Low-complexity arrays of patch signature for efficient ancient coin retrieval

Florian Lardeux, Petra Gomez-Krämer, Sylvain Marchand

Low-complexity Arrays of Patch Signature for Efficient Ancient Coin Retrieval

Florian Lardeux[1], Petra Gomez-Krämer[1*], Sylvain Marchand[1]

[1]L3i Laboratory, La Rochelle University, Avenue Michel Crépeau, La Rochelle, 17042, France.

*Corresponding author(s). E-mail(s): petra.gomez@univ-lr.fr;
Contributing authors: sylvain.marchand@univ-lr.fr;

Abstract

We present a new recognition framework for ancient coins struck from the same die. It is called Low-complexity Arrays of Patch Signatures (LAPS). To overcome the problem of illumination conditions we use multi-light energy maps which are a light-independent, 2.5D representation of the coin. The coin recognition is based on a local texture analysis of the energy maps. Descriptors of patches, tailored to coin images via the properties provided by the energy map, are matched against a database using a system of associative arrays. The system of associative arrays used for the matching is a generalization of the Low-complexity Arrays of Contour Signatures (LACS). Hence, the matching is very efficient and nearly at constant time. Due to the lack of available data, we present two new data sets of artificial and real ancient coins respectively. Theoretical insights for the framework are discussed and various experiments demonstrate the promising efficiency of our method.

Keywords: Ancient coin recognition, Texture descriptors, Associative arrays, Ancient coin data set

1 Introduction

Ancient coins are quasi-flat objects, namely flat shapes with specific reliefs on both sides. These characteristic features are obtained via the fabrication of these objects: they are struck with a die, which imprints a distinctive relief on it. They are best described by this relief, therefore any relevant information to identify or classify them has to be looked for in it. 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 new die are different from coins struck by the former die, even though they have the same symbols and monetary value. Hence, there are several aspects of the fabrication which make each object really unique, although certain classes can be determined.

Ancient coins provide particular characteristics which make their recognition challenging:

- Illumination conditions: These objects do not feature specific textures. So what is perceived is mainly determined by the relief and the light direction.
- Occlusions: They may originate from various reasons. The object can be struck off-center, parts of the object are missing or it might have undergone oxidation.
- Altered quality: Due to the way the object is struck – for instance, hammering a flan with two unparalleled dies – and due to erosion, wear and tear, the object may have lost its initial details, depending on the way it has survived several centuries. This can also entail a form of occlusion because, in a way, it may imply that some information is missing.
- Non-rigid deformations: Changing the die inevitably changes the exactitude of the relief as it is handmade. Therefore, there can be differences between coins of the same type. For instance, when a die is replaced, but keeps its original symbols, legend and overall monetary value, the new die is undoubtedly different and therefore slight non-rigid deformations are visible from one die to another

and consequently also from a coin from the former die and a coin from the new one.

For these reasons, recognizing ancient coins is a difficult task, often requiring an expertise so that each object can be identified as belonging to a certain type, being created during a specific era, etc. Five different numismatic research areas are identified [1]: the classification of coins into given types, the identification of concrete coin specimens, the identification of coins struck by the same die, the reassembling of broken coin fragments, and the segmentation and surveying of coins. Years of expertise in the field can sometimes not aid in performing efficient recognition, especially if the number of specimens is large. Pairwise matching of coins, for instance, is extremely strenuous and time-consuming.

Hence, we present in this article a new way of matching ancient coin descriptors against a known database. This database is structured in such a way that it makes the retrieval of matching features extremely fast. This model is based on the Low-complexity Arrays of Contour Signature (LACS) model described in [2] specifically applied to the retrieval of ancient coins *struck by the same die*. Contrary to traditional shape matching methods, which retrieve similar shapes, the objective of LACS is to retrieve exact shapes. This means that a shape is only retrieved if there is a rigid transformation or noise between the query and retrieved shape. This is also the objective of the work at hand. Here, we aim to match efficiently exact shapes via sparse texture patches in order to retrieve coins struck by the same die. Traditional shape matching methods retrieving similar shapes would retrieve coins of the same type.

The contributions of the present work are the following:

- To our knowledge, we present the first method for the recognition of ancient coins struck by the same die.
- We propose to compute descriptors for ancient coins on multi-light energy maps [3] to overcome the problem of light conditions.

- We generalize the LACS model to high dimensional descriptors.
- We present two new data sets of artificial and real ancient coins.

The rest of this article is organized as follows. Section 2 reviews related work on coin recognition. Section 3 presents the computation of a light-independent representation of ancient coins, called the multi-light energy map, and the computations of two new descriptors on energy maps for coin recognition. The matching of these descriptors using the generalized LACS model is described in section 4. Our experiments and results are presented and discussed in section 5. Finally, we conclude our work in section 6 and present some perspectives for future research.

2 Related work on coin recognition

Early techniques focus on directly measurable information such as mechanical properties: weight, diameter, thickness etc. More recent methods are based on computer vision using images of coins. That can be global methods such as [4, 5] which are based on the Fourier-Mellin transform for the registration of two coins and the cross-correlation to compute their similarity. Yet, the correlation is not robust enough to deal with small variations in light conditions.

For various reasons besides light conditions, it is preferable to locally describe the objects. For instance occluding manifestations such as wear, tear or oxidation as well as non-rigid deformations present within the relief of the object make it unlikely for a global description to adequately compare two objects. Since coin recognition undergoes first and foremost illumination issues, many techniques deemed the use of edges and contours to be relevant. The authors of [6] used the significant edges, also called edgels for edge elements, independently and sampled their occurrence in quantized polar coordinates for the recognition of modern coins. Of course, modern coins, contrarily to ancient coins, feature sharp edges which are easily captured by an edge detector. Also on modern coins, the authors of [7] performed a classification

using an eigenspace approach on edge images of coins. Various other works [8–10] used similar approaches for both modern and medieval coins, using for instance angular and radial histograms of edgels. 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. In [11] this idea exploited to identify ancient coin images taken under different conditions. They used shape context for the contour representation combined with a texture descriptor. However, edges can be tricky to analyze because they feature topological discontinuities which are influenced by a variety of factors including light conditions and occlusions. Furthermore, the contour of a coin can identify a certain ancient coin, but varies for coins struck by the same die and thus can not be used in our context. The authors of [12] used edge features such as edge width, edge thickness, the number of horizontal and vertical edges, and the total number of edges for counterfeit coin detection. Nevertheless, they used 2D images of modern coins coming from a specialized high resolution 3D scanner to overcome the problem of light conditions.

Hence, texture-based methods have emerged for coin recognition. The method of [13] began by deriving the image gradient, from which they infered the gradient direction in order to align modern coins. In [8] Gabor as well as Daubechies wavelets are tried. These features alone gave medium results on modern coins and very poor results on ancient coins. The authors of [11] used the standard SIFT descriptor for coin classification. In [14] a survey of various descriptors is presented, including SIFT, GLOH, SURF and shape context, along with several point of interest detectors. The results depended largely on the choice of the detector and the descriptor. Overall, SIFT and SURF seemed to be the most robust with respect to the choice of the detector. In another work [15], the authors implemented a solution using SIFT-flow, which consists in minimizing an energy function over a dense grid of SIFT descriptors. Results showed that SIFT alone performs poorly for ancient coin classification whereas SIFT-flow ensured a more reliable score. The works [16, 17] also computed a dense SIFT over

the coin. Their strategy was to use graph transduction games inspired from the game theory to let the images "choose" their class label according to their descriptors and the choice of other coins. The authors of [18] 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. Finally, in [19] Local Binary Patterns (LBP) are used for local texture description of modern coins and Local Pattern of Co-occurrence Matrix (LPCM) for rotation and illumination invariance. It is also possible to combine different ways of describing the coin in order to increase the accuracy of the recognition. In two works [20, 21], both contour information – described with DCSM, a descriptor based on the coin radius variations – and texture description (SIFT) were analyzed and fused for ancient coin identification. The use of SIFT together with DCSM increased only by 1 point the accuracy of DCSM alone due to the discriminative power of the shape of ancient coins. A different approach to information fusion was also tested [22, 23] in which local image description (SIFT-flow) and textual information (extracting letters from the legend of coins) are merged. One can note that legends are not always present on ancient coins, so this might not generalize well to coins other than Roman Republican coins, which they used.

A common general idea is to consider that the coin structure is critical for its recognition. The experiments of [24] took a SIFT-based approach with a dense sampling and adopted a bag of words strategy. The authors tried several tiling schemes to compensate the loss of spatial information of the bag of words approach. The method of [25] also used a bag of words approach without tiling. To take into account the geometric configuration between descriptors corresponding to local features, the author 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 works of [26, 27] relied on the bag of words model with circular tiling, to which they concatenated what

they called pairwise identical words angles histograms (PIWAH) in order to add structural information. In [28] the structural issue is tackled by considering the geometrical consistency in a set of matched features between two coin images. Finally, the works of [29, 30] intended to use the coin composition knowledge as a mean to improve the 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 handcrafted features, some works applied the neural network framework to coin classification. Applications first focused on modern coin recognition [31–33]. With the arrival of convolutional neural networks, deep learning approaches were adopted. The authors of [17] 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 by [34] to recognize modern coins and to build a client/server architecture to enable the use of a smartphone for the recognition. Because of their highly distinguishable relief features and their large availability, modern coins pose 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. The method of [35] employed a hierarchical knowledge scheme to make use of the relationship between Roman imperial coins, the depicted emperor and the symbols on the reverse side. The main idea was to detect landmarks in the coin, which embody areas of significant changes in the final classification accuracy. In the same fashion, the method of [36] intended to identify the emperor shown on the obverse side of the coin. The works of [37, 38] followed a multi-modal perspective by making use of both the image of the coin and its textual description. The learning was thus weakly supervised, meaning labels were not assigned pixel-wise but image-wise, with a filtering of the description only focusing on 5 symbols for semantic learning with AlexNet. Also, the authors

of [39] constructed a new deep learning architecture, named CoinNet, and centered yet again on semantic understanding, associating reverse side symbols to their signification. Although machine learning remains an attractive way, this is not an option in our case due to the limited size of our data set. Moreover, it is very rare that a museum and data set feature several specimens struck by the same die.

To resume, several promising works have been proposed for coin recognition, but methods for modern coins do not adapt to ancient coins as they are subject to wear and occlusion. Moreover, the light conditions remain still a major challenge and none of these methods addresses the problem of coin recognition struck by the same die.

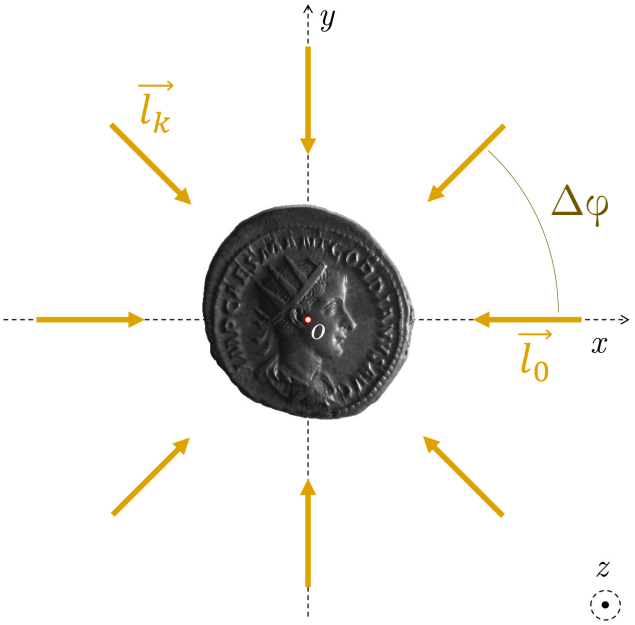
3 Local description of coins

The goal is to determine adequate and robust descriptors with respect to the nature of the object and the task at hand. As stated above, the representation of ancient coins is very sensible to the light conditions and makes it essential to take into account the camera angle and the lighting [14]. The available imaging models for ancient coins range from the simple image to the entire 3D representation of the object. On the one hand, a lot of information is lost in the projection involved in the 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. Hence, we use the representation of multi-light energy map [3] for ancient coins, which is an easy model, both fast to compute and memory-efficient.

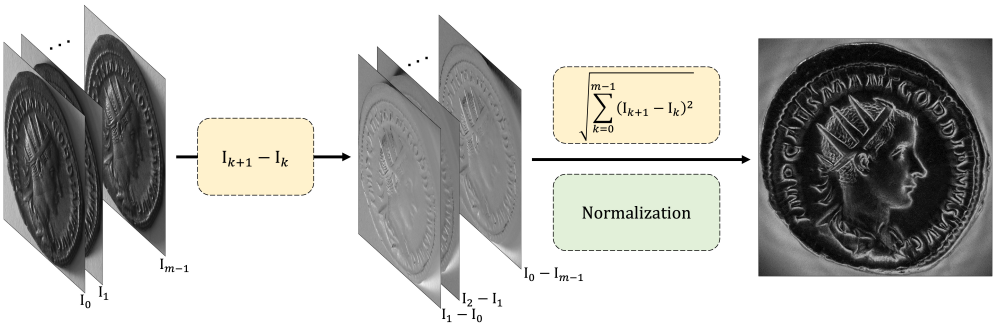
3.1 Multi-light energy map

The multi-light energy map [3] reflects the surface variations of the object. It is akin to an height map model and is therefore a 2.5D model. It physically refers to a sum of

irradiance per unit of angle across light azimuths, which amount to a form of energy received by the object.



(a) Acquisition of an energy map seen from above with $m = 8$.



(b) Pipeline of an energy map computation.

Fig. 1: Acquisition and computation of the multi-light energy map.

Considering the schema in Figure 1a, the object is placed at the point O and m light sources $\vec{l_k}$, $k \in \{0, ..., m-1\}$ are arranged in a circle, equally spaced around the

object. The angular shift between each light source is $\Delta\varphi = \frac{2\pi}{m}$. The camera is placed above the origin O, facing the object along the z-axis. Then, the multi-light energy map $\mathcal{E}$ can be computed as:

$$\mathcal{E}_{(x,y)} = \sqrt{\sum_{k=0}^{m-1} (\Delta I_k)^2} = \sqrt{\sum_{k=0}^{m-1} (I_{k+1} - I_k)^2} \tag{1}$$

where ΔI_k is the image gradient of the kth image. The whole process is summarized in Figure 1b. For more details, we refer to [3].

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. Then, the orientation map can be calculated as the mean angular value weighted by I_k:

$$\forall (x,y) \in U, \quad \mathcal{A}(x,y) = \arctan\left(\frac{\sum_{k=0}^{m-1} I_k(x,y) \cdot \sin(\varphi_k)}{\sum_{k=0}^{m-1} I_k(x,y) \cdot \cos(\varphi_k)}\right) \tag{2}$$

where U is the 2D lattice.

3.2 Computation of texture patches

The idea is to use the surrounding neighborhood around points of interest of the ancient coin to describe and match local textures in the energy map. In our case, we do not have any information about the scale; only keypoints are found with no additional information about the region. Thus select a fixed size of 128×128 to create a patch around the keypoint. 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. To be specific, a patch of length l and centered on (x, y) is defined

as a region $\mathcal{P}$ such that $\mathcal{P} = \left\{(x', y') \mid x - \frac{l}{2} \leq x' \leq x + \frac{l}{2} \text{ and } y - \frac{l}{2} \leq y' \leq y + \frac{l}{2}\right\}$. The values extracted from an energy map $\mathcal{E}$ at the location of the patch $\mathcal{P}$ is denoted as $\mathcal{E}_\mathcal{P}$, and similarly the values extracted from an orientation map $\mathcal{A}$ at the location of the patch $\mathcal{P}$ is denoted as $\mathcal{A}_\mathcal{P}$.

3.2.1 Adapted Harris for the computation of points of interest

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. Hence, applying the Harris 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. However, we can use the object's gradient information formalized as the energy map for the magnitude and the orientation map for the angle. They can be projected back to Cartesian coordinates as follows:

$$\begin{cases} \mathcal{E}_x &= \mathcal{E} \cdot \cos \mathcal{A} \\ \mathcal{E}_y &= \mathcal{E} \cdot \sin \mathcal{A} \end{cases} \tag{3}$$

Based on this we can, we formulate a structure tensor directly from $\mathcal{E}_x$ and $\mathcal{E}_y$:

$$\mathcal{T} = \sum_{(x,y) \in W} \begin{pmatrix} \mathcal{E}_x^2 & \mathcal{E}_x \mathcal{E}_y \\ \mathcal{E}_y \mathcal{E}_x & \mathcal{E}_y^2 \end{pmatrix} \tag{4}$$

where W is a local window in the 2D plane. The remainder 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.

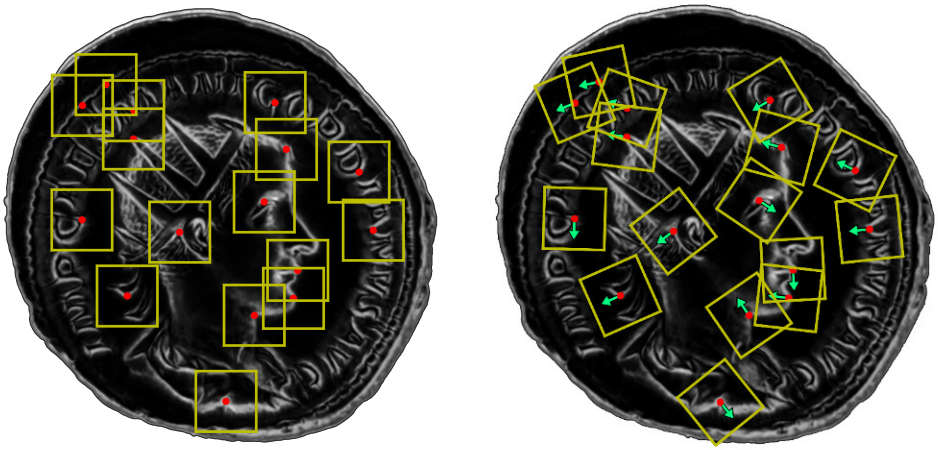
In order to guarantee that the detected points are indeed within an informative region – for instance points detected on the coin field – a selection based on thresholding is performed. The energy map being normalized between 0 and 1, any element of a set of points of interest X, for which the energy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

$$(x,y) \in X \implies \sum_{-\frac{c}{2} \leq i,j \leq \frac{c}{2}} \mathcal{M}(x+i, y+j) \geq \beta \cdot c^2 \tag{5}$$

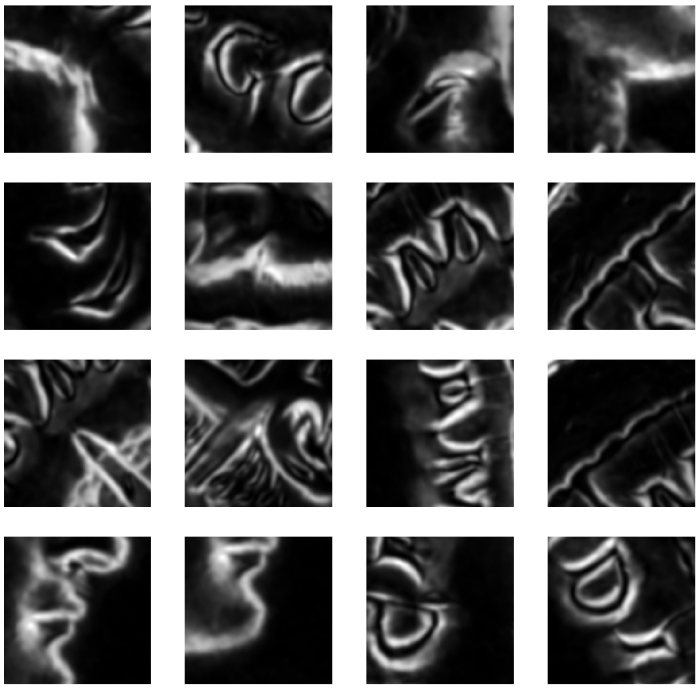
Usually, we choose $\beta \geq 0.8$ in our experiments. Figure 2 displays some examples of patches detected in an energy map.

3.2.2 Texture description with HOG features

In order to describe the patches, the approach formulated by [40] is employed. The Histogram Of Gradient (HOG) is a concatenation of histograms condensing the information of the local gradient in the image. 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 the 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



(a) Patches detected in the energy map. (b) Rotated patches via local pooling.



(c) Details of the energy patches.

Fig. 2: Example of 16 patches found in an energy map.

patch since histograms are built within cells. This is pertinent given that the patch may present signs of erosion and thus looses a part, but not all of its information.

Our energy patches are resized to 64×64 for computational purposes. 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 find the local orientation. Basically, the patch is divided into 8×8 cells, and an 8-bin histogram is computed within each cell. Therefore each histogram is built out of 64 orientation values. Block normalization is performed as illustrated in Figure 3.

The final descriptor is written as in Equation (6), 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.

$$d = \left(\underbrace{d_0^0, ..., d_7^0}_{d^0}, \quad \underbrace{d_0^1, ..., d_7^1}_{d^1}, \quad ... \quad \underbrace{d_0^{195}, ..., d_7^{195}}_{d^{195}} \right) \tag{6}$$

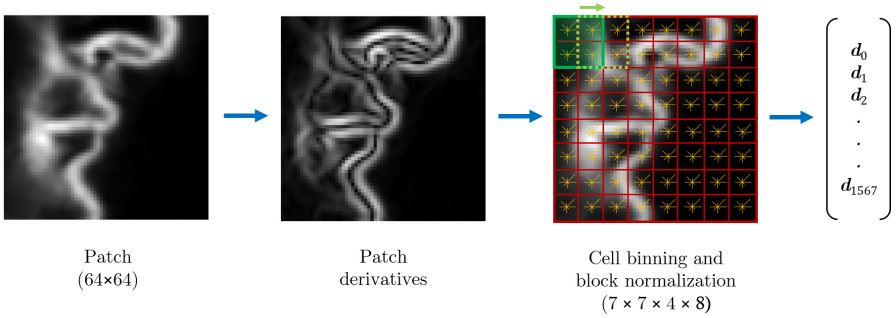


Fig. 3: Principle of HOG. There are 8×8 cells (red) containing an 8-bin histogram (yellow). Each cell is normalized through block normalization. The blocks (green) contain 2×2 cells and are positioned so as to overlap by 50%. There can be 7×7 blocks within the frame, therefore the final descriptor has a length of $7 \times 7 \times 2 \times 2 \times 8 = 1568$.

3.2.3 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. 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 Equation (6). 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 a coarser description, underlining only large variations.

The general observation is that both descriptors are quite similar, but a closer look reveals that they seem to highlight different scales of description. HOME coarsely describes the patch while HOG seems to describe more finely the structure of the patch as shown in Figure 4.

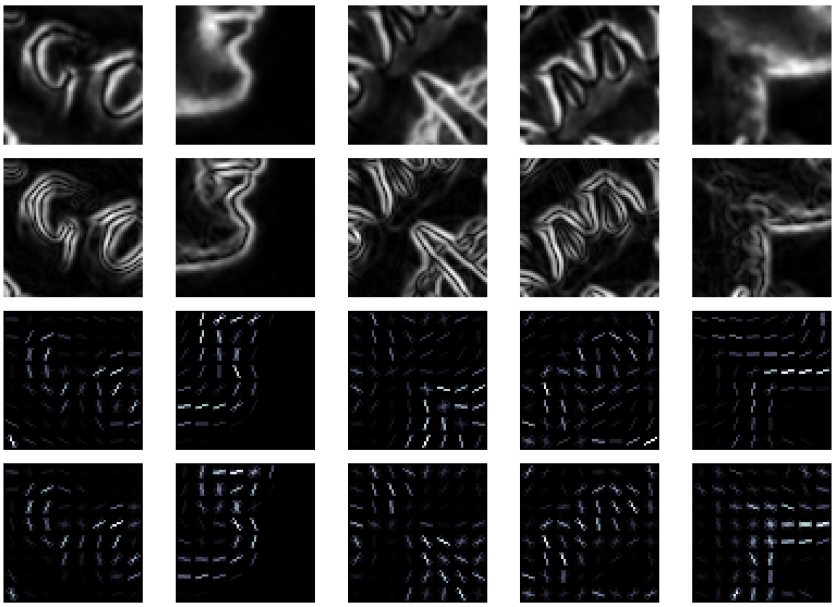


Fig. 4: Various patches and their description. Top row: energy map patches. Top middle row: gradient magnitude of energy patches. Middle bottom row: HOME descriptors. Bottom: HOG descriptors.

3.2.4 Rotation invariance by local pooling

The description of patches through histograms of gradient, should it be HOG or HOME, assumes 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, as 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 consider each patch content regardless of the other features of the coin. The core idea in this method is to extract a small patch (a disk of radius 20 px) $\delta\mathcal{P}$, centered

on each keypoint, of the orientation map, and to compute the average orientation as:

$$\begin{aligned} &\forall (x,y) \in X, \\ &\theta_{\mathcal{P}} = \arctan\left(\frac{\sum_{(x',y')\in\delta\mathcal{P}(x,y)} \mathcal{E}(x',y') \cdot \sin \mathcal{A}(x',y')}{\sum_{(x',y')\in\delta\mathcal{P}(x,y)} \mathcal{E}(x',y') \cdot \cos \mathcal{A}(x',y)}\right) \end{aligned} \tag{7}$$

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

$$\forall (x',y') \in \mathcal{P}, \quad \mathcal{A}_{\mathcal{P}}(x',y') \leftarrow \mathcal{A}_{\mathcal{P}}(x',y') - \theta_{\mathcal{P}} \tag{8}$$

An example of rotated patches and their orientations is illustrated in Figure 2b.

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. This 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.

The strategy taken unfolds in the following manner. Each coin having been described is now a collection of keypoints (patches) and descriptors associated with a label gathering the type index and the die index. All the coins are stored in a database

whose structure is based on the LACS model [2]. The retrieval is performed in a similar manner as well. As long as collisions are well-handled, the 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 outlining the LACS principle and then by describing the general structure of the LAPS system. The outline of the whole process of storage and retrieval is synthesized in Figure 5.

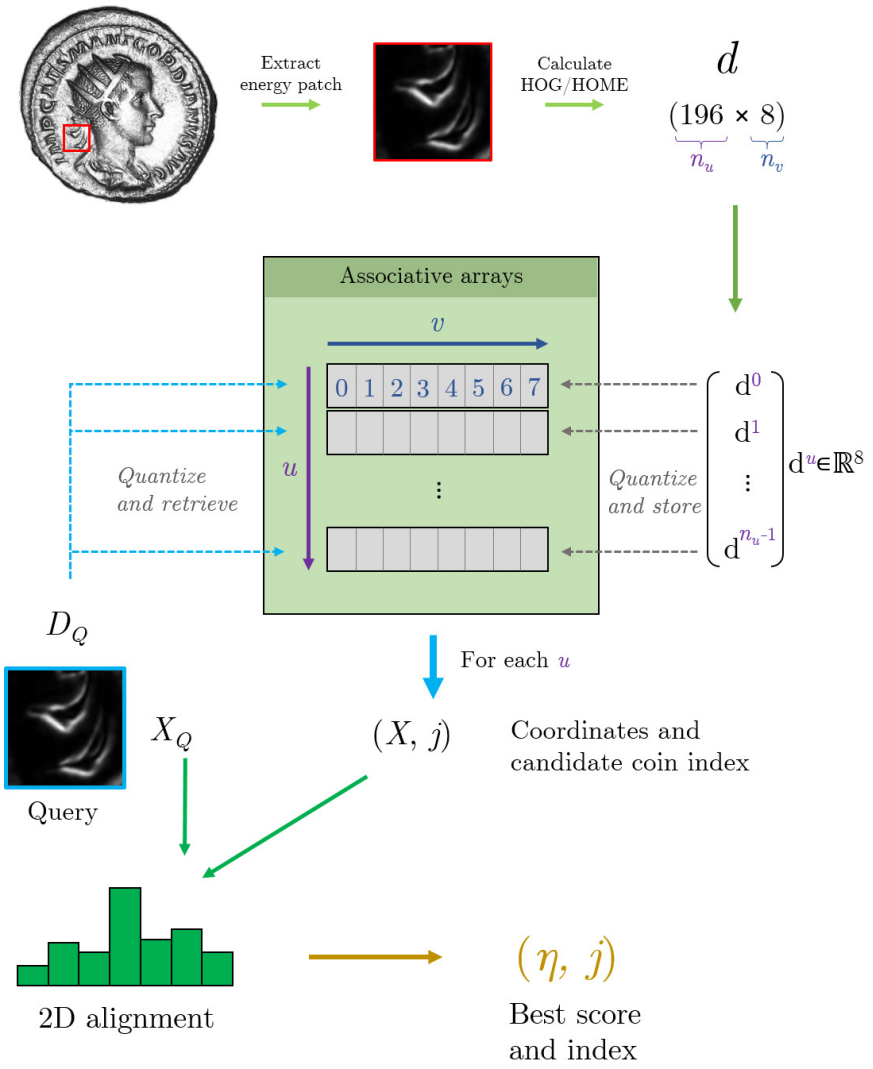


Fig. 5: General structure of the LAPS system. The storage is presented in the first half of the schema. Every patch is converted into its descriptor, each component of which is quantized and stored in each of the n_u=196 tables, containing n_v=8 nested arrays. The information stored in the last array 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.

4.1 LACS

In a previous publication [2], the LACS were introduced as a system of associative arrays. 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. 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.

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 (9):

$$\forall v = 0, ..., n_v - 1, \quad h_v(d_v) = \lfloor q_v \cdot d_v \rfloor \tag{9}$$

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.

In [2], we used a descriptor of three components ($n_v = 3$) along with a set of parameters $\{q_0, q_1, q_2\}$. Figure 6 graphically explains the structure of the LACS system of depth 3 and presents a query example. In the last array, the retrieved value is stored. Thus, $\lfloor q_0 \cdot d_0 \rfloor$ 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 $\lfloor q_1 \cdot d_1 \rfloor$ is used. Finally, at the bucket indexed by this location, the last array h_2 can be found, for which the key $\lfloor q_2 \cdot d_2 \rfloor$ leads to the retrieved value val.

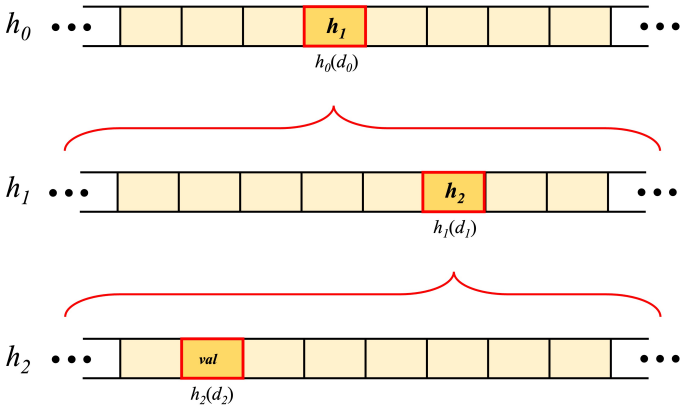


Fig. 6: Structure of the LACS system.

4.2 LAPS structure

In [2] we exploited the signature of the contour, which was just a fragment of a descriptor. By enabling the use of several signatures, that is several descriptor fragments, we propose here a more general form of LACS. We use HOG/HOME descriptors with the LACS paradigm.

The general structure is built around the geometrical arrangement of the descriptor 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 5. Furthermore, since each table corresponds to a histogram of

orientations in HOG or HOME, the structure takes into account the locality of each patch sub-descriptor as illustrated in Figure 7.

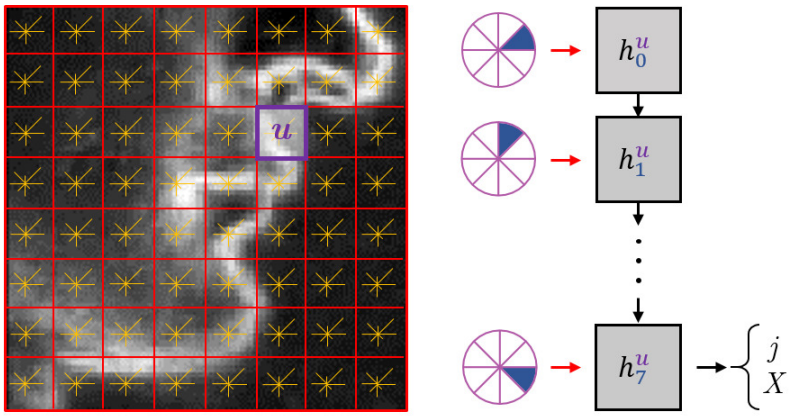


Fig. 7: Details of one nested array for a sub-descriptor of index u. Each sub-descriptor component, represented as an eighth of a disk, leads to a key via h_v^u to determine the location of the next array. During the storing phase, every component is used to determine the final result's position. During the retrieval phase, if one calculated index points to nothing, the retrieval stops and *None* is returned.

We denote by h_v^u the key function of the u^{th} sub-descriptor for its v^{th} component (see Equation (6)). The key function h_v^u is defined as:

$$\forall u \in \{0, ..., n_u - 1\},\ \forall v \in \{0, ..., n_v - 1\},\ h_v^u(d_v^u) = \lfloor q \cdot d_v^u \rfloor \tag{10}$$

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 to retrieve more candidates; 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.

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 1 and Figures 5 and 7 provide illustrations of how to go from a patch and its descriptor to the storage of the components. The final array ($v = 7$ in Figure 5) 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 1, line 4. It has been found experimentally that the recognition performs better when one excludes sub-descriptors having a low Euclidian norm (denoted by $\|.\|$). 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 1: *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 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, 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 2. The computation here is always limited by m_r, the maximum number of retrieved items (Algorithm 2, line 12). If this value is set to infinity then the algorithm ends up exceeding $O(1)$. So, in this case, m_r 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 and Algorithm 3). The value m_r is critical for speed 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. 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_r 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.

Algorithm 2: *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_r: 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**
  $L \leftarrow get(\mathcal{H}^u, q, d^u)$ `// 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`
  $truncate((\eta, \mathrm{J}, X),\ m_r)$ `// keep the` m_r `best indices`
return J, X

The way the structure, the storage and the retrieval behave is certainly different from how a classic Euclidean brute force matching would be. Indeed it can be shown that our 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.

4.4 Geometric alignment

We want to be able to assess during the retrieval if the geometric transformation from one set of points to the other is the same for a significant amount of points. 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 the LACS algorithm: we want to align the query with the candidates 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 are two-dimensional. 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. The index $\mathcal{Q}$ denotes one query patch in the query coin.

4.4.1 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 (11), where x is the abscissa of the retrieved patch. This definition is identical along the y axis.

$$\Delta x = \left\lfloor \frac{x - x_{\mathcal{Q}}}{\varepsilon} \right\rfloor \tag{11}$$

The quantity ε is a quantization divisor which simply represents the step, in pixels, between each bin in the histogram. In essence, ε is merely a tolerance in the 2D lattice. The histogram is built for a coin in particular, so that each candidate coin has its own histogram.

4.4.2 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. The usual way to perform a rotation is to capture the coin in a random orientation. 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 $(x, y) \in X$:

$$\begin{cases} \rho = \sqrt{(x - x_C)^2 + (y - y_C)^2} \\ \theta = \arctan\left(\frac{y - y_C}{x - x_C}\right) \end{cases} \tag{12}$$

We can now build a histogram and use a similar expression for the quantization (Equation (13)), where $\delta\theta$ is defined by the minimum angle difference in Equation (14).

$$\begin{cases} \Delta\rho = \left\lfloor \frac{\rho - \rho_{\mathcal{Q}}}{\varepsilon_\rho} \right\rceil \\ \Delta\theta = \left\lfloor \frac{\delta\theta}{\varepsilon_\theta} \right\rceil \end{cases} \tag{13}$$

$$\delta\theta = (\theta - \theta_{\mathcal{Q}} + \pi \pmod{2\pi}) - \pi \tag{14}$$

This time though, Equation (13) 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 one 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 and rotation 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. 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.

4.4.3 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 8, 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.

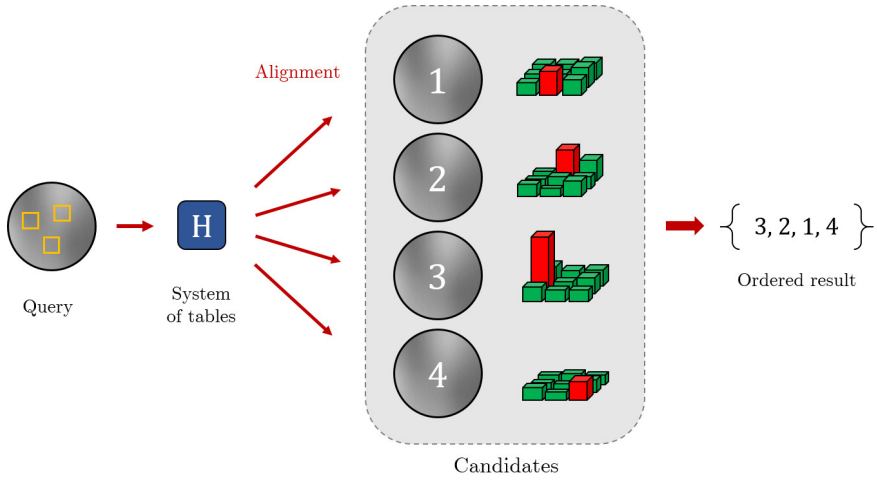


Fig. 8: Pipeline of the LAPS system. Descriptors from a query coin are used to retrieve values in the system of tables. Retrieved keypoints are then compared to those of the query during the geometric alignment phase. Histograms are built for each candidate coin and the final result is an ordered sequence of the highest scores.

The whole computation is presented in Algorithm 3. 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. It is important to keep in my mind that the algorithm is able to keep a list of K final matches at the time, which is performed by the *update_matches* function in Algorithm 3. It takes five parameters: the current list of matches J, the corresponding scores η, the index of the the current match j (which will potentially be added to J), its current score b in the histogram and the maximum number of retained matches K. Each time a candidate match is proposed (line 11 to 16), *update_matches* adds it to the list of K best matches J if and only if its score is at least superior to the lowest ranked match. The update then straightfowardly inserts the match within the list according to its score, and the previous lowest value is discarded. The process is thus repeated for each match provided successively. It follows of course that the list may alternatively add and delete a match if it is proposed several times.

We can also 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.

5 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 are performed under a same objective, that is to perform

Algorithm 3: *match*: Finding matches in the database from a coin query

Input : $\mathcal{H}$: system of tables
X: query keypoints
d: query descriptors
m_r: 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`
  $R \leftarrow retrieve(\mathcal{H}, d, m_r)$
  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 (11) or (13))
      $hist[j][\Delta x][\Delta y] \leftarrow hist[j][\Delta x][\Delta y] + 1$ `// ensure it exists first`
      $b = hist[j][\Delta x][\Delta y]$
      $update\_matches(\mathrm{J}, \eta, j, b, K)$

return J, η

die recognition; it means retrieving, for a given query, coins which were made from the same die. We perform comparative experiments of the HOG and HOME descriptors with the use of five classical key point detectors: Gaussian and multi-Gaussian maxima, Harris, and Laplacian of Gaussian. The experiments begin with an artificial

data set to compute quantitative results. Then, tests are performed on a data set of real coins to qualitatively assess the recognition pipeline. All the computations were made on an Intel®Core^{TM}i7-7820HQCPU @ 2.90GHz processor, using Python 3.6.

5.1 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. The data set presented below is sufficiently large and precise – coins from the same die share many exact resemblances with each other – to validate the tools built so far. In order to keep the experiments simple, the final score of the retrieval will be calculated as presented in Algorithm 3, that is the amount of elements in the largest bin of the histogram of alignment.

5.1.1 Artificial coin data set

The main problem with coin data sets 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 several captures can be taken in order to build the energy map as described above.

In order to perform computations in good conditions, build algorithms and collect results, the idea has been to use artificial coins instead of real ones. This enables the creation of a suitable data set 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.

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.

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. 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.

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 choose 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 $D \leftarrow D - \frac{1}{2}S$. We consider large and small symbols. One large symbol is always present at the center of the die, and small symbols are placed around it, much like letters would be engraved around a roman emperor's bust.

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 allows 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 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. Finally, another noisy layer is superimposed by adding a normal noise smoothed by a Gaussian filter of standard deviation 13.

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. A grid of regularly spaced points is set across the 2D lattice containing the symbol. 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 piece-wise affine transform, which is finally applied to the symbol itself.

Using the entire protocol described above, a series of 16 types of coins were generated. These types are displayed in Figure 9. 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 10a. As one can note, the difference between each die can be subtle. 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 10b, 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.

Fig. 9: 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.

(a) Dies from the same types. They all feature the design of the type, but exhibit some small difference due to non-rigid deformations.

(b) 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.

Fig. 10: Examples of coins produced by the algorithm for different dies and for the wear of a die.

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.

5.1.2 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 1. One may notice that the parameter m_r is quite low. Considering the amount of data, it has been clear from the start that keeping all retrieved values in Algorithm 2 leads to a very long run time. This was an expected risk considering that it directly impacts the complexity. Therefore, the value m_r was chosen to be low and is set to the average number of

[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.

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

Table 1: Parameters of the experiments for the retrieval of artificial coins.

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 tolerance 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 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.

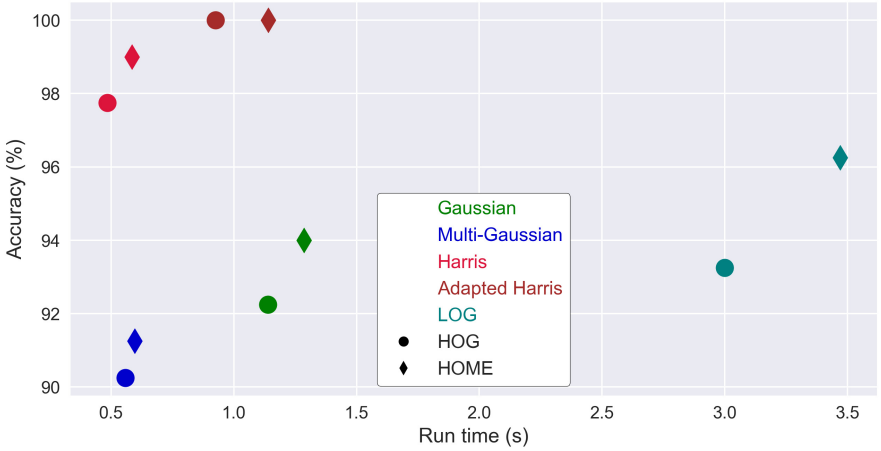


Fig. 11: Accuracy of retrieval against the average run time per coin. There are 5 colors for the detectors and 2 shapes for the descriptors. The results show a high accuracy and a quite low computational time for each method. Note that the ordinate axis begins at 90%.

The results are presented in Figure 11. 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 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 it detects only a low amount of points of interest, which does not enable a rich description. In terms of retrieval time per coin, one can say that the results are also satisfactory: all 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 of such variations in run time is that there are variations in the number of key points, depending on the method. 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). To conclude, the adapted Harris detector performs best, but has also a slightly increased run time with respect to the simple Harris detector. The HOME descriptor achieves a better accuracy than the HOG, but has also a slightly increased run time.

As stated before, the number of retrieved values should be determined by the number of collisions in the database. In Figure 12, the distribution of collisions in the tables for the 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.

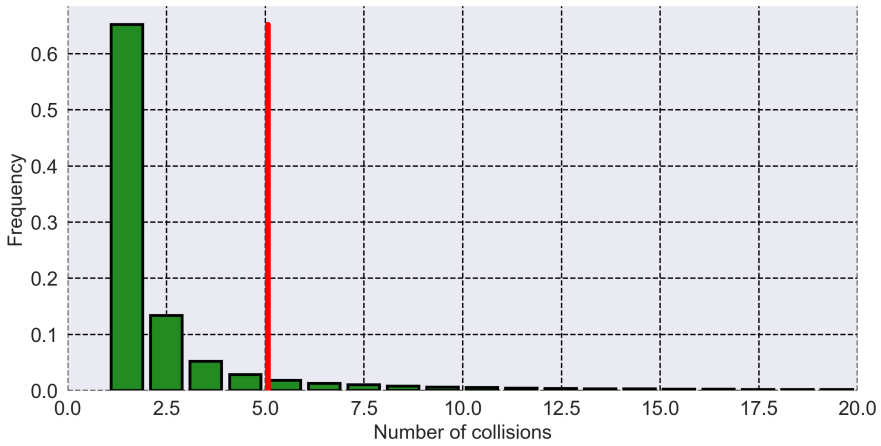


Fig. 12: Distribution of collisions in the database for HOME ($q = 10$). The red vertical line is the average number of collisions $\gamma = 5.06$. There are a total of 663887 values stored.

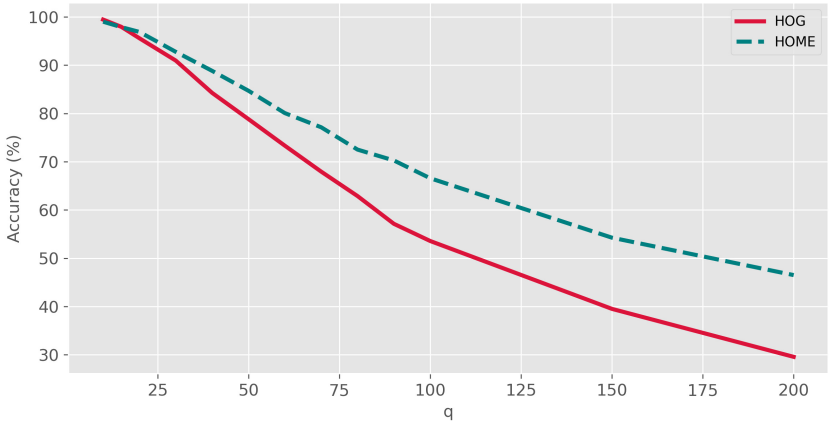
5.1.3 Accuracy and run time against quantization

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. This is controlled by the quantization parameter q, which determines the width of cells of the associative arrays. 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.

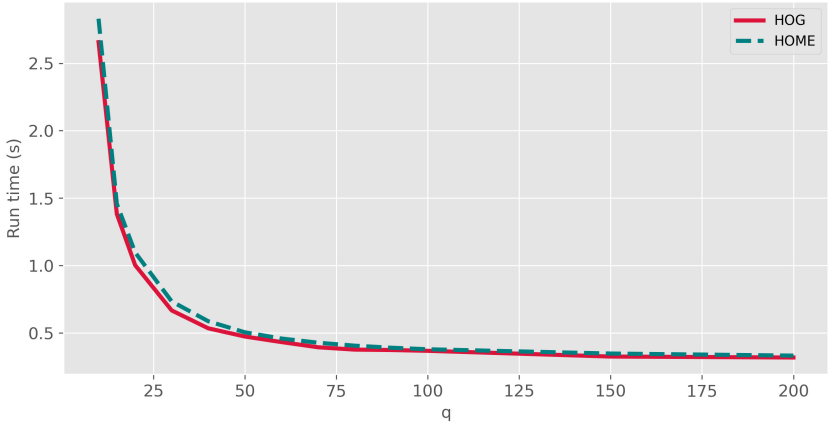
In Figure 13a, the accuracy of retrieval against q is presented. As one can expect, the accuracy is dependent on q and decreases when q increases. This is fairly logical: the cells contain less and less values (collisions are fewer and fewer), which requires for the descriptor values to be more and more 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 13b. High values of q (> 50) produce a very fast retrieval, with under 0.5 seconds per coin (around 3 minutes for all the coins). There is a hyperbolic increase of the run time as q decreases under 50. A quantization of 10, used previously, gives a run time superior to 2.5 seconds per coin (around 17 minutes 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.



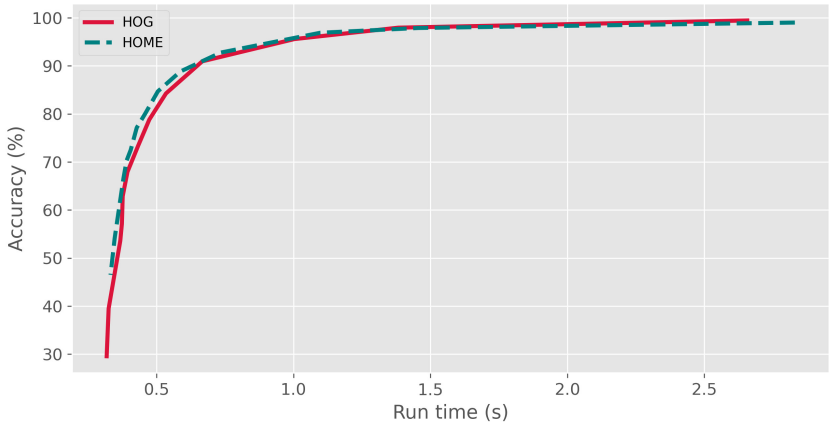
(a) Accuracy against quantization.



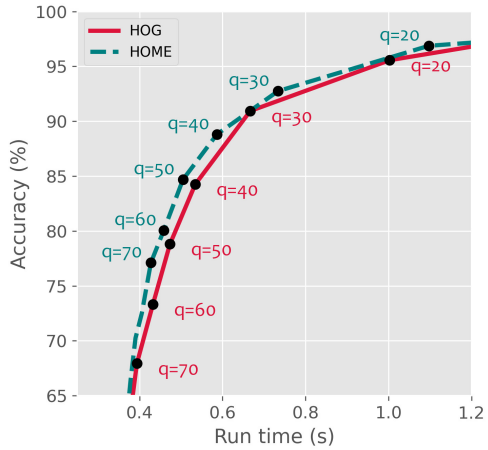
(b) Run time against quantization.

Fig. 13: Accuracy and run time against quantization from $q = 10$ to $q = 200$. The results for both descriptors are shown.

To summarize those two graphs, it is interesting to plot the graph of accuracy against run time, which is displayed in Figure 14. 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 in Figure 14a, 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 14b. 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 loose 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.



(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.

Fig. 14: Accuracy against run time trade-off.

Finally, in Figure 15 is represented the run time per coin as a function of the number of collisions. 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.

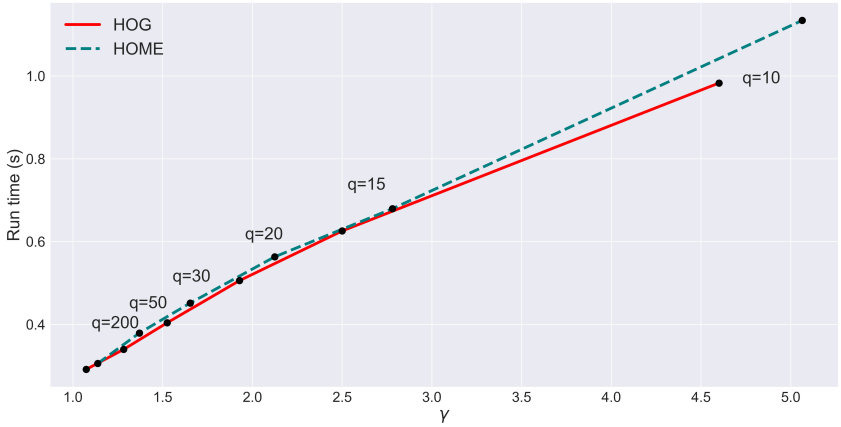


Fig. 15: Run time per coin against the average number of collisions γ in the system database.

5.1.4 Rotated coins

In this case, the patches are extracted 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). The experiment follows the same protocol as for the previous tests.

The results are presented in Figure 16. As a general observation, the accuracy drops with respect to unrotated coins. The detectors follow the same order as already established, with the adapted Harris ahead and reaching up to 80.5% using HOG. However, the order of the descriptors has changed: HOG seems to prevail over HOME for the Harris and the adapted Harris detectors. 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.

# coins	Straight		Rotated	
	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 2: Accuracy (%) of the retrieval against the number of coins per die.

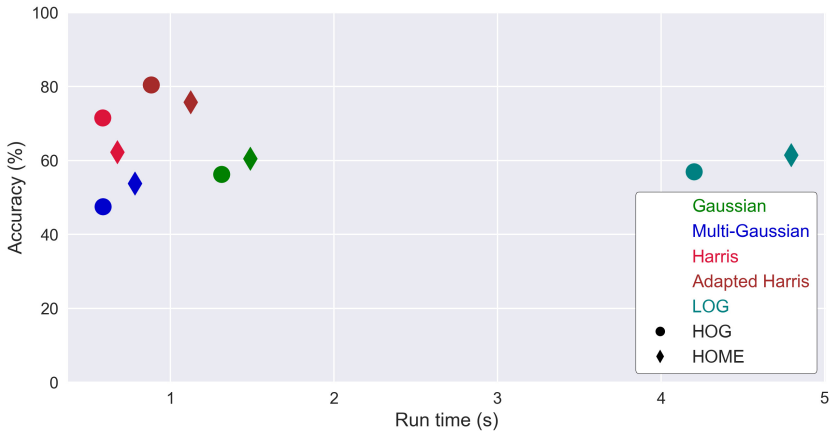


Fig. 16: Accuracy against run time trade-off for patch extraction with rotation.

Finally, 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 2. 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. 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.

5.1.5 Comparison with other methods

Finally, we compare here the performance of our methods with other methods of detection, description and comparison for upright coins. Because SIFT is a well-established

solution programmed within code libraries, it is easy to test. It was used on the artificial 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 [41]. The overall angle is always the same, but varies slightly via a small random angle. The LIDRIC method of [18] 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 key points and descriptors have been used as they are.

The matching technique for SIFT is traditionally a 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, we present also some results for the LACS method [2]. LACS uses a modified version of the TCD descriptor [42] to describe the shapes in the image. The TCD is based on the Fourier transform of distances between points of the shape from which the local extrema are used as feature points in the LACS system. Hence, this means that contours have to be extracted from the energy map to obtain shapes. We test two methods for contour extraction of the energy map, Canny and the skeletonization method [43]. The parameters of LACS have been adapted to the length of the contours: $q = \{q_0 = 1,\ q_1 = 1,\ q_2 = 1\}$, $\delta s_{min} = -100$, $\delta s_{max} = 100$, $N_B = 200$, and $\varepsilon = 100$ px. Moreover, we used $n_u = 4$ parallel tables in the results.

The results are shown in Table 3. The retrieval times should only be considered as an indication. The computations were not run with the same programming language for each of them (Python for LAPS and LACS, 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

Method	Accuracy (%)	Total run time (h)
SIFT on $\mathcal{E}$ (L_2)	91.8	13.30
SIFT on images (L_2)	80.8	3.16
LIDRIC on images	63.3	> 140
LACS with skeleton	87.0	1.01
LACS with Canny	78.2	0.0063

Table 3: Accuracy and retrieval time for various methods. The retrieval times displayed are just raw indicators of speed because different programming languages were used.

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 ones found in simple images. LIDRIC seems not sufficiently discriminatory and is very expensive in terms of computation time. It is supposed to be very robust to illumination changes, yet it does not seem to take advantage of this property in this experiment. Finally, we can note that LACS strongly depends on the robustness of the contour extraction in the energy map, whereas the skeletonization method gives better results than the Canny method. This is even worse for images of real coins. Hence, this is the reason why we are working with texture patches and not with shapes.

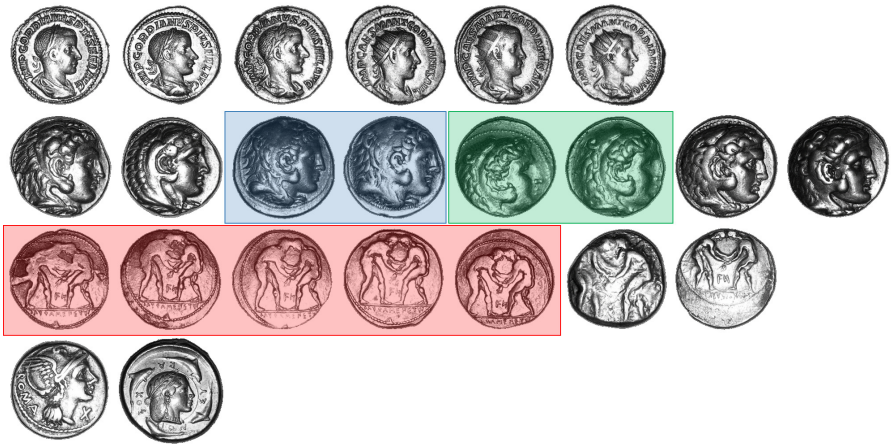
5.2 Retrieval of real coins

So far, the experiments have been based on an artificial data set. While this is useful to infer statistical results about the retrieval pipeline, it constrains the tuning of the parameters to this specific data set, which may of course slightly differ for real acquisitions. Hence, we present here a real data set and results obtained with it.

5.2.1 Real coin data set

We present a data set of 23 coins. The coins are introduced in Figure 17, 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.

One die is used for one side and both sides of a coin may not have been struck from the same pair of dies. For instance, the blue highlighted coins in Figure 17a does not have a reverse counterpart in Figure 17b. We can therefore consider that the data set has 46 ”coins”, composed of some obverses and some reverses (although in reality there are 23 coins, each with one obverse and one reverse). Figures 17a and 17b show one picture per coin. Of course, the acquisition is done on each coin (obverse and reverse) according to the process described in Section 3.1 so as to be able to compute the energy map and the orientation map.



(a) Obverse of the dataset.

(b) Reverse of the dataset.

Fig. 17: Data set 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.

We performed the image acquisition using a smartphone (iPhone 6S). In theory, placing the light source at a colatitude of 90° is the best choice for a highly contrasted image. In practice, however, the light source cannot be a single point and covers a range of angles in horizontal and vertical directions – in our case between 72° and 82°.

The main drawback of this data set 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.

5.2.2 Parameter settings

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. 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_r 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. 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_r	10
ε	64
$(\varepsilon_\rho, \varepsilon_\theta)$	(50,15°)
K	5

Table 4: Parameters of the experiments for the retrieval of real coins.

5.2.3 Scoring methodology

It has been observed that the simple score employed in Section 5.1 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 a 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 18a, the correct coins are retrieved, but the scores are not discriminative.

As mentioned in Section 4.4.3, the score can also be calculated using 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 the basic one and thus was not suited for the previous extensive calculations of run times.

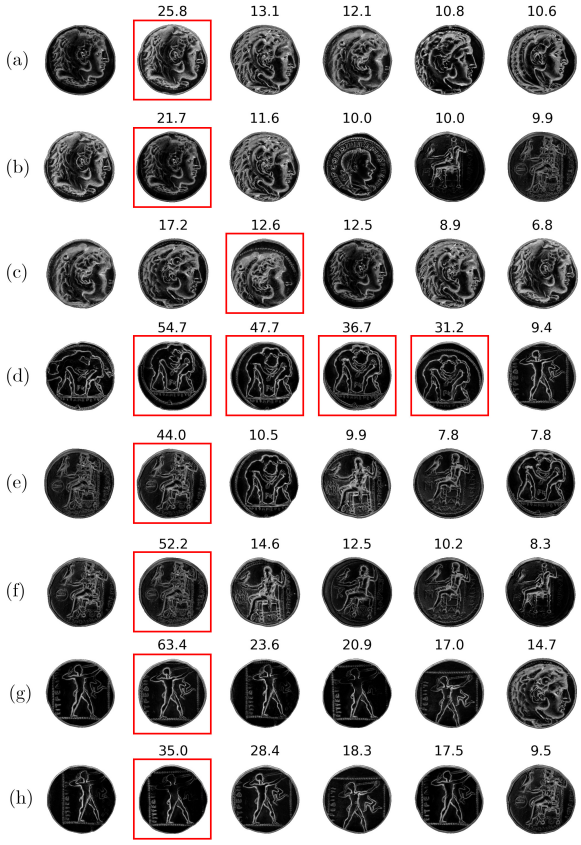
The new scoring method is presented in Figure 18b. There is no significant difference in the candidates, but their order varies. In particular, it enables to discriminate between the two last coins. The results are more interpretable and they are less influenced by the number of key points in the candidates. Surjections are ignored because they cover the same area.

(a) Amount in the largest bin of alignment.

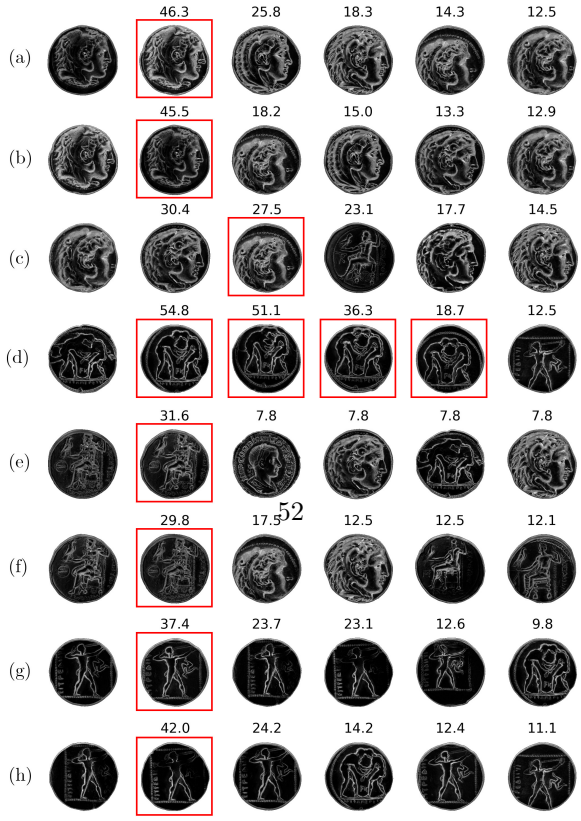
(b) Percentage of area covered in the query.

Fig. 18: Difference between two types of scoring method with the 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.

5.2.4 Visual inspection



(a) HOG descriptor.



(b) HOME descriptor.

Fig. 19: Retrieval results obtained using the two descriptors. The query is the leftmost coin and the candidates are ordered from left to right after it. Red squares indicate the correct expected candidates.

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 5.2.2, the algorithm returns $K = 5$ candidates. The results are presented in Figures 19a and 19b.

Several comments can be made: 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.

5.2.5 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 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 the artificial database that 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. 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 the thresholding (Algorithm 1 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. These results are listed in Table 5. As already noted, HOG discriminates better than HOME, albeit featuring a

Descriptor	Min	Max	Mean	*st.d.*
HOG	3.48	41.6	21.6	11.1
HOME	3.91	30.2	17.2	8.16

Table 5: 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.

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

Table 6: Run time (s) vs q per descriptor.

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.

Quantization q: Finally, results of Mean Average Precision (MAP) and run time have been computed against the quantization factor q, and can be found respectively in Figure 20 and Table 6. Top results in MAP are obtained across the range from $q = 5$ to $q = 7$ for HOME, 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 seconds for HOG. 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.

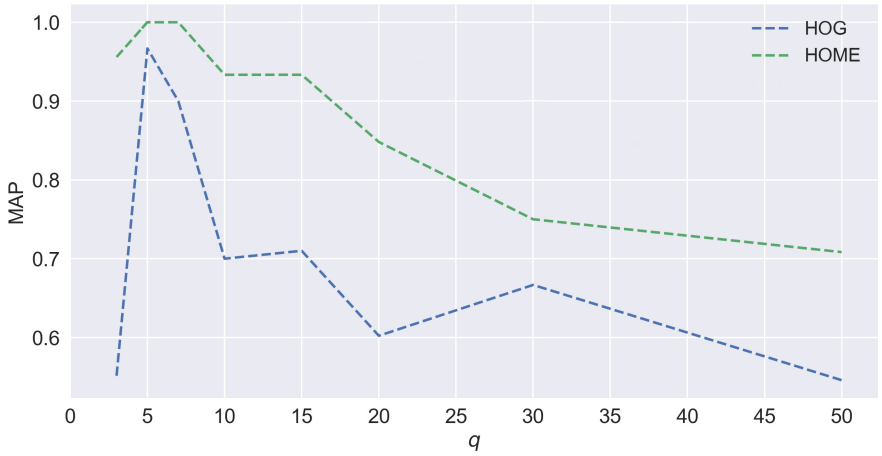


Fig. 20: Mean Average Precision against the quantization factor q for both descriptors.

5.2.6 Randomly rotated coins

In much the same way as was done previously in Section 5.1.4, 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 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 enables to fetch values in the tables, which here depends on the patch orientation, and the second one is a polar alignment of the patch position. Coins were randomly rotated around the middle of the frame (512,512). The protocol is then the same as usual and the MAP 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 reliefs by themselves are not automatically centered this way.

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, where p designates a point in U:

$$p_{\text{center}} = \frac{\sum_p \mathcal{E}(p) \cdot p}{\sum_p \mathcal{E}(p)} \tag{15}$$

(a) Original coin.

(b) Coin after centering.

Fig. 21: 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).

The results are gathered in Table 7. The MAP is, on average, worse than for upright coins (with the same general parameters). As predicted by the study on artificial coins, HOG performs on average better than HOME. 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. 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 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 patch extraction and one on the alignment.

To resume, the current results on the real data set are promising. The various scores obtained for upright and rotated coins, the behavior of the tables with respect

	HOG	HOME
Original	0.63 (± 0.11)	0.52 (± 0.08)
Centered	0.66 (± 0.07)	0.58 (± 0.08)

Table 7: 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.

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 data set. Yet, those modifications are not major in that they do not differ radically from previously chosen parameters.

A quite good retrieval can be performed with the current parameter settings. It is more than likely that those parameters are transferable to other, larger data sets. The transition from artificial to real coins went, overall, rather smoothly, which both indicates the relevancy of the carefully constructed artificial data set 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.

5.3 Retrieval of the PHOS data set

In order to show that our method can be applied to other applications. We show here some retrieval results for the PHOS data set [44]. It consists of images of 15 scenes captured under different illumination conditions. More particularly, for every scene 15 different images are available: 9 images captured under various strengths of uniform illumination, and 6 images under different degrees of non-uniform illumination. The images contain objects of different shapes, colors, and textures. We compare our method here with two shape retrieval methods [45] and [46].

Method	Uniform illumination	Non-uniform illumination
TFD	85.50	94.92
OLTT	84.44	93.23
LAPS with HOG	85.93	88.89

Table 8: Retrieval accuracy (%) for the PHOS data set.

Fig. 22: Top-8 retrieval results for the images with uniform illumination. The left image is the query image (black border) and the following images are the retrieved images. Images with a red border are wrongly retrieved images.

We show the retrieval results in Figures 22 and 23 as well as the retrieval accuracy in Table 8. We can note that our method has a slightly better performance than the TFD [46] and OLTT [45] methods for the images with uniform illumination, and a small loss in performance for the images with non-uniform illumination. However, the

Fig. 23: Top-5 retrieval results for the images with non uniform illumination. The left image is the query image (black border) and the following images are the retrieved images. Images with a red border are wrongly retrieved images.

first retrieved images are all correct for our method. This can be explained by the fact that the TFD and OLTT methods extract binary shapes from the image which are then transformed into a feature vector, whereas our method is a sparse texture patch based method. The images of the PHOS data set contain clear contours and few textures. Moreover, we used the same parameter settings than for the other data sets. Tuning the parameters might lead to better results. The time complexities are very different. TFD and OLTT use a kNN classifier and cross-validation for the retrieval which have a linear time complexity with respect to the number of images whereas our

method retrieves in nearly constant time (0.214 s for a request). Hence, our method is much more faster, but with a small loss in retrieval accuracy.

6 Conclusion and perspectives

We presented in the article a new method for the retrieval of ancient coins stuck by the same die. To overcome the problem of light condition, we used the multi-light energy map as a representation of the coin. We presented a method, the adapted Harris, to compute points of interest in the energy map. Patches centered at the points of interest are described by two methods: the histogram of gradients of energy map patches and the histogram of oriented map of energy. Compared to other descriptors, we deem that the adapted Harris is nonetheless the best choice for detection, as it detects a reasonable amount of points which were shown to be located in highly informative areas.

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 gathers meaningful data about the object.

As a matching strategy, the LACS system [3] was upgraded so that it fits not only signatures but entire descriptors. This new matching system is called LAPS for Low-complexity Arrays of Patch Signatures. 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 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 enabled to demonstrate the transferability 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 visible. Furthermore, we have shown on a different data set which does not contain coin images that the method can be adapted to other retrieval tasks.

Declarations

Ethical Approval. This declaration is not applicable.

Funding. This work was financed by the Region of Nouvelle-Aquitaine, France.

Availability of data and materials. The data sets generated and analyzed during the current study are available in the L3i repository (http://l3i-share.univ-lr.fr/2024AncientCoins/index.html).

Competing interests. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Zambanini, S., Kampel, M., Schlapke, M.: On the use of computer vision for numismatic research. 9th International Symposium on Virtual Reality, Archaeology and Cultural Heritage **45**(7), 17–24 (2008)

[2] Lardeux, F., Marchand, S., Gomez-Krämer, P.: Low-complexity arrays of contour signatures for exact shape retrieval. Pattern Recognition **118**, 108000 (2021) https://doi.org/10.1016/j.patcog.2021.108000

[3] Lardeux, F., Marchand, S., Gomez-Krämer, P.: Multi-light energy map. In: Eurographics Workshop on Graphics and Cultural Heritage (EG GCH), Vienna, Austria (2018)

[4] Marchand, S., Desbarats, P., Vialard, A., Bechtel, F., Amara, B., Cicuttini, B., Bost, J.-P., Alain, B., Koray, K., Beurivé, A.: IBISA: Image-Based Identification / Search for Archaeology. In: International Symposium on Virtual Reality, pp. 57–60 (2009)

[5] Marchand, S.: IBISA: Making Image-Based Identification of Ancient Coins Robust to Lighting Conditions. In: EUROGRAPHICS Workshop on Graphics and Cultural Heritage (GCH), pp. 13–16 (2014)

[6] Nölle, M., Penz, H., Rubik, M., Mayer, K., Holländer, I., Granec, R.: Dagobert – A New Coin Recognition and Sorting System. In: Digital Image Computing: Techniques and Applications, pp. 329–338 (2003)

[7] Huber, R., Ramoser, H., Mayer, K., Penz, H., Rubik, M.: Classification of coins using an eigenspace approach. Pattern Recognition Letters **26**(1), 61–75 (2005)

[8] Maaten, L.J.P., Postma, E.O.: Towards automatic coin classification. In: EVA-Vienna, pp. 19–26 (2006)

[9] Maaten, L.J.P., Boon, P.J.: COIN-O-MATIC : A fast system for reliable coin classification Muscle CIS benchmark. In: MUSCLE Coin Workshop, pp. 7–17 (2006)

[10] Zaharieva, M., Kampel, M., 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**, 547–554 (2007)

[11] Zaharieva, M., Huber, R., Nölle, M., Kampel, M.: On ancient coin classification. In: International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST), pp. 55–62 (2007)

[12] Hmood, A.K., Suen, C.Y.: Statistical edge-based feature selection for counterfeit coin detection. Multimedia Tools and Applications (79), 28621–28642 (2020)

[13] Reisert, M., Ronneberger, O., Burkhardt, H., Rösch, P., Harz, M., Schmitt, M., Peschke, K.D., Motzkus, H.W., Lankers, M., Hofer, S., Others: An efficient gradient based registration technique for coin recognition. In: Muscle CIS Coin Competition Workshop, vol. 19, p. 31 (2006)

[14] Kampel, M., Zambanini, S.: Coin data acquisition for image recognition. Proceedings of the 36th CAA Conference, 163–170 (2008)

[15] Zambanini, S., Kampel, M.: Automatic Coin Classification by Image Matching. In: International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage (VAST), pp. 65–72 (2011)

[16] Aslan, S., Vascon, S., Pelillo, M.: Ancient Coin Classification Using Graph Transduction Games. In: IEEE International Conference on Metrology for Archaeology and Cultural Heritage (MetroArchaeo), pp. 127–131 (2018)

[17] Aslan, S., Vascon, S., Pelillo, M.: Two sides of the same coin: Improved ancient coin classification using graph transduction games. Pattern Recognition Letters **131**, 158–165 (2019)

[18] Zambanini, S., Kampel, M.: A local image descriptor robust to illumination changes. Lecture Notes in Computer Science **7944 LNCS** (2013)

[19] Lin, W.J., Jhuo, S.S.: Coin recognition based on texture classification on ring and fan areas of the coin image. In: IEEE Instrumentation and Measurement Technology Conference, pp. 1–6 (2016)

[20] Huber-Mörk, R., Zambanini, S., Zaharieva, M., Kampel, M.: Identification of ancient coins based on fusion of shape and local features. Machine Vision and Applications **22**(6), 983–994 (2011)

[21] Huber-Mörk, R., Nölle, M., Rubik, M., Hödlmoser, M., Kampel, M., Zambanini, S.: Automatic Coin Classification and Identification. Advances in Object Recognition Systems, pp. 127–154. IntechOpen, London (2012)

[22] Zambanini, S., Kavelar, A., Kampel, M.: Improving ancient Roman coin classification by fusing exemplar-based classification and legend recognition. Lecture Notes in Computer Science **8158 LNCS**, 149–158 (2013)

[23] Kavelar, A., Zambanini, S., Kampel, M., Vondrovec, K., Siegl, K.: The ILAC-Project: Supporting ancient coin classification by means of image analysis. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **XL-5/W2**, 373–378 (2013)

[24] Anwar, H., Zambanini, S., Kampel, M.: A bag of visual words approach for symbols-based coarse-grained ancient coin classification. ArXiv **abs/1304.6192** (2013)

[25] Arandjelović, O.: Automatic Attribution of Ancient Roman Imperial Coins. In: Conference on Computer Vision and Pattern Recognition, pp. 1728–1734 (2010)

[26] Anwar, H., Zambanini, S., Kampel, M.: A Rotation-invariant Bag of Visual Words Model For Symbols Based Ancient Coin Classification. In: International Conference on Image Processing (ICIP), pp. 5257–5261 (2014)

[27] Anwar, H., Zambanini, S., Kampel, M., Vondrovec, K.: Ancient coin classification using reverse motif recognition: Image-based classification of roman republican coins. IEEE Signal Processing Magazine **32**(4), 64–74 (2015)

[28] Zambanini, S., Kavelar, A., Kampel, M.: Classifying ancient coins by local feature matching and pairwise geometric consistency evaluation. In: International Conference on Pattern Recognition, pp. 3032–3037 (2014)

[29] Kim, J., Pavlovic, V.: Ancient coin recognition based on spatial coding. In: International Conference on Pattern Recognition, pp. 321–326 (2014)

[30] Kim, J., Pavlovic, V.: Improving Ancient Roman Coin Recognition with Alignment and Spatial Encoding. Lecture Notes in Computer Science **8927**, (2015)

[31] Fukumi, M., Omatu, S., Takeda, F., Kosaka, T.: Rotation-Invariant Neural Pattern Recognition System with Application to Coin Recognition. IEEE Transactions on Neural Networks **3**(2), 272–279 (1992)

[32] Bremananth, R., Balaji, B., Sankari, M., Chitra, A.: A new approach to coin recognition using neural pattern analysis. In: International Conference of IEEE India Council (INDICON), pp. 366–370 (2005)

[33] Modi, S., Bawa, S.: Automated coin recognition system using ANN. International Journal of Computer Applications **26**(4), 13–18 (2011)

[34] Capece, N., Erra, U., Ciliberto, A.V.: Implementation of a Coin Recognition System for Mobile Devices with Deep Learning. International Conference on Signal

Image Technology and Internet-Based Systems (SITIS), 186–192 (2016)

[35] Kim, J., Pavlovic, V.: Discovering characteristic landmarks on ancient coins using convolutional networks. In: International Conference on Pattern Recognition, pp. 1595–1600 (2016)

[36] Schlag, I., Arandjelović, O.: Ancient roman coin recognition in the wild using deep learning based recognition of artistically depicted face profiles. In: IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 2898–2906 (2018)

[37] Cooper, J., Arandjelović, O.: Understanding ancient coin images. arXiv (2019) https://doi.org/10.1007/978-3-030-16841-4_34

[38] Cooper, J., Arandjelović, O.: Learning to Describe: A New Approach to Computer Vision Based Ancient Coin Analysis. Sci journal **2**(1), 8 (2020)

[39] Anwar, H., Anwar, S., Zambanini, S., Porikli, F.M.: Deep ancient roman republican coin classification via feature fusion and attention. Pattern Recognition **114**, 107871 (2021)

[40] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 886–893 (2005)

[41] Goodman, M.: Numismatic Photography. Zyrus Press, Newport Beach (2008)

[42] Yang, J., Wang, H., Yuan, J., Li, Y., Liu, J.: Invariant multi-scale descriptor for shape representation, matching and retrieval. Computer Vision and Image Understanding **145**, 43–58 (2016)

[43] Chen, Y.-S., Hsu, W.-H.: A modified fast parallel algorithm for thinning digital

patterns. Pattern Recognition Letters **7**(2), 99–106 (1988) https://doi.org/10.1016/0167-8655(88)90124-9

[44] Vonikakis, V., Chrysostomou, D., Kouskouridas, R., Gasteratos, A.: A biologically inspired scale-space for illumination invariant feature detection. Measurement Science and Technology **24**(7), 074024 (2013) https://doi.org/10.1088/0957-0233/24/7/074024

[45] Kanimozhi, M., Sudhakar, M.S.: Octagonal lattice-based triangulated shape descriptor engaging second-order derivatives supplementing image retrieval. Journal of Visual Communication and Image Representation **98**, 104005 (2024)

[46] Priyanka, S., Sudhakar, M.S.: Tetrakis square tiling-based triangulated feature descriptor aiding shape retrieval. Digital Signal Processing **79**, 125–135 (2018)