotation. We take 16 angles equally distributed across the $[0°, 360°)$ domain, and apply the same repeatability measurements by mapping those points onto the original, unrotated object.

Results per angle are shown in Figure B.2 and the AUC scores (Equation (4.9)) are listed in Table B.2. All detectors are impacted by rotation, especially for the angles 45 (mod 90). This indeed corresponds to the orientation for which the largest amount of interpolation is involved so that the image is fully transformed. Repeatabilities appear nonetheless to stay rather high ($r \geq 65\%$). The blob centroids are barely affected and gather the highest overall repeatability (AUC). LoG keypoints and Gaussian maxima as well as both versions of the Harris detector are rather robust, featuring more than a 80% AUC. DoG, DoH and entropy keypoints present the lowest AUC and are especially largely impacted by 45 (mod 90) angles, falling below 75%.

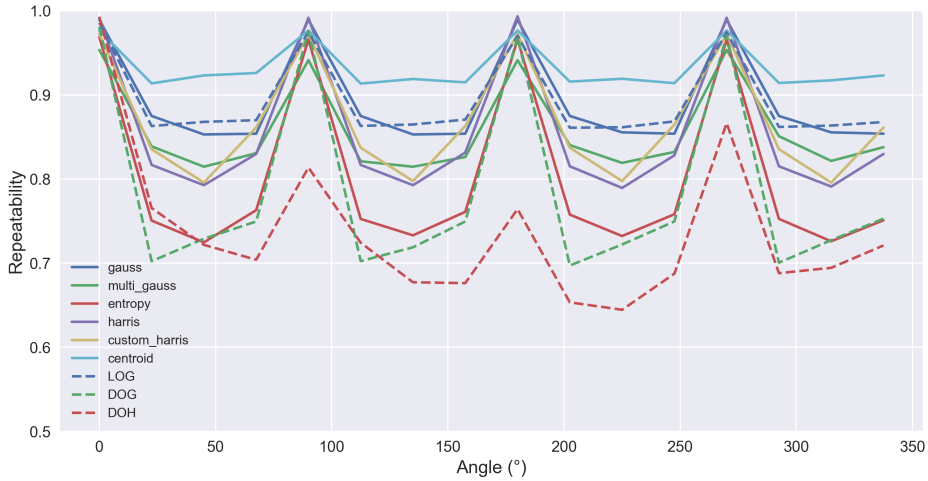


Figure B.2: Repeatability for rotation.

B.3 Noise

To assess resistance to noise, a Gaussian random noise is applied to the images of each coin. The varying parameter is the noise variance, which was empirically chosen to be between 0 and 0.02. This choice comes mostly from a visual examination of the impact of noise. In Figure B.3, an example of noisy image is shown. Figure B.3a features the original image and in Figure B.3b a noise of variance 0.02 has been applied. The perturbation induced by the noise may appear shallow, but it quite noticeably affects the way some detectors work and corrupts the repeatability. For instance, observe that, in Figure B.3b, the noise may lead to random interest points within flat areas while at the same time diminish the precision of detection within highly discriminative edges and corners (the crown, the eyes, the ear,...) and may even prevent the detection in textured areas such as the within the hair.

Results are presented in Figure B.4 and Table B.3. Overall, there is mostly a large and unmistakable decrease in repeatability. The Difference of Hessian displays the worst resilience to noise. Note that both Harris are definitely affected, but the standard formulation performs significantly worse than the adapted version. This may come from the nature of the inspected corners. The latter works directly on the energy map, which exhibits rather large details considering the image size; the former focuses on the derivative components of the energy map, which are are spatially finer. The centroids of blobs feature a reasonable decrease in repeatability due to noise, which was expect since noise can alter the geometry of the blobs. It is clear that any detector that includes some smoothing will easily bypass the impact of noise. In our case, the Gaussian and multi-Gaussian maxima appears rather unaffected, managing to keep an AUC of around 94%. Based on this, it is likely that a simple Gaussian filtering prior to any detection can

(a) Original image. (b) Noisy image ($\text{std}^2 = 0.02$).

Figure B.3: Visual impact of noise on the quality of the coin image.

Method	Gaussian	Multi-Gaussian	Entropy	Harris	Adapted Harris	Centroids	LOG	DOG	DOH
Accuracy (%)	94.1	93.6	41.1	49.1	66.4	71.9	58.7	79.3	26.4

Table B.3: Area under curve of repeatability for noise.

generally improve the repeatability.

B.4 Erosion

Erosion is an important component of coin recognition since it is more than likely that one has to deal with wear and tear. The perturbation is simulated much in the same way we created erosion for the artificial coins. A Gaussian filter of standard deviation 32 px is applied within a patch of size 128×128. The impact of the filter is decreasing from the center of the patch to the borders, so that there is less and less influence of the filter on the targeted area as we progress towards the edges. That way, the connection between the modified patch and the rest of the image is seamless. The chosen decreasing function is a Gaussian of standard deviation 64 px. An example with three erosion locations is shown in Figure B.5. The impact is soft, but physically credible. The parameter to assess the repeatability will be the total eroded area. Each new erosion is considered to occur within the disk inscribed in the patch. There may be overlap, but those will not be taken into account in the total area covered.

It is important to note that obtaining a 100% repeatability for any quantity of erosion does not make sense; this score will fall down eventually. There is indeed a decreasing tendency for the repeatability of all descriptors, which can be observed in Figure B.6. In contrast to every other perturbations, erosion impacts significantly the repeatability of the centroids of blobs. This is not surprising considering the binary nature of blobs; erosion changes their shape and therefore centroids are easily disturbed. The most

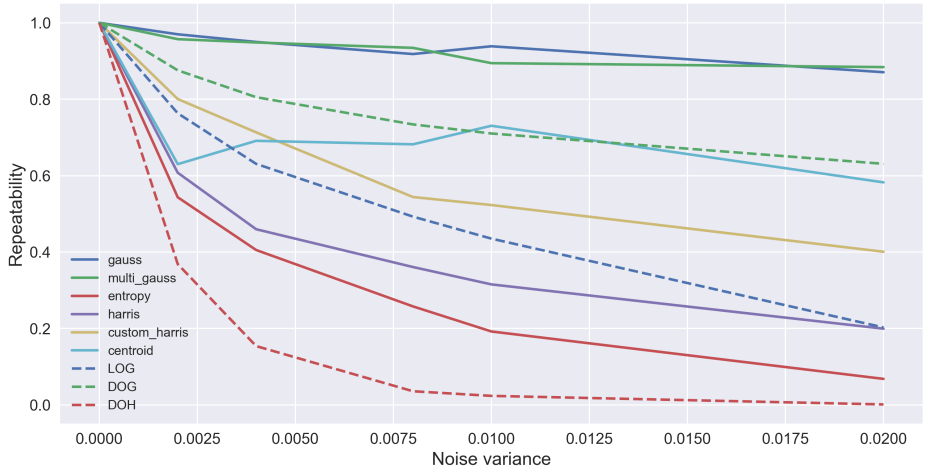


Figure B.4: Repeatability for noise.

(a) Single image. (b) Energy map.

Figure B.5: Example of a coin with three eroded areas with a patch of 128×128.

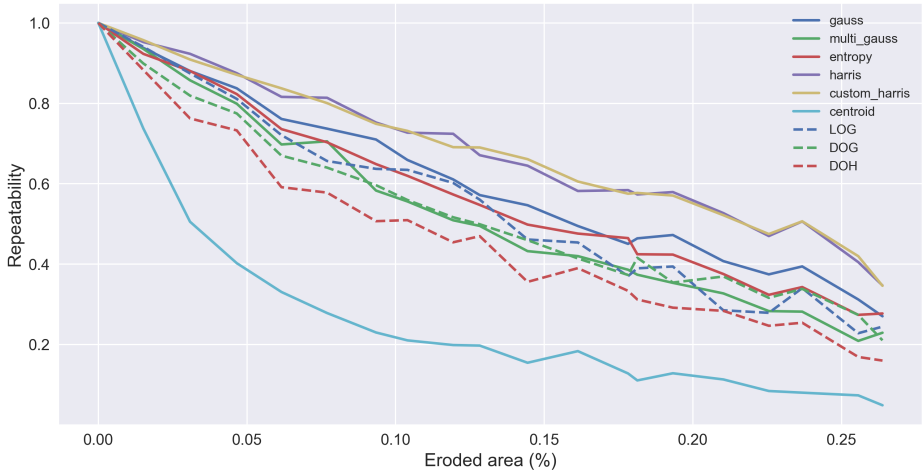


Figure B.6: Repeatability for erosion.

Method	Gaussian	Multi-Gaussian	Entropy	Harris	Adapted Harris	Centroids	LOG	DOG	DOH
Accuracy (%)	57.3	49.7	54.4	65.7	65.8	22.1	52.0	50.0	43.6

Table B.4: Area under curve of repeatability for erosion.

robust detectors, according to Figure B.6 and Table B.4 are Harris and the adapted Harris, reaching around 66%, 9 points above the third best detector. Being the only corner detectors, it seems to indicate that this method of detection is preferable when dealing with erosion-like perturbations.

B.5 Conclusion

Regarding the general quality of the detectors, the centroids of blobs can be ruled out simply because of their volatility in presence of erosion, which happens frequently. The entropy maxima as well as the Determinant of Hessian detector are very sensitive to noise. The involvement of noise is non negligible, since it encompasses various causes such as oxidation, dirt and camera noise. It is therefore preferable to eliminate these detectors. The Gaussian detectors are, in general, extremely robust, especially to noise. Their general principle is simple and their computation is straightforward. Both Harris detectors have proven to be quite robust as well. Their relevancy is accentuated by their resilience to erosion which outperforms the other techniques. However, they remain sensitive to noise, which may alter future recognition results. Both blob detectors Laplacian of Gaussian and Difference of Gaussian tend to behave equally, which is not surprising given their very close mathematical formulation. They are overall also rather robust to all perturbations.

Résumé : Les objets quasi-plans sont obtenus à partir d'une matrice qui définit des caractéristiques spécifiques observables dans leur gravure. Les exemples de ce type d'objets incluent les timbres à sec, les timbres amphoriques ou encore les pièces de monnaies anciennes. Les objets quasi-plans sont par conséquent des formes très plates sur lesquelles un relief caractéristique est inscrit. Reconnaître de tels objets est une tâche compliquée car de nombreux verrous entrent en jeu. Le relief d'objets quasi-plans est sujet à des déformations non rigides et les conditions lumineuses influencent la façon dont ils sont perçus. De plus, ces objets ont pu subir diverses détériorations menant à l'occlusion de certaines parties de leur relief. Dans cette dissertation, nous adressons le problème de la reconnaissance d'objets quasi-plans. Cette thèse est articulée autour de trois grands axes. Le premier axe vise à créer un modèle adapté pour représenter l'objet en exposant ces caractéristiques et en prenant en compte les divers verrous précités. Dans ce but, le concept de carte d'énergie lumineuse est introduit. Les deuxième et troisième axes introduisent des stratégies de reconnaissance. D'un côté, nous proposons l'utilisation de contours de l'objet en tant que données caractéristiques. Ceux-ci sont représentés via un modèle de signature à partir duquel sont calculés des descripteurs. Afin de stocker, retrouver et comparer ces descripteurs, une structure de donnée basée sur des tableaux associatifs, les LACS, est présentée ; elle permet une reconnaissance rapide de contours similaires. D'un autre côté, l'utilisation de textures comme descripteurs de l'objet est envisagée. Dans cette partie, l'étude est centrée sur l'emploi de régions 2D et leur description comme moyen de reconnaissance. Un angle similaire est adopté pour stocker et retrouver l'information ; une structure de donnée proche de celle précédemment décrite, mais plus complexe, est introduite.
Mots clés : objets quasi-plans, géométrie 2.5D, pièces de monnaie anciennes, requête rapide, reconnaissance de contour, reconnaissance de texture

Abstract: Quasi-flat objects are obtained from a matrix which defines specific features observable in their engraving. Some examples of these are dry stamps, amphora stamps or ancient coins. Quasi-flat objects are subsequently understood as very flat shapes onto which a characteristic relief is inscribed. Recognizing such objects is not an easy feat as many barriers come into play. The relief of quasi-flat objects is prone to non-rigid deformations and the illumination conditions influence the perception of the object's relief. Furthermore, these items may have undergone various deteriorations, leading to the occlusion of some parts of their relief. In this thesis, we tackle the issue of recognizing quasi-flat objects. This work is articulated around three major axes. The first one aims at creating a model to represent the objects both by highlighting their main characteristics and taking into account the aforementioned barriers. To this aim, the concept of multi-light energy map is introduced. The second and third axes introduce strategies for the recognition. On the one hand, we propose the use of contours as main features. Contours are described via a signature model from which specific descriptors are calculated. In order to store, retrieve and match those features, a data structure based on associative arrays, the LACS system, is introduced, which enables a fast retrieval of similar contours. On the other hand, the use of textures is investigated. The scope here is centered on the use of specific 2D regions and their description in order to perform the recognition. A similar angle is taken to store and retrieve the information as a similar, yet a more complex data structure is introduced.
Keywords: quasi-flat objects, 2.5 geometry, ancient coins, modelization, fast retrieval, contour recognition, texture recognition

Laboratoire Informatique, Image, Interaction
Faculté des Sciences & Technologie
Avenue Michel Crépeau

17042 LA ROCHELLE CEDEX 1