University of Amsterdam

Automatic Egyptian Hieroglyph Recognition by Retrieving Images as Texts

Thesis report in completion of the master in Artificial Intelligence

Author:
Morris Franken

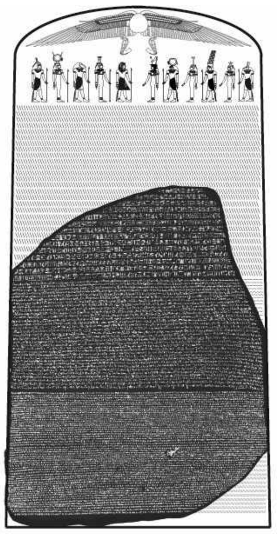
Supervisor:
Dr. Jan van Gemert

August 23, 2013

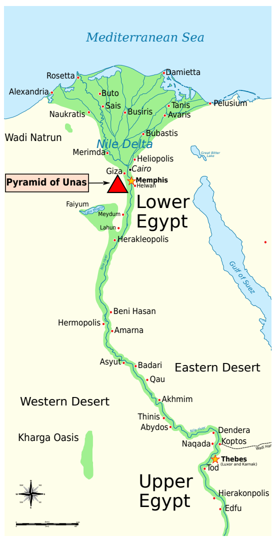
Contents

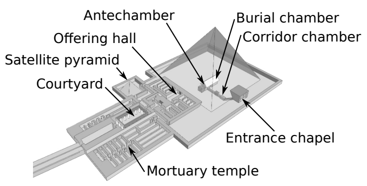
Introduction 1

THE ancient Egyptian hieroglyphs have always been a mysterious writing system as their meaning was completely lost in the 4th century AD. The discovery of the Rosetta stone in 1799 allowed researchers to investigate the hieroglyphs, but it wasnt until 1822 when Thomas Young and Jean-François Champollion discovered that these hieroglyphs dont resemble a word, but each hieroglyph resembles a sound and multiple hieroglyphs form a word. Their work has had a revolutionary impact on the deciphering of ancient Egyptian hieroglyphs. The ability to understand hieroglyphs has uncovered much of the history, customs and culture of Egypts ancient past. It is these ancient hieroglyphs that tell the tales of the otherwise long forgotten Pharaohs. Their achievements and victories on the battlefields have all been stored within these hieroglyphs.



(a)





(c)

(b)

(d)

Figure 1.1: (a) One possible reconstruction of the original stele of the Rosetta stone. (b) Map of ancient Egypt, the pyramid icon indicates the pyramid of Unas. (c) Schematic drawing of what the pyramid once used to look like. (d) Current state of the pyramid of Unas. *Images courtesy of Wikipedia, creative commons license.*

Much of the knowledge about the ancient Egyptian civilization would have been hidden if it wasn't for the Rosetta stone which was the key for deciphering the hieroglyphs. The Rosetta stone was originally inscribed in 196 BC to acknowledge the accomplishments of the Pharaoh *Ptolemy V* and to reassure the Egyptians that tax money was well spent. The Rosetta stone offers three versions of the same text: the top is written in ancient Egyptian hieroglyphs, the middle part is written in Demotic (derived from the ancient Egyptian hieroglyphic script, but used for writing on documents) and the bottom is written in ancient Greek. The stone was re-discovered by Napoleons army on a campaign through Egypt.

The ancient Egyptian civilization lived through three main stable periods known as the *Old Kingdom*(2686-2181 BC), *Middle Kingdom*(2055-1650 BC) and the *New Kingdom*(1550-1069 BC). They were separated by relatively unstable periods (intermediate periods). During the Old Kingdom, the Egyptian art flourished and it was during this time that the Great pyramids of Giza were constructed as a burial tomb for the Pharaoh Khufu. It was also in the same period that the pyramid of Unas was build, which plays a mayor role throughout this study (see figure 1.1). Between 2200 and 2150 BC Egypt was hit by a severe drought period which contributed to the collapse of the Old Kingdom. After the collapse, the Egyptians entered the first Intermediate period, which is often described as a period of chaos and disorder. More than a century after the collapse, a Pharaoh by the name of Mentuhotep II restored order and marked the beginning of the Middle Kingdom. During the Middle Kingdom the Egyptians expanded their empire and developed an understanding of basic mathematics, such as calculating the area of a circle or the volume of a pyramid. With the death of Queen Sobekneferu came an end to the Middle Kingdom. In the consecutive years Egypt was ruled by many different Pharaohs in a relative short period of time. The second intermediate period came to an end when Ahmose I ascended to the throne and led Egypt to what is known as the New Kingdom. It is thought to be Egypts most prosperous era and marked the peak of its power. The New Kingdom brought forth some of the most famous Pharaohs, including the controversial Pharaoh Akhenaten who abandoned the traditional Egyptian religion and started worshipping a god by the name of Aten. Akhenatens son Tutankhamun who became famous since the discovery of his grave in 1922 by Howard Carter. Ramesses II (Ramesses the Great), who lived approximately 90 years, during his lifetime he founded the city of Pi-Ramesses. He fought many battles and triumphed over Egypts archenemy the Nubian and Hittite empires. Less than 150 years after the death of Ramesses II, Egypt suffered from internal conflicts as well as new enemies invading Egypt. This meant the end of the New Kingdom known as the golden era of Egypt.

Since the deciphering of the ancient Egyptian hieroglyphs, much of their culture has been uncovered. By now, most of the ancient texts have been translated, but this does not put an end to the mystic air revolving around the ancient Egyptian culture. In fact, many movies and games only fuel the thought of Egypts magical powers and mummies. To this day only a few people are capable of reading the ancient Egyptian hieroglyphs, but with the aid of current technology it is possible to provide others, like tourists, the ability to uncover the mysteries of ancient texts themselves. This can be made in the form of an App that allows tourists to receive the translation and other background information of a given text, simply by taking a picture.

The aim for this thesis is to develop the essential first step; the creation of a system that is able to automatically recognize ancient Egyptian hieroglyphs from photographs. I have subsequently evaluated 5 visual descriptors in 3 different matching schemes on a newly created Egyptian dataset. In addition to visual-only cues, I made use of a corpus of Egyptian texts to learn language models that help re-rank the visual output.

Because the hieroglyphic writing system has been used for over 4 millennia, many regional and temporal dialects existed. Therefore I chose to focus my thesis on the hieroglyphs found in one pyramid to eliminate issues with different writing styles. The pyramid of Unas was a perfect

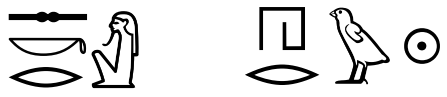

(a) (the god) Sokar (b) Day

Figure 1.2: Use of determinants

candidate because all the pyramid texts have been carefully documented and photographed. The pyramid itself was built in the fifth dynasty as a burial place for the Pharaoh *Unas* who ruled over Egypt from 2375 BC to 2345 BC. It is located just south of the city of Giza.

One of first things to notice about the ancient Egyptian writing system is that it does not contain any spaces. The lack of separation between words makes it difficult for the inexperienced eye to find out where one word ends and the other begins. The relation between verbal and written text is different from current day English. Each word can be written in a number of different ways as long as the pronunciation remains the same. This can be compared by writing *bare* when **bear** is meant. However some of the hieroglyphs do not have any sound related to them and are used to illustrate the meaning of the word, these are called determinative hieroglyphs. An example of this can be seen in figure 1.2a showing the name *Sokar* with an added glyph of a deity to emphasize that *Sokar* is a god. Another example can be seen in figure 1.2b, representing the word *day* with an added glyph of the sun. Furthermore, it was allowed to combine different hieroglyphs in case the writer was lacking space. All these conditions present severe challenges to automatic hieroglyph recognition.

Many different Computer Vision techniques have been developed, while most of them have been created for a specific domain. The hieroglyphs that I'm interested in can be categorized as a mix between handwritten characters and images of animals and objects known to the ancient Egyptians. Therefore I make use of those Computer Vision techniques that were originally developed for image recognition or handwriting recognition.

This study has 3 contributions, the first contribution is the introduction of a hieroglyph dataset, containing nearly 4000 images of ancient Egyptian hieroglyphs manually annotated from photos of the hieroglyphs texts inside the pyramid of Unas. Secondly are the results of various known Computer Vision algorithms on this dataset, and finally are the results combined with a language model in order to refine the classification results. The entire process is illustrated in figure 1.3.

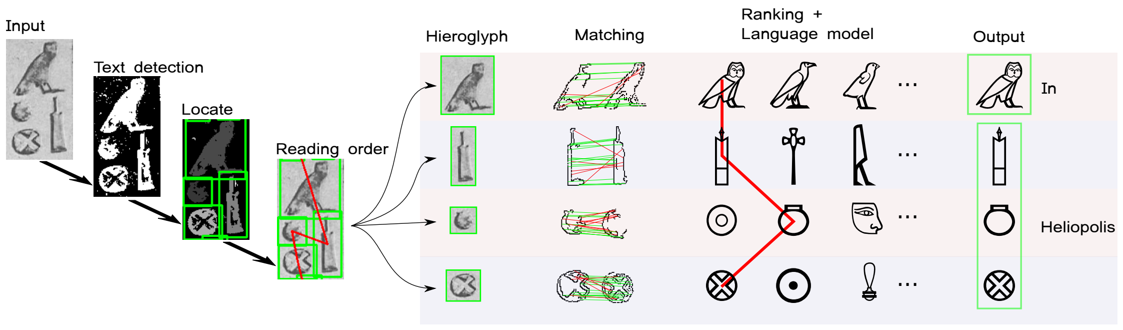


Figure 1.3: Pipeline for hieroglyph recognition. The 3rd output hieroglyph from the top is corrected by the language model in order to find the word Heliopolis (birth-place of Unas).

Computer Vision

Computer Vision is a field in Artificial Intelligence with the aim of using computerized software for analysing and understanding images. Unlike our biological ability to analyse visual input effortlessly, computers have trouble gaining a deep understanding of these images. It is not so strange that computers have trouble grasping the essence of an image, since to a computer, an image is just a bunch of numbers. Over the years researchers have investigated how the brain manages to analyse visual input so efficiently, and sought for ways to imitate its behaviour.

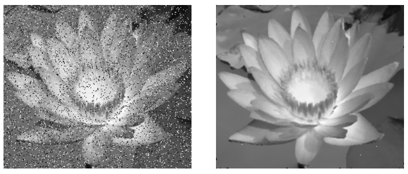
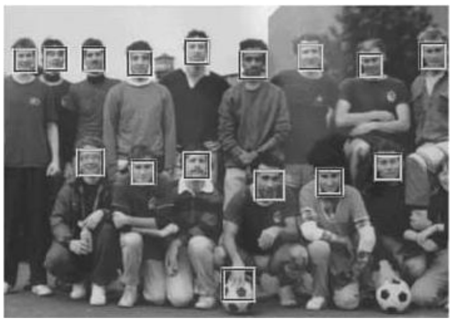


Image processing

Image segmentation [20]

Computer Vision is a broad term that includes:

- **Image processing:** applying certain operations on the image such as sharpening the image, changing the contrast or removing noise from an image.
- **Image segmentation:** assign each pixel in an image with a global label, for example the sky/water/grass/building/person.
- **Object detection:** detect a certain type of object in an image such as faces or written characters.
- **Image recognition:** the ability to classify the main object in an image.



Object(Face) detection [35]

Image recognition [9]

Although there is still a lot of ground to cover in the field of Computer Vision, many Computer Vision techniques have already been applied to real world applications with much success. For example, 3D modelling to reconstruct 3D buildings from images. Face detection and Face recognition which is used for many applications including autonomous surveillance. Automatic character recognition for recognizing postal codes on letters or number plates on cars. Recently Microsoft introduced an application named Bing Translator, it is a smart phone App that can read text in an image and is able to translate it to one of the other 45 languages.

Related work 2

Multimedia tools have aided preservation, analysis and study of cultural, historical and artistic content. For example, the digital Michelangelo project [22] created high quality 3D models of Michelangelo's sculptures and architecture. Furthermore, wavelet analysis of brush strokes in paintings can reveal artist identity [14], image composition has shown to aid category labeling [12] and photographs can be classified as memorable or not [13]. In this thesis I follow in these footsteps, and propose a new dataset, and a multimodal (visual and textual) approach for automatic Egyptian hieroglyphs recognition.

Automatic Optical Character Recognition (OCR) has been a long term goal in Computer Vision, and is currently at a state where it can be considered achieved. Current OCR systems have been applied to real world applications, such as postal automatic postal code recognition on letters [21], or number plate recognition [28]. Other applications of OCR have been utilized, such as document scanning to digitalize books, a prominent example of this is the Tesseract open-source OCR engine that was initially developed by HP [32]. Another example is proposed by Karaoglu et al. (2012) [16] where text recognition is used to aid in object recognition tasks.

Related work on automatic hieroglyph recognition focuses on Mesoamerican culture, and in particular on the ancient Maya hieroglyphs [10, 29, 30]. To this end, the HOOSC descriptor was developed [30], which is a combination of HOG [5] and the Shape-Context [1]. The HOOSC descriptor offers a robust approach in object recognition by adding contextual information to the HOG descriptor based on the shape context. The authors of the HOOSC descriptor trained their system on two small datasets, and reported mean average precision of 0.39, which is an improvement of 21.1% compared to the Shape Context algorithm on the same datasets. Such descriptors can be used for direct matching [29] or with a bag-of-words (BOW) approach [30]. Other work extracts detailed line segments for Maya hieroglyph matching [10]. In all these works the hieroglyphs are typically manually extracted and individually digitized. In contrast, the photographs of Egyptian hieroglyphs consists of noisy plates, which each typically contain around 400 hieroglyphs (see figure 3.2). Moreover, the Maya culture used a different type of hieroglyphs and I therefore evaluate the HOOSC and other descriptors on Egyptian hieroglyphs.

More work has been conducted on automatic character recognition for other writing systems such as the Chinese or Hangul script. Much research on Chinese character recognition is currently ongoing, with an annual *Chinese character recognition contest* [23] where contestants are presented a dataset to train their systems, in the end they are evaluated on an unseen dataset. The best system in 2011 achieved a 92.18% correct classification rate for for offline character recognition, and 95.77% for online character recognition. The most successful method for the offline character recognition was proposed by The Dalle Molle Institute for Artificial Intelligence

(IDSIA) [4, 3]. It uses a CNN (Convolutional Neural Network) to classify the characters. The most successful method for the online character recognition was a system proposed by Zsolt Wimmer. It makes use of a MLP (Multilayer Perceptron) enhanced with a tri-gram language model.

Unlike the Chinese script, the Hangul script is a phonetic script, and is currently used by the Koreans. Hangul character recognition has been performed by Kim and Kim (2001) [17], where Hierarchical Random Graphs are used to recognize the Hangul characters. The authors reported a 90.4% correct classification rate on a restricted dataset with 520 syllables.

Improving the recognition rates is not only achieved by improving the Optical Character Recognition (OCR) method itself, but can also be achieved by looking at what the recognized characters represent. In 1996 Tong and Evans [34] proposed a system to enhance recognition rates on handwritten symbols with a statistical language model based on n-grams. It computes the probability of inserting, deleting or replacing characters proposed by the Optical Character Recognition (OCR) system. These probabilities for each n-gram are extracted from a database containing around 100.000 words. The authors reported a 60.2% error reduction.

Aside from the Optical Character Recognition, statistical language models play an important role in Automatic Speech Recognition (ASR). Magdin and Jiang (2009) [25] have proposed a discriminative training method based on n-grams that manages to reduce the error rate by 3% SPINE1 speech recognition task. The method formulates a discriminative object function of all parameters of the n-gram language model.

Current work on automatic scene text detection and recognition [8, 16, 18] are typically hand-tuned to the specific western or Chinese characters at hand which are quite different from Egyptian hieroglyphs. In this work, I will draw inspiration from text detection to localize the hieroglyphs and use generic image descriptors for the recognition.

The dataset 3

The dataset is composed of two parts, the first part is the visual component which contains photographs of pyramid walls, each containing a large amount of hieroglyphs which were used to train and test the different classification methods. The second part is the textual component which contains a lexicon of ancient Egyptian words as well as the pyramid and coffin texts written in transliteration, these are used to train a language model.

3.1 Images

There are sixty-six grey-scale photos taken of walls inside the pyramid of Unas. The photo's are taken from the book *The Pyramid of Unas* by Alexandre Piankoff [27]. Each photograph contains around 400 hieroglyphs, I selected ten of these photographs for this database. The hieroglyphs within the pyramid of Unas are written in columns such that the proper reading order is from top to bottom. This is not always the case, the ancient Egyptians allowed the hieroglyphs to be written in 3 directions, either from left to right, right to left and from top to bottom. The only indication of the correct reading order is the direction in which the animals or humans are facing. The rule of thumb is that these glyphs will always face the beginning of a line unless the hieroglyphs are written from top to bottom, in which case vertical columns will indicate the text should be read from top to down. Moreover, multiple hieroglyphs can be placed next to each other in a single column which should be read by treating it as if it is a miniature horizontal line. An example of a single column can be seen in figure 3.2, with its translation below. From this point on I will distinguish the word "picture" and "image", where I will use picture when I mean a photograph containing multiple hieroglyphs, and when I write image, I mean an image containing a single hieroglyph.

The hieroglyphs in the photographs were cut out and labelled according the Gardiner Sign list [11], where each unique hieroglyph is labelled with an alphabetic character followed by a number.

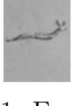
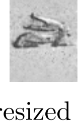
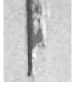
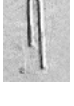
Gardiner label	I9	G17	E34	M17	S29
Image					

Figure 3.1: Examples of resized images.

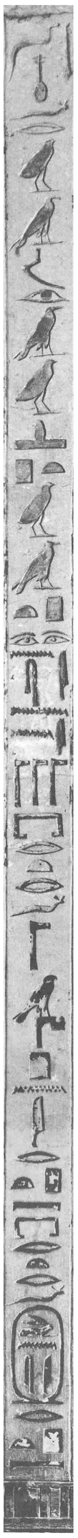


Figure 3.2

The alphabetic character represents the class of the hieroglyph, for example, class 'A' contains hieroglyphs about a man and his occupations while class 'G' contains hieroglyphs of birds. The entire Gardiner Sign list consist of more than 1000 hieroglyphs in 25 classes. Furthermore, the images were resized to 50x75 pixels. The reason to resize all images of hieroglyphs to a uniform size was to make the system scale-invariant. The images are not simply resized as this would induce a loss in heigh/width information from the hieroglyph. Instead the images were resized in such a way that they would fit in a 50x75 image while preserving the hight/width ratio of the hieroglyph. The remaining unfilled background was automatically generated to mimic the background texture from the image. More over resizing the images and generating the background is explained in section 4.2. A few examples of hieroglyphic images with their Gardiner label are shown in figure 3.1.

“. . .
he shall not write with his little finger. How beautiful is indeed the sight, how good indeed to see, so say they, so say the gods, (when) this god ascends to heaven, (when) Unas ascends to heaven while his power (bA.w) is over him
. . .

The English translation of column 476 (see figure 3.2).
Translated by J.D. Degreef and R.O. Faulkner.”

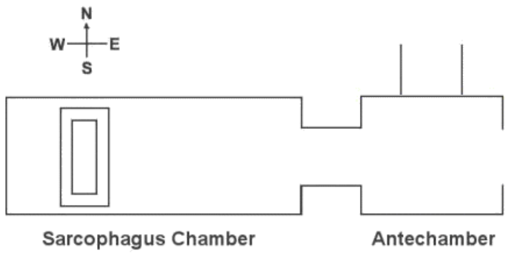


Figure 3.3: Map of the Pyramid of Unas *Image from www.pyramidtextsonline.com/plan.html*

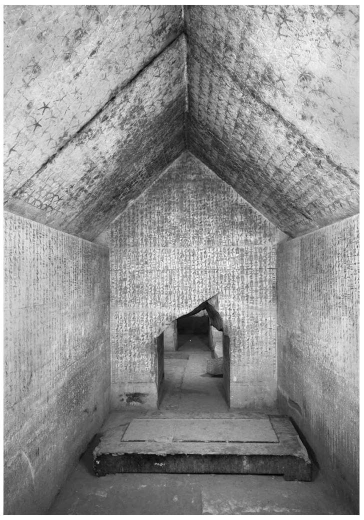


Figure 3.4: Photograph of the sarcophagus chamber *Image from www.pyramidtextsonline.com/platevi.html*

Figure 3.3 shows the map of the sarcophagus chamber and the ante chamber within the pyramid of Unas. Figure 3.4 shows a peek inside the sarcophagus chamber. The inscriptions on the walls in the pyramid of Unas are the oldest known pyramid texts, the purpose of these texts were to describe the life of Unas with all of his achievements, as well as to guide Unas on his journey to heaven.

3.2 Text

Two different methods are used to refine the classification by means of a language model, first is the lexicon-based method which searches for words in the proposed classification, the other method is based on n-grams, where the classification results are refined according the probability of certain hieroglyphs occurring in a sequence.

For the first method a lexicon is used which holds over 12.000 ancient Egyptian words along with their translation, transliteration and the hieroglyphs used to write the words. For the n-grams I made use of a corpus from JSesh[1], containing 158 pyramid texts, all written in a mixture of transliteration (which is the language used to map hieroglyphic texts to a computer readable format) and Gardiner labels. Once the texts are converted to only Gardiner labels, they can be used to find the occurrence probability of n-grams as well as determining the occurrence probability of a word in combination with the lexicon.

I should mention that the JSesh dataset was not the first choice, in fact a dataset provided by Dr. H. Hays contained 444 coffin texts and 783 pyramid texts. The problem was that this dataset is written in a transliteration that is different from the lexicon I possessed, and the lexicon provided with this dataset does not contain any information about the hieroglyphs used to write the words. Therefore it was impossible to transform the texts into the Gardiner format which I'm interested in. The JSesh dataset did allow me to transform the texts into the Gardiner format, which is the reason for choosing the JSesh dataset.

A representation of the JSesh dataset can be seen in table 3.1, it depicts a part of the *Second Semneh stela under Sesostris III*, which I can only describe as an inspirational speech from the Pharaoh to it's successor(s). The JSesh format contains not only a mix between Gardiner labels and transliteration but also contains information on how the hieroglyphs are positioned or rotated. Although this information can be valuable in a latter stage of the project, it is not useful for training the language model.

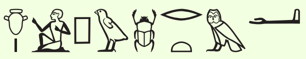
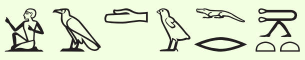
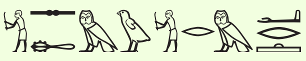
JSesh Format	Hieroglyphs	Translation
ib:Z1-A1-.:p\430:.-w-xpr-r:t-m-a:.-!		. . . I am a king who speaks and acts. I make happen what I conceive, eager to seize, hasty to succeed, in whose heart a matter doesn't slumber, . . .
A1-A-d:.-w-I3:r-V15:t*t-!		
A24-z:X-m-w-A24-r-m-a:r:Y2-!		

Table 3.1: Part of the Second Semneh stela under Sesostris III, translated by M.J. Nederhof.

[1]Open source hieroglyph text editor

Locating Hieroglyphs 4

LOCATING hieroglyphs is done both manually and automatically. The manually annotated hieroglyphs are used in the train set and as a ground truth for evaluating the automatic hieroglyph detection system.

4.1 Manual detection

To create the dataset containing so called perfect images of hieroglyphs (images portraying exactly one hieroglyph), a large number of hieroglyph had to be annotated manually. To make this task slightly more manageable I developed a program that allows for easy annotating of hieroglyphs.

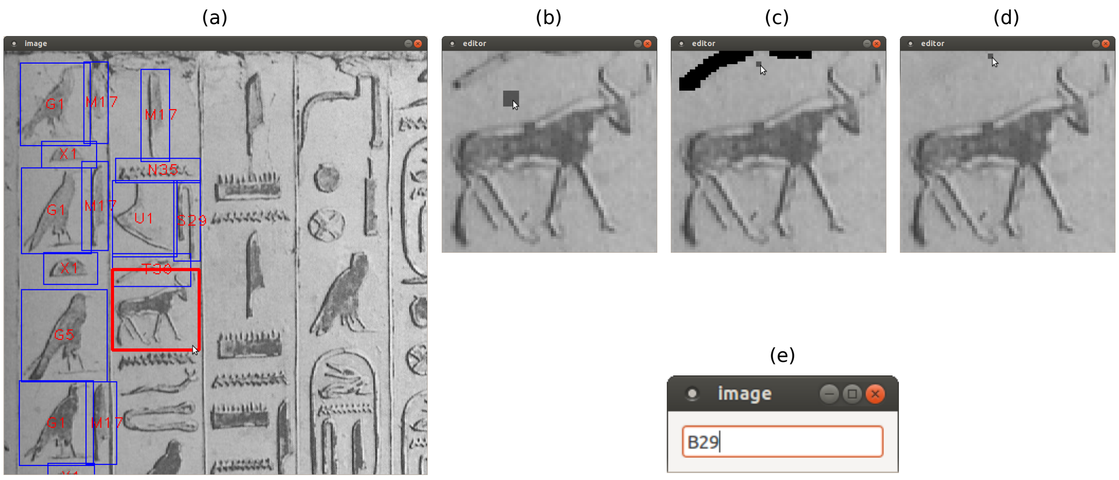


Figure 4.1: Manually annotating hieroglyphs

Figure 4.1 depicts the process of annotating the hieroglyphs. Previously annotated hieroglyphs are shown as a blue rectangle, while the current selected hieroglyph is indicated by a thick red rectangle (a). Once the hieroglyph is selected, it is possible to remove any unwanted noise or parts of other hieroglyphs that are not relevant for the main hieroglyph. This can be done in the editor window (b), which shows an enlarged view of the selected hieroglyph. A simple tool allows the user to select parts that should be removed (c) and when pressing the space bar the selected parts are filled with background texture (d). The last step is to actually label the hieroglyph with the correct Gardiner label (e).

4.2 Generating background

Once the hieroglyphs are located with a bounding box, they are cut out and saved as a individual image such that the classification algorithm can proceed to classify them. However, when looking at the bounding boxes it becomes apparent that some of them are overlapping, which causes parts of other hieroglyphs to exist in the same image. This can confuse the classification algorithm since it's only looking for a single hieroglyph in an image. To solve this, parts that do not belong to the main hieroglyph are removed and a background generator will attempt to fill these pixels with the appropriate colour such that they blend into the background. For the background generator I made use of the Texture Synthesis method proposed by Efros and Leung (1999) [7]. The Texture Synthesis works by filling the gaps layer after layer from the outside in. The surrounding of each to be generated pixel is compared to the surrounding of every other pixel in the image, the gray value of the most similar surrounding of a pixel is picked based on the least SSD (Sum Squared Distance). Although the approach works well to remove other objects, it is a rather cpu-intensive method, and in the light of creating an App for a smart-phone that can detect the hieroglyphs, this could become a major problem. Therefore I implemented a faster approximation to the Texture Synthesis method which is about 400 time faster and simply takes a random grey value from the 20 nearest pixels. Once all gaps are filled with the background texture, a blur-filter will smooth the generated pixel such that they blend in to the background. The Result for both methods is shown in figure 4.2. Although the Texture Synthesis method generates an almost indistinguishable background, it is rather unnecessary to have that level of cloaking. The approximation is sufficient in masking the undesired parts in the image such that the classification algorithm is able to train on them.

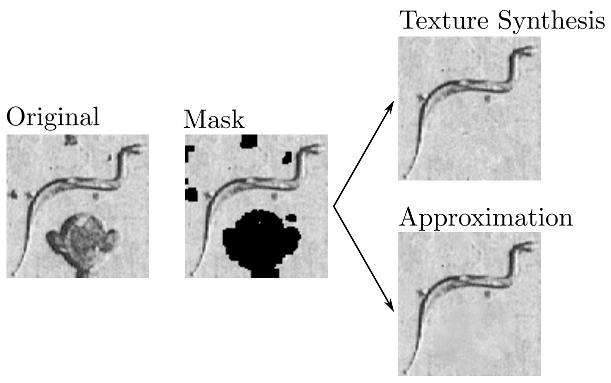


Figure 4.2: Removing other objects from the bounding box

Furthermore, images are resized to a 50x75 format. Resizing the images is important since some methods require the images to have the same size. However, simply resizing the images would induce a loss of width/height information which is undesired since the height/width information is often important as it distinguishes an egg from a circle. Another possibility is to resize the image to fit the desired size and fill the empty space with a certain grey-value. This would work, although the images won't look natural and it might disturb the edge detector. To solve this problem I made use of a variation of the same background generator used to remove unwanted objects from images. It attempts to mimic the background found within the image by detecting the actual background in the image followed by copying these background colours to nearby empty pixels. This first step is to create a mask of the image which extracts the foreground from the background. An important observation is that the hieroglyphs are always located in the center of the image, which means that the edges of the images most likely contain background texture. I make use of this by computing the the median grey-value at the edges of an image. A mask of the entire image is created where each pixel within a certain distance of the median value is labelled as background, whereas others are labelled as foreground (see figure 4.3).

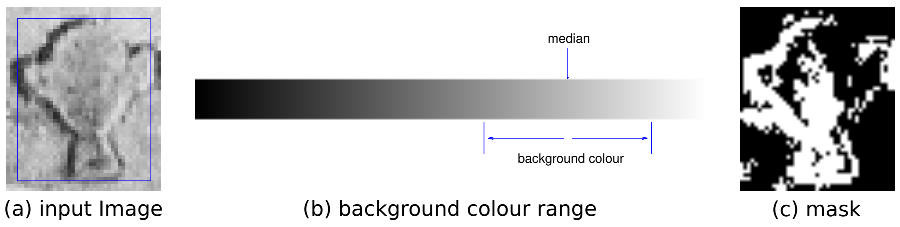


Figure 4.3: Computing the background mask

An overview of the algorithm which generates the background is shown in figure 4.4. First, pixels that have been labelled as foreground in the mask are removed (b). For each to be generated pixel, it locates the closest pixel in the image, from there on it will create a search window around this pixel where the size of the window is determined by the distance to this pixel (c). A random pixel is chosen from this search window (d), if the pixel is labelled as background the colour is copied, if the pixel is labelled as foreground, the algorithm increases the size of the search window and tries again. Finally, the generated pixels are smoothed, but the smoothing factor is determined by the distance to the image, such that pixels further away from the image are smoothed more than those who are close to the image. This will preserve local variations and noise in the image and mimics those in the generated part of the image. At the same time, the further the pixel is from the image the more it will be smoothed such that it does not introduce much noise in the generated part and produces a more monotone colour at the edges, see figure 4.5 for the results after generating the background.

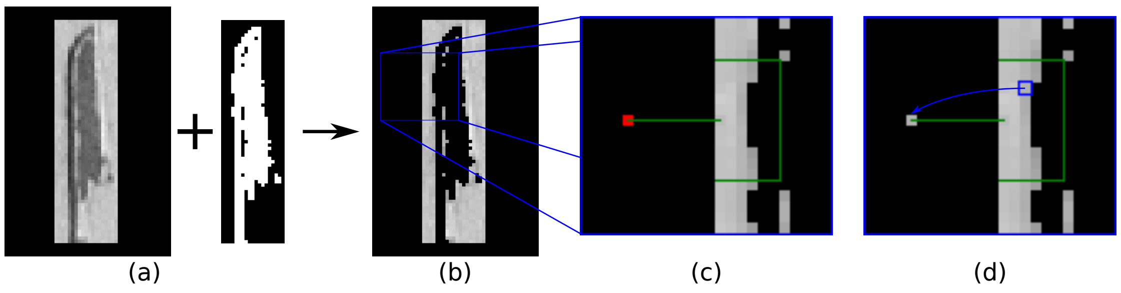


Figure 4.4: Choosing the right color for each pixel

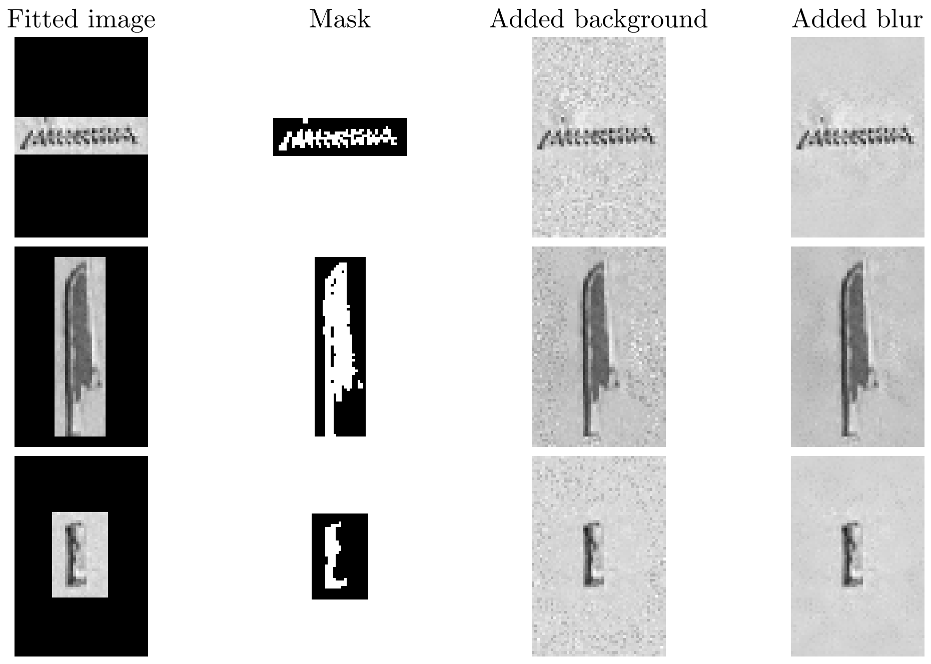


Figure 4.5: Examples with generated background

4.3 Automatic detection

For the automatic hieroglyph detection system I made use of a text-detection algorithm proposed by Karaoglu et al. (2012) [16]. The algorithm was originally designed for text detection in natural images, but can also be used for object detection. It makes use of a saliency detector, which is used to find strong curves in the image that could indicate the presence of text. When applied to the a picture containing hieroglyphs, it produces a binary picture which highlights the pixels with a high likelihood of being part of a hieroglyph (see figure 4.6b). In order to locate the hieroglyphs, neighbouring pixels are grouped to form objects in a picture. Small objects are possibly combined with the nearest larger objects depending on the distance and size. The result is an unordered list of objects which are possibly hieroglyphs. After applying a noise filter which removes small objects and cartouches (frame around the name of a Pharaoh), a list of possible hieroglyphs remains as illustrated in figure 4.6c.

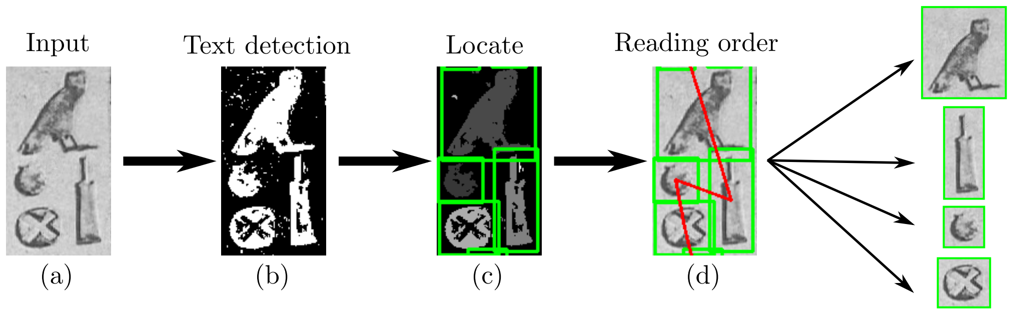
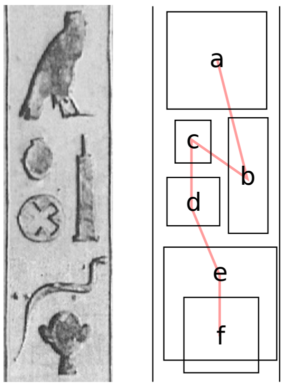


Figure 4.6: Process of locating hieroglyphs on an image.

Once all the objects have been detected in the picture, they are sorted by their reading order such that the language model can find the words in a given text. To this end I created a system that is able to determine the correct reading order of the hieroglyphs based on their position. The first step in this method is to find the columns along which the hieroglyphs are written. These can be located by applying a line filter on the picture to extract the vertical lines. Lines that are longer than a certain threshold value are used as column separator. The algorithm will proceed to sort the hieroglyphs by their reading order based one a few simple rules. These rules distinguish horizontally- and vertically aligned hieroglyphs, for horizontally aligned hieroglyphs the x values are used to sort them, while for vertically aligned hieroglyphs the y values are used (see figure 4.7 for a more detailed explanation of this process).



$a \to b$	No parts of a and b are overlapping, therefore they are ranked according their y-values (top to bottom).
$b \to c$	The center of c lies within the y-boundaries of b, therefore they are horizontally aligned. This means that b and c are ranked according their x-values (right to left).
$c \to d$	Same as for $a \to b$
$d \to e$	Same as for $a \to b$
$e \to f$	e is completely overshadowing f in the x-dimension, therefore they are not horizontally aligned and will also use the y-values to sort them.

Figure 4.7: Determining the order in which the hieroglyphs should be read

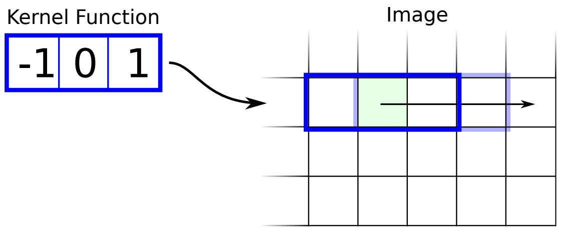
Classifying Hieroglyphs 5

THE main part of the thesis was to classify the hieroglyphs. Therefore I've implemented a range of different methods which have their origin in either image recognition or handwriting recognition. Classifying the hieroglyphs goes through two stages after the images have been preprocessed, first the descriptors of the images are computed, and in the second stage the descriptors are used to compare images to each other and to rank them according their similarity.

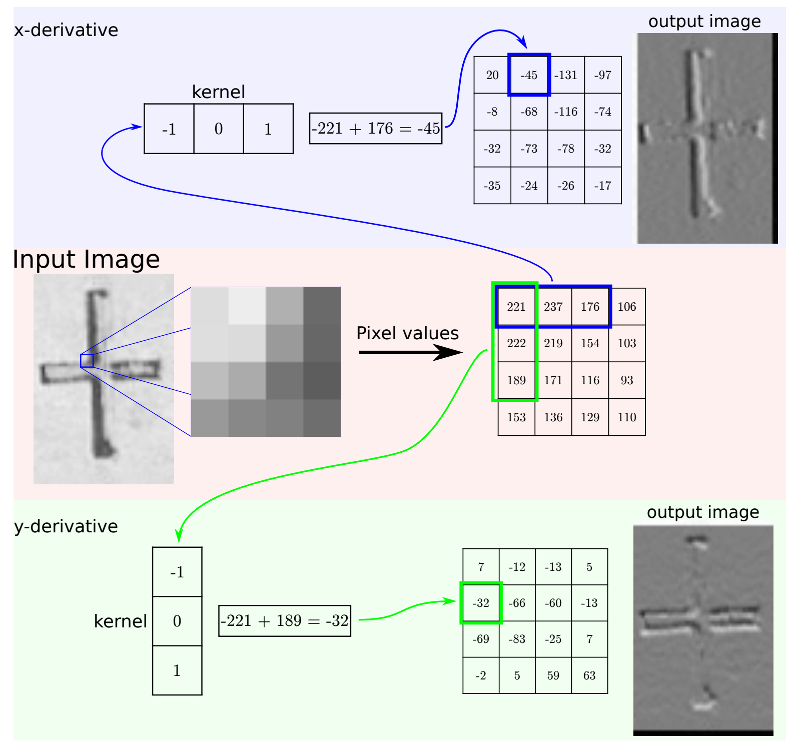
Some of the following methods make use of image derivatives. The derivative of an image can be used to find variations in an image and can therefore be used to find edges and to determine the image gradient.

Image derivative

Computing the image derivative is a standard Computer Vision practise, and is used to determine changes in either the x or the y direction of the image. Large changes can indicate the presence of edges, which is why it is used in edge-detection algorithms or to determine the gradient in an image. It works by convolving the image with a certain kernel function, convolving an image with a kernel function is simply sliding the kernel function (aka. sliding window) over the image, and multiplying the kernel values with the pixel values, the sum of all these multiplications is taken as the new pixel value. An example is shown in the figure below.



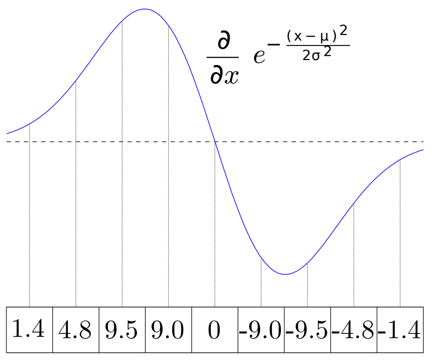
The most simple kernel function would be $[-1, 0, 1]$ which translates to the difference between the next pixel and the previous pixel. By convolving this kernel function with the image, the derivative in the x direction is computed. While the transposed version $[-1, 0, 1]^T$ would result in the image derivative in the y direction.



More advanced kernels take more pixels in to consideration, and are typically a the first derivative of a Gaussian function or a Sobel operator:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel operator



Resulting kernel of Gaussian derivative ($\times 10^{-2}$)

The difference between the various kernel function lies in how much of their surrounding is taken to compute the pixel gradient. The size of the kernel function should reflect the scale of the interesting edges. A large kernel will find thick edges but could miss small changes in the image that could be important, while a small kernel will potentially label both sides of a thick line as a separate edge.

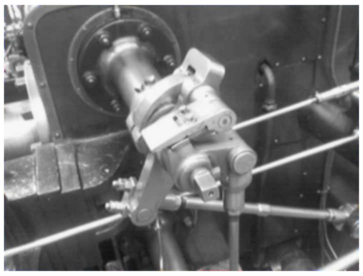
5.1 Descriptors

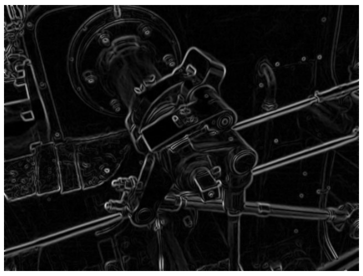
A popular Computer Vision method for comparing images is to make use of local patches (small parts of an image). The aim is to compare patches to each other, and patches of similar objects should result in a high similarity score. The difficulty lies in matching patches of objects that have been slightly rotated, skewed or zoomed in/out. Not to mention the different light conditions under which the photos are taken. The task of the descriptor algorithm is to formulate the descriptor in such a way that it is capable of dealing with these issues.

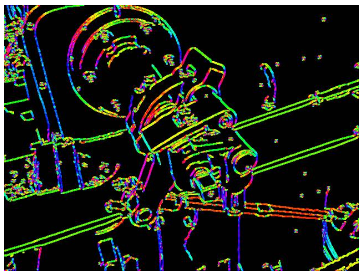
The location of the patches must be carefully selected to capture the most important aspects of an image. Methods such as the Harris affine region detector [26] are specialised in finding the most interesting locations in an image. For this study however, I used the Canny-edge detector which finds the edges in an image. I chose to use the Canny-edge detector since the edges are the most important features of a hieroglyph.

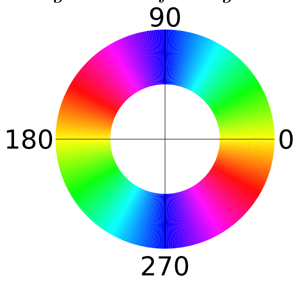
Canny-Edge detector

The Canny-edge [2] detector algorithm was proposed by John F. Canny and is designed to extract the edges in an image and thereby to create an outline of the image. The first step in this algorithm is to compute the $magnitude = \sqrt{G_x^2 + G_y^2}$ and the angle of the gradient with $direction = arctan\left(\frac{G_y}{G_x}\right)$. Where G_x and G_y are the image derivatives computed by the Sobel edge detector in the x and y direction respectively. What follows is the *non-maximum suppression*, where non local-maximum pixels are removed. This works by comparing the magnitude response for the horizontal, vertical and both diagonal directions. An edge can only head in on of those 4 directions, the responses for the non-maximum directions are suppressed. All pixels that exceed a certain threshold are now considered part of an edge. After the *non-maximum suppression*, only a set of thin-edges remains with possible gaps in between them. These gaps are fixed in the *thresholding with hysteresis* step, where the objective is to grow broken edges. Growing an edge is done by looking at the direction of the edge, if pixels at the end of the edge exceed a second threshold they are added to the edge. This process is repeated until there are no edges left to add.


Original image

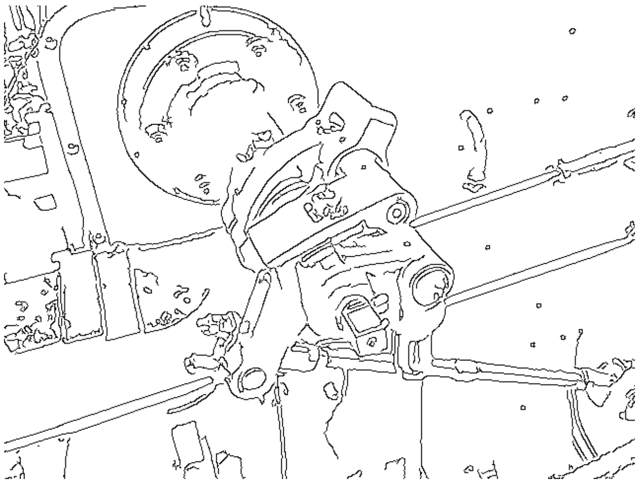

Magnitude of the gradient


Line direction



Colour code

The threshold value must be chosen with caution, as a high threshold value will make the detector miss important edges, while a low threshold will make the detector identify noise as being an edge.


Canny-edge detector result

The objective is to compare two images with each other to obtain a similarity score. Therefore the Canny-edge detector is applied to both input images and the edge-pixels (pixels that are identified as part of an edge) are used as patch-locations. The next step is to compute a descriptor of the patch. The last step is to use the descriptors to compute a similarity score between two images.

Shape Context Descriptor

The shape context algorithm was first proposed by Belongie et al. (2002) [1]. It was originally designed for recognizing handwritten symbols and other objects by including the contour and contextual information of an object in the descriptor. This information is important since it is characteristic for each symbol. Because the Egyptian hieroglyphs are a type of handwritten symbols it made the shape context an interesting choice to see whether or not it would work on the hieroglyphs.

Once the Canny edge detector is applied to an image to extract the outline of the hieroglyph, a descriptor for all the edge-pixels is computed by placing a log-polar mask around these pixels and count the number of other edge-pixels that fall in each bin (see figure 5.1). This will generate a histogram based descriptor for each edge-pixel.

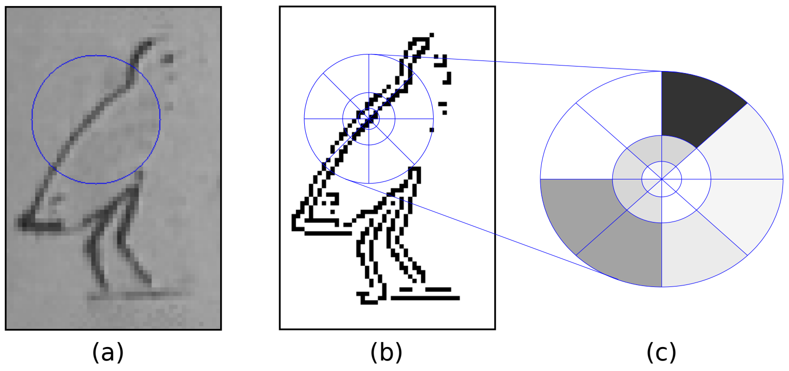


Figure 5.1: Computing the patch descriptor with Shape Context

Figure 5.1 gives an insight in the Shape Context algorithm, 5.1a is the input image where the blue circle indicates the patch area around an edge pixel. 5.1b shows the contour image generated by the Canny edge detector, on top of that the log-polar is shown (in blue) which indicates the bins of the resulting descriptor. Lastly, 5.1c gives an impression of the final Shape Context descriptor where the colour in each bin represents its value.

Self-Similarity Descriptor

Self-Similarity is an algorithm proposed by Shechtman and Irani (2007) [31]. Its roots lie in the object detection and image recognition. It's main objective is to find shapes in an image even though they are suffering from global geometric distortions. The Self-Similarity algorithm attempts to omit these issues by looking at the similarities among parts of an object. Figure 5.2

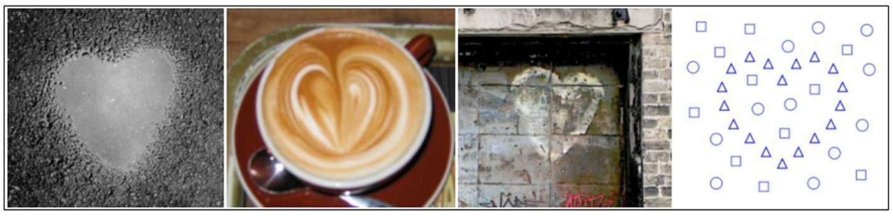


Figure 5.2: Heart shaped objects identified using a Self-Similarity descriptor

shows examples of heart-shaped objects which do not share common image properties (colours, textures, edges), but do share a similar geometric layout of local internal self-similarities

The Self Similarity algorithm computes a correlation surface for each patch, a log-polar histogram is placed on top of the correlation surface and the average correlation for each bin is taken as value for that bin. A simple method to compute this correlation surface would be to take the difference in grey value between 2 pixels. However this method does not capture the surrounding pixels, so therefore the comparison is done on a bounding box around the main pixel as well. Equation 5.1 shows the formula used to create the correlation surface. Hereby the $SSD_q(x, y)$ is the Sum Squared Difference between the main patch q and the patch at x, y. The entire process is captured in figure 5.3 where the small green rectangle indicates the window area of the main patch q, which is then compared to all other window areas to compute the correlation surface. The descriptor is extracted from the correlation surface by taking the average grey value for each bin on a log-polar scale.

$$S_q(x, y) = exp\left(-\frac{SSD_q(x, y)}{max(var_{noise}, var_{auto}(q))}\right) \tag{5.1}$$

The var_{noise} is constant value for the noise, and the $var_{auto}(q)$ is the noise specific for the patch at location x, y. This currently equals the variance of this patch.

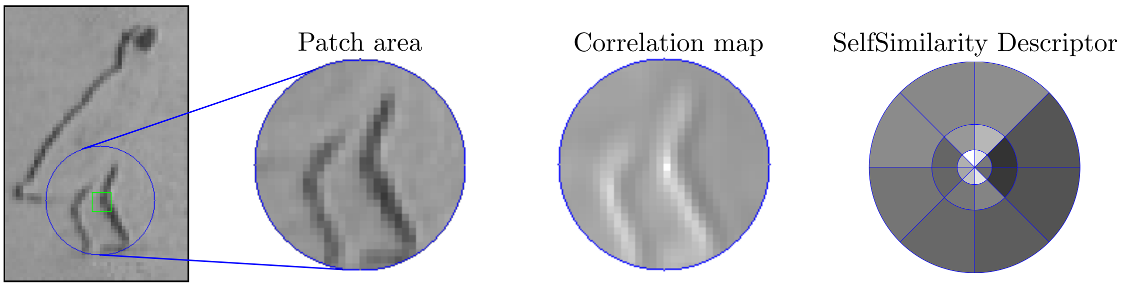


Figure 5.3: The Self-Similarity descriptor

HOG Descriptor

The *Histogram of Orientated Gradients* (HOG) descriptor is a widely used descriptor within the field of Computer Vision. It has proven successful in many domains including human recognition, vehicle recognition (car/bus/bicycles) and animal recognition (dog/cat/cows). The success of the HOG descriptor persuaded me to try it on the Egyptian hieroglyphs.

Histograms of Oriented Gradients

HOG was fist proposed by Dalal and Triggs (2005) [5] to detect human pedestrians in images. The thought behind the HOG descriptor is that local features can be described by a set of edge gradients. In the original implementation of HOG, the image was divided in to a number of equally sized cells (6x6 pixels), the derivative for both the x and the y direction was computed by convolving it with a simple kernel: $[-1, 0, 1]$ and $[-1, 0, 1]^T$ respectively. Although the authors have experimented with other kernels, they found that the most simple and straightforward option works best on their dataset. The HOG descriptor makes use of the gradient and magnitude information of each pixel. These are computed according equation 5.3

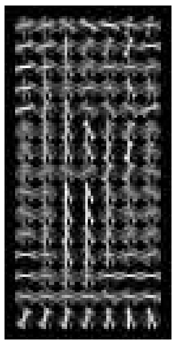
and 5.2 respectively

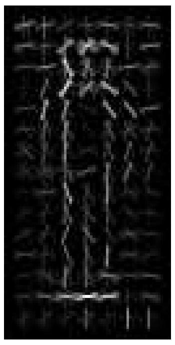
$$Magnitude_{x,y} = \sqrt{(\delta y_{x,y})^2 + (\delta x_{x,y})^2}, \tag{5.2}$$

$$Angle_{x,y} = atan(\delta y_{x,y}, \delta x_{x,y}), \tag{5.3}$$

where $\delta y_{x,y}$ represents the y derivative at location (x, y) in an image. The HOG descriptor of a cell (6x6 pixel sized window) is computed by compressing the gradient and magnitude information in to an nine-bin histogram (from 0 to 360 degrees where each bin ranges 45 degrees). Each pixel in a cell contributes to the histogram by adding its magnitude to the correct bin (based on the gradient value). The entire HOG descriptor is then constructed by combining 9 different cells in a 3x3 grid, this is called the rectangular HOG (R-HOG). The authors have experimented with other shapes such as a circular HOG (C-HOG) but found similar results.
Once the HOG descriptors have been computed, the results are fed to a recognition system such as the Support Vector Machine (SVM). SVM is a classifier that can analyse data and recognize patterns. In this case it is used to classify an input image as being a pedestrian or not, based on the HOG descriptors. The SVM is first trained on a set of labelled data, and computes a certain weight for each entry value. When a new entry arrives, the weighted sum of all entry values and the SVM-weights is computed. If the weighted sum exceeds a certain threshold value, the SVM will classify the instance as being a pedestrian. The figure below vaguely shows the results of the HOG descriptors when applied to the input image. The figure on the right shows the HOG-values multiplied with the SVM-weights, which clearly demonstrates the most important features of a pedestrian.

Input image


HOG descriptors


SVM weights

The authors tested their method on two datasets. The first one being the MIT-pedestrian database which contained 509 trainings images and 200 test images, the HOG descriptor had a nearly zero miss rate (10^{-4}) . The second dataset was developed by the authors themselves to include more difficult images of pedestrians in different poses. The results for the second dataset was a miss rate of roughly 0.1 at 10^{-4} false positive rate.

I implemented the HOG descriptor by dividing each patch in to a 3x3 grid, the magnitude and gradient information is computed for all 9 cells and is stored as a single histogram. An overview of this process is shown in figure 5.5. A complete patch is shown in figure 5.4, which is constructed from 9 cells and 8 bins for each histogram in the cell. The direction of the lines represents the gradient and the length represent the magnitude of that particular bin. The image shows that cells which consists mainly of background pixels have low values for the magnitude, while cells which contain a mixture of background and foreground pixels have a high magnitude.

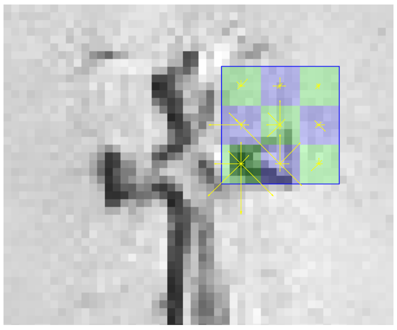


Figure 5.4: HOG descriptor

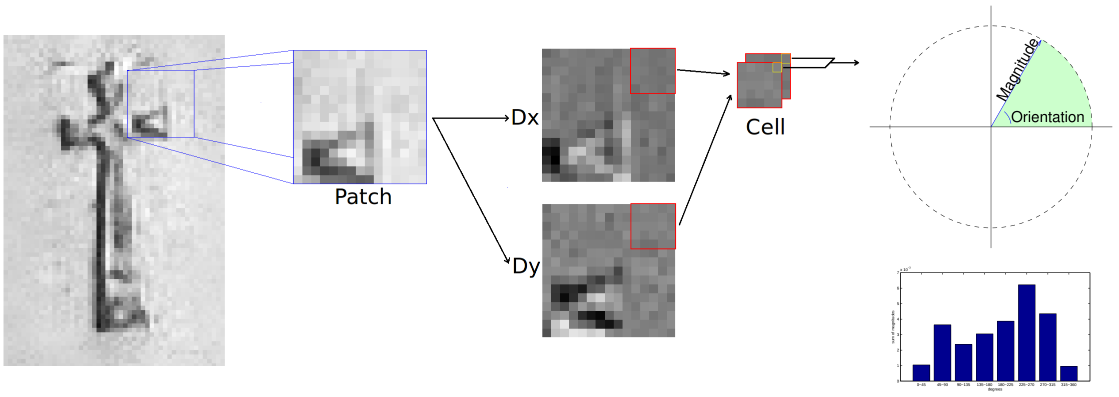


Figure 5.5: HOG pipeline

HOOSC Descriptor

Histogram of Orientation Shape Context (HOOSC) is a method proposed by Roman-Rangel et al. (2011) [30]. The algorithm was initially developed for recognizing Maya hieroglyphs. Although the Maya hieroglyphs are different from the Egyptian hieroglyphs, they may share some of the same characteristics.

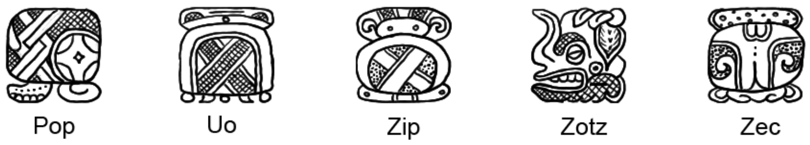


Figure 5.6: A few Maya glyphs taken from the Maya *Haab Calendar*

The Mayan script differs from the Egyptian script mainly because most characters in the Mayan script do not depict any particular object (see figure 5.6), while the Egyptian hieroglyphs depict things they saw in real life such as animals, plant and other objects.

The HOOSC algorithm is a combination of the HOG descriptor and the Shape context descriptor. The HOOSC algorithm takes a patch and its edge-pixels. A small window (5x5) around each edge-pixel is formed and the 8-bin HOG features are computed for each HOG-patch. A log-polar is formed over the entire patch, and the HOOSC descriptor is build up by adding each 8-bin HOG-histogram to the appropriate HOOSC bin. Afterwards the bins are normalized to compensate for the difference in the amount of edge-pixels across the image. An example of this process is shown in figure 5.7.

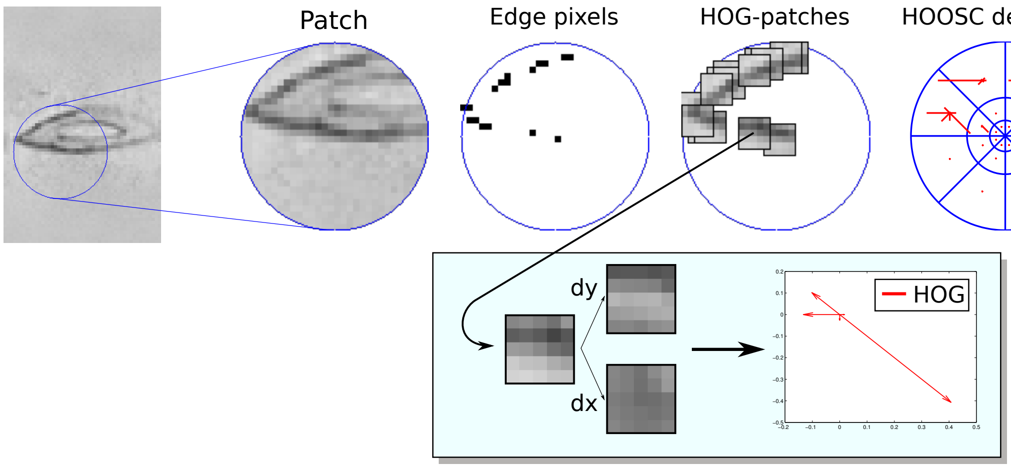


Figure 5.7: HOOSC process

The thought behind combining the HOG descriptor and the Shape context descriptor is to add more contextual information to the HOG descriptor which is important to detect icons such as Maya hieroglyphs.

HOOSS Descriptor

I tried a straightforward combination of the HOOSC descriptor with the Self Similarity algorithm, thus being a **Histogram Of Orientated Self Similarity** (HOOSS). The method starts by extracting HOG features for all pixels (not just the edge-pixels), and similar to the Self Similarity method it will create a correlation map. But instead of using the raw gray values, it will use HOG-features to compute the similarity score between 2 pixels. The resulting correlation map will be divided into a log-polar where each bin will be assigned with the average value of each pixel in the correlation map.

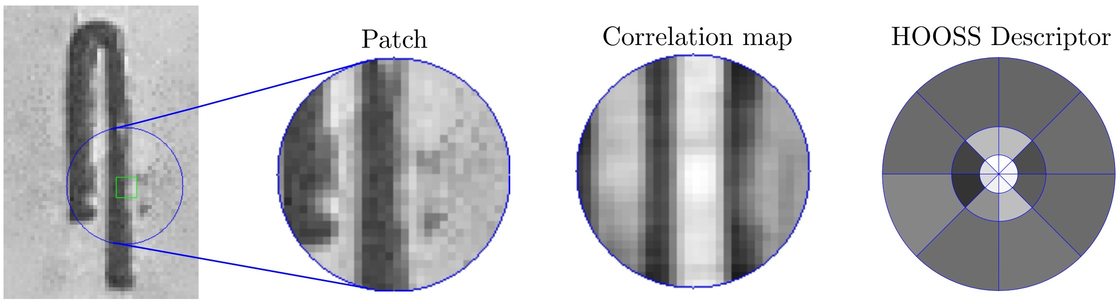


Figure 5.8: HOOSS process

5.2 Classification

Once the descriptors have been computed, all the information needed to classify an input image is present. The general approach for all methods is to compare an input image with every image in the train set (dataset which contains labelled images for different types of hieroglyphs). The comparison produces a similarity score according which the hieroglyphs are ranked.

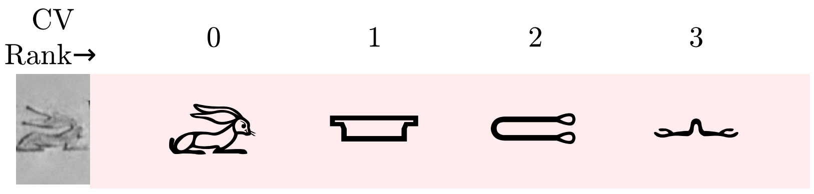


Figure 5.9: Example of ranked classification results

The similarity score is utilized as a confidence score for the Computer Vision part. The confidence score is important to the language model which will be explained later. I distinguish three different matching schemes for classifying an input image;

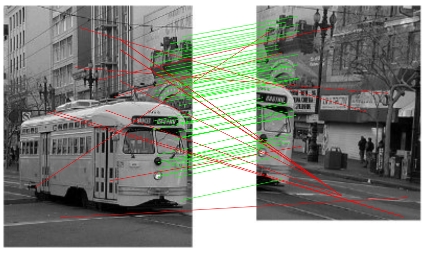
Image matching with RANSAC

The main classification scheme computes a similarity score between two images by comparing all the descriptors from both images with each other. It's a CPU-intensive process where each descriptor of one image is compared with every descriptor in the other image according the χ^2-distance (Chi-squared distance) (see eq. 5.4),

$$\chi^2_{ij} \equiv \chi^2(p_i, q_j) = \frac{1}{2}\sum_{k=1}^{K} \frac{(h_i(k) - h_j(k))^2}{h_i(k) + h_j(k)}. \tag{5.4}$$

Where $\chi^2(p_i, q_j)$ denotes the comparison between descriptor i from image p and the descriptor j from image q, $h_i(k)$ and $h_j(k)$ denotes the k-th bin of descriptor p_i and q_j respectively. The result is a cost matrix which contains the differences between all the descriptors in the first image and all the descriptors in the second image. The next step is to match each patch in the first image to exactly one patch in the second image such that the sum of all the costs is minimal. The general idea is to connect similar patches in both images, for example if I were to compare two images with the same bird, I want to connect the legs of the bird in the first image to the legs of the bird in the second image (as well as connecting the eyes/beak/wings/etc...). This task is known as a *Linear Assignment Problem* where each column can be matched with exactly one row (in a square matrix). Several algorithms can be used to solve such a problem such as the the Hungarian method [19] and LAP [15]. I chose to use the LAP algorithm as the authors guarantee that it is much faster than the Hungarian method. The result is a set of matches between the two images, however, a low difference score between two descriptor does not guarantee that the match is correct, to find out whether or not a match is correct can be computed by the RANSAC algorithm.

RANSAC
Ransac is a popular method that can determine how an image should be transformed to fit an other image and is usually applied when stitching multiple images on each other to create a panorama view. RANSAC is essentially a simple process, it takes a set of matching points, these are certain points in the image which correspond to the same point in another image. The Ransac algorithm selects a few (in this case 10) matches and computes an affine transformation matrix (with the least squares method) such that transforming the points in the first image according this transformation matrix will move each point as close to their matching point in the second image. The entire image is then transformed according this matrix, after the transformation is completed it will evaluate the transformation by computing the distance between each point and their matching point in the second image. If the distance exceeds a threshold value it is considered an outlier, otherwise it's an inlier. These in- and outliers are important as they indicate how successful the transformation is (the more inliers the better). This step is repeated a number of times and eventually the transformation matrix which yields the most amount of inliers is taken as the best transformation and is applied to the whole image which results in the stitched image as shown in the figure below.


Matches

Stitched

I have used the RANSAC method to find the affine transformation between the two images, it will transform one image to more resemble the other. The use of this transformation will make the algorithm more robust against against a difference in scale of the hieroglyph and a shift in the position. An overview of the stages within the RANSAC matching scheme is shown in figure 5.10. Once the transformation is completed the similarity score between the two images is computed according the following formula,

$$SimilarityScore = \frac{P^2}{m \cdot \sum_{(p_1,p_2)}^{P} \chi^2(p_1,p_2)}, \tag{5.5}$$

where m denotes the number of matches and (p_1, p_2) the matching pairs in P. The more inliers a transformation yields, the higher the *SimilarityScore*, it is divided by the total number of matches as well as the total χ^2 distance.

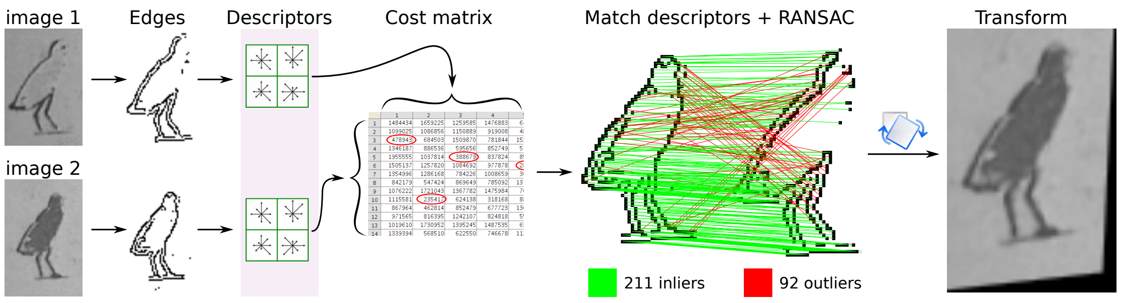


Figure 5.10: Pipeline for the matching scheme using RANSAC

Image matching with BOW

A popular image matching method is the *Bag Of Words* (BOW) approach where image features can be seen as words. The basic idea is to cluster similar features from all the images in the trainings set to obtain the so called Bag Of Words. A histogram is made from each image simply by counting how many of each word occur in the image.

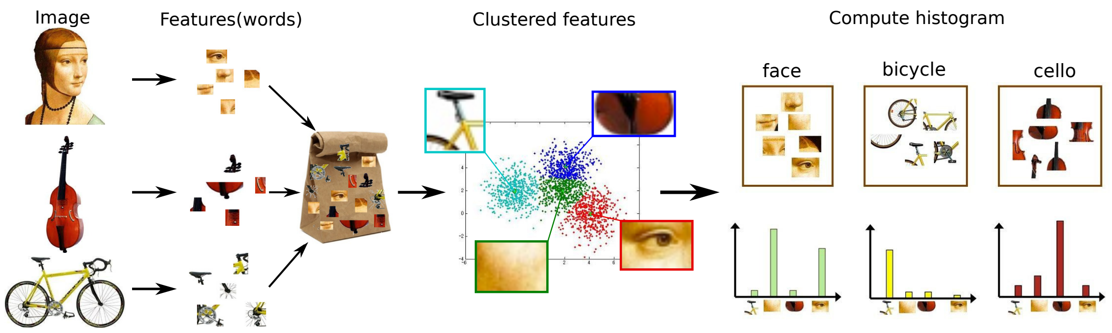


Figure 5.11: Bag Of Words pipeline

In figure 5.11 the BOW pipeline is depicted, but I have to point out that even though the features are depicted as a part of the image, they are actually the descriptors of these patches. Before the method can start to classify images, it first needs to do compute the most prominent features in the train set. First, all the features are extracted from the images in the train set, and then mixed together, a k-means [24] clustering algorithm will determine the k most prominent features (otherwise known as *visual words*). These visual words are then used to construct a histogram for all the images, this is done by finding the most similar visual word for each feature in an image and simply counting how many times each word occurs in that image.

The matching between two images comes down to comparing their histograms according the χ^2-distance (see equation 5.4).

Image matching with Single Descriptor

Although the RANSAC works great, it suffers from one major drawback, which is the large amount of time it takes to classify a single image because of the transformation it has to compute. This wouldn't normally be a big issue, but with the prospect of running it on a smart-phone it is vital to be able to classify a picture within a reasonable amount of time. Therefore the *Single*

Descriptor method was developed which simply computes a single descriptor of the entire image. Figure 5.12 shows how an image is divided in to an 8x8 grid.

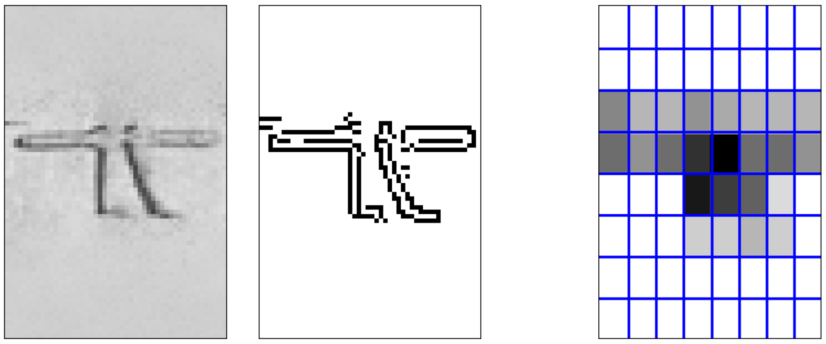


Figure 5.12: Single Descriptor bins

The matching between two images is once again computed by the χ^2-distance (see equation 5.4). The 8x8 bins means that computing the similarity between two images only requires the computation of the χ^2-distance for 64 elements (after the descriptor has been computed).

Using the language model 6

In the last phase the system has a chance to correct mistakes made in the previous step by using a language model to refine the classification results. Two different methods have been implemented where the first method uses a lexicon to find Egyptian words in the top-n proposed classification. The second method uses an n-gram approach which determines the likelihood of certain hieroglyphs occurring in a sequence.

6.1 Model by words

The Computer Vision part generates a ranking of possible classes for each input image, for the language model only the top-n is used. Each input image is evaluated by iterating over the top-n proposed classification, for each proposed classification the system searches for all words that can possibly be made with that hieroglyph and computes the score for that word based on the confidence of the Computer Vision part and the probability of that word occurring in an Egyptian text. the word confidence is the product of the Computer Vision scores of each hieroglyph in that word, normalised by the score of the highest ranked hieroglyph

$$\text{cvWordConfidence}_w = \prod_{j=0}^{length(w)} \left(\frac{\text{cvScore}_{(j+s),0}}{\text{cvScore}_{(j+s),\text{cvIndex}(j+s,w(j))}} \right)^3, \tag{6.1}$$

where j iterates over the every hieroglyph in word w. cvScore is a matrix containing all the ranked Computer Vision scores for each hieroglyph, as an example $\text{cvScore}_{a,0}$ is the best Computer Vision score for hieroglyph at position a and $\text{cvScore}_{a,1}$ the second best. s indicates the starting index of word w in a given text, cvIndex(i,g) indicates the rank of glyph g at position i, and $w(j)$ indicates the hieroglyph in word w at position j.

To improve the results even more I've incorporated the probability of each word occurring in an actual ancient Egyptian text. The addition of the word occurrence resolves issues with words that are rarely used, but do match the Computer Vision part slightly better than a common word. The word occurrence is computed by counting how many times a word appeared in the JSesh database divided by the total amount of words in that database. The addition of the word occurrence leads to the WordScore, and is computed by

$$\text{WordScore}_w = \frac{\lambda}{\text{cvWordConfidence}_w \cdot (P(w) + \lambda)} \cdot length(w), \tag{6.2}$$

where $P(w)$ is the probability of a word w occurring in an Egyptian text, λ indicates the smoothing that determines the impact of adding the probability scores to the equation. The λ variable should be > 0 and is necessary to avoid a *division by zero* in-case the word occurrence could not be determined because that word did not exist in the database used to mine the word-occurrence. Furthermore, when the system is unable to find any words that match the proposed classification, it will award the hieroglyph a default score based solely on the Computer Vision confidence. Once the WordScore is computed, the system will begin to re-rank the hieroglyphs based on the words that can be made with them.

A good example can be found in table 6.1, here the correct sequence of the hieroglyphs spells the word *Unas* (which is the name of the Pharaoh who was buried within this pyramid). However, the Computer Vision part makes a mistake and identifies the second hieroglyph as being a snake instead of a water hieroglyph. The language model will attempt to correct this by searching for words within the given top-n by the Computer Vision part.

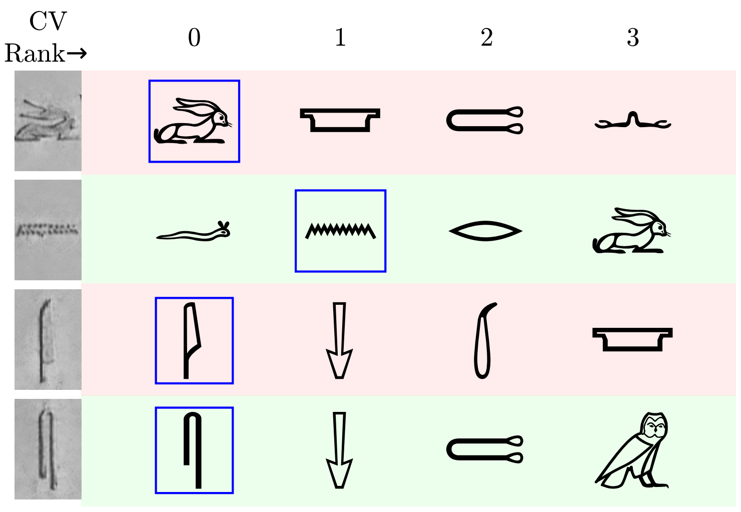

CV
Rank→
0
1
2
3

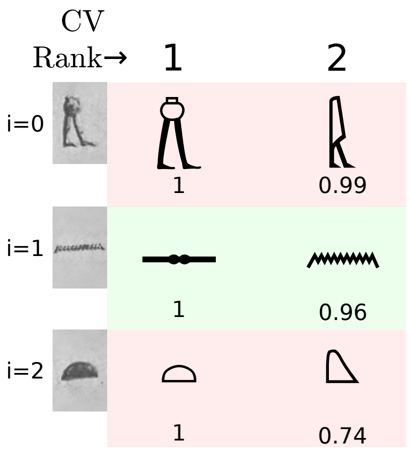
Table 6.1: Example of where the word-model will correct the CV results (These hieroglyphs spell the word *Unas*).

6.2 n-Grams

n-Grams are a number of subsequent characters which have a certain probability of occurring in any given text, where n is the number of subsequent characters. n-grams with $n = 1$ are referred to as uni-grams, while $n = 2$ are referred to as bi-grams and $n = 3$ are referred to as tri-grams. For example the English word 'beautiful' contains the following n-grams:

n-gram	beautiful								
uni-gram	b	e	a	u	t	i	f	u	l
bi-gram	be	ea	au	ut	ti	if	fu	ul	
tri-gram	bea	eau	aut	uti	tif	ifu	ful		

The most common tri-gram in English is "*the*", which makes up 3.4% of all tri-grams [33]. Tri-grams are currently widely used to determine the language of a given text simply by looking at which type of letter combinations are in present in the text. Dunning (1994) [6] demonstrated that is is surprisingly easy to determine the language of a given text using n-grams. The author tested with 9 different languages, he reported a 99.9% accuracy on large texts of 500 bytes and 92% accuracy on small texts of only 20 bytes.



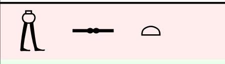
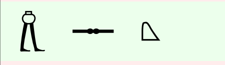
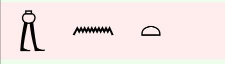
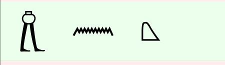
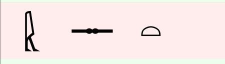
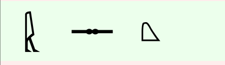
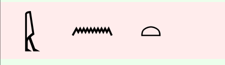
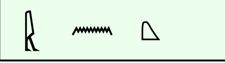
tri-Grams	Probability (x10^{-5})	Score (x10^{-5})
	0.5	50.5
	0	37
	73.5	118.6
	0	35.5
	0	49.5
	2.7	38.6
	2.2	49.6
	0	35.2

Figure 6.1: Example of n-gram process

To make use of the tri-grams for the hieroglyphs, I first train a model based on the Egyptian texts found in the JSesh database. Each tri-gram in the Egyptian texts is counted to obtain a probability score. The tri-grams are used to refine the classification results by mixing the tri-gram probability scores with the Computer Vision confidence such that common tri-grams will receive a boost in their score while rare tri-grams will receive a penalty. Although the effect of the tri-grams is low, it is enough to rearrange the classification score when the Computer Vision confidence is low.

In figure 6.1 an example is shown where the Computer Vision part misclassified the middle hieroglyph. The normalised Computer Vision confidence is shown below the proposed classification, this is the difference in score for the proposed classification at rank n and rank 1. The larger the difference in score between the first ranked hieroglyph and the second, the higher the confidence of the Computer Vision part. For the sake of this example, other glyphs surrounding these three have been removed and only the top-2 of the proposed classification results is taken as input. Eight different tri-grams can be made of the proposed classification and are shown in the table. The score for each tri-gram is computed according the following formula,

$$Score(\text{triGram}) = (P(\text{triGram}) + \lambda) \cdot \prod_{i=0}^{3} (\text{CvConfidence}(\text{triGram}_i))^3. \tag{6.3}$$

To give the Computer Vision confidence more influence it is taken to the power of 3. P(tri-gram) is the probability of that particular tri-gram occurring in an Egyptian text, this number is smoothed with a Laplace (λ) value to allow tri-grams that did not occur in the train set to still receive a proper score. The Laplace (λ) value is typically set to $50 \cdot 10^{-5}$. The n-gram which yields the highest score will be picked as final classification, in this case the mistake made by the Computer Vision part is corrected by the language model.

Results 7

THE dataset consist of 10 pictures of pyramid walls, totalling 3993 images of hieroglyphs and 161 different classes. For testing purposes, one pictures is taken aside and is used as the test set while the other nine pictures are used as train set. This test is conducted twice for two different pictures and the results are averaged. The two selected pictures for the test set contain a total of 793 hieroglyphs, both input images have been manually annotated as well as automatically annotated. The figure below illustrates the result for the automatically annotated hieroglyphs using the HOG descriptor with the Single Descriptor comparison matching scheme. The superimposed colours are identified below.

- 🟥 Failed to detect the hieroglyph.
- 🟨 Successful detection of the hieroglyph, but failed to classify it correctly.
- ▨ Successful detection but misclassified because the hieroglyph is not present in the trainings set, or because I was unable to determine the correct class.
- 🟦 Successful detection and classification.

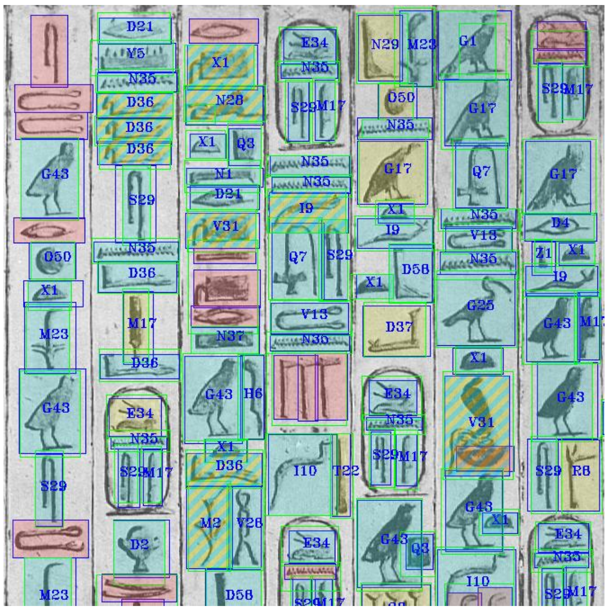


Figure 7.1: Visualization of the final results

The automatic detection method finds 83% of all manually annotated hieroglyphs and 85.5% of the detections are correct according the Pascal VOC overlap criteria (where a correct match has more than 50% overlap between the manually annotated hieroglyph and the automatically annotated).

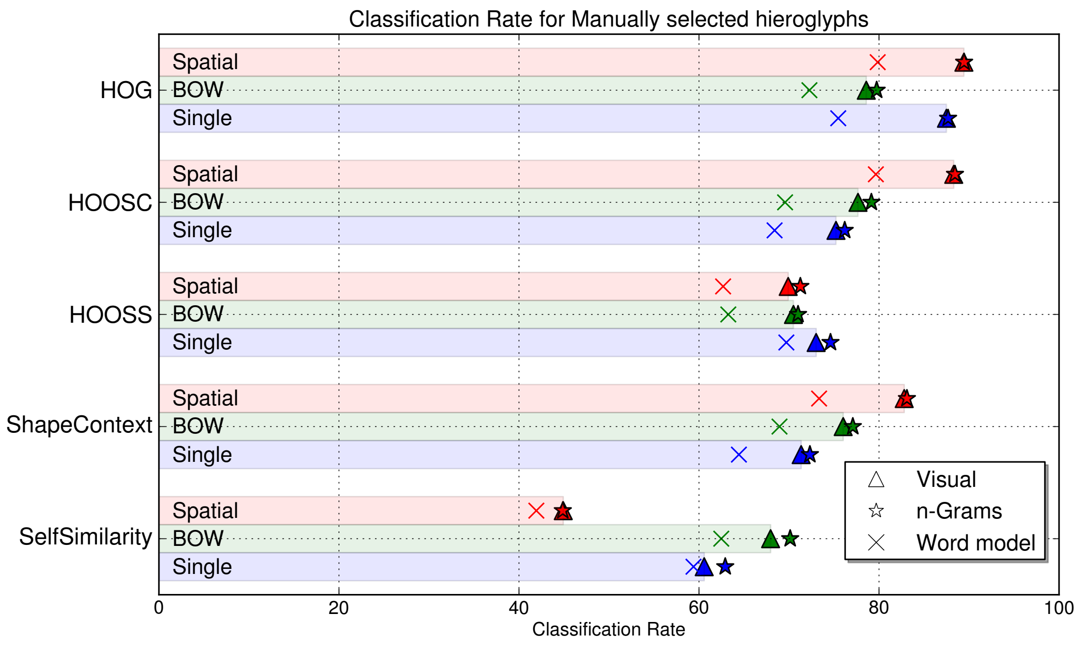


Figure 7.2: Results for the manually annotated dataset

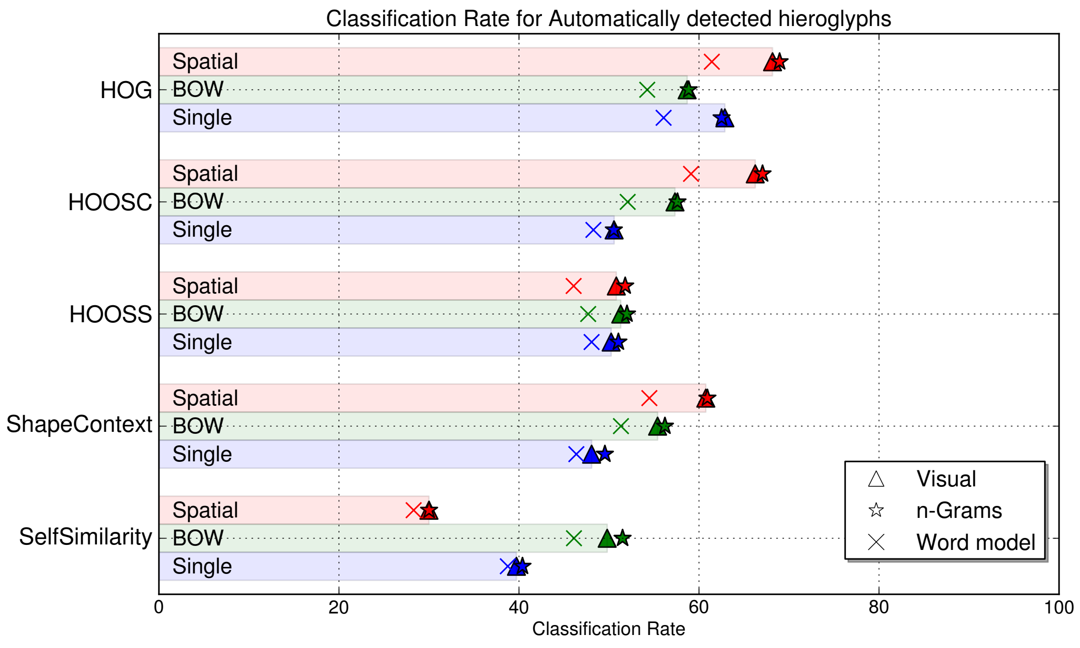


Figure 7.3: Results for the automatically annotated dataset

The results of the manually annotated hieroglyphs can be seen in figure 7.2, where the percentage of correct classifications is shown for all possible combinations of the following methods:

Descriptors:	HOG, HOOSC, HOOSS, ShapeContext and SelfSimilarity
Matching scheme:	Spatial (with RANSAC), BOW and Single descriptor
Language model:	Visual (no language model), *n*-grams and Lexicon

One of the first things to notice about the classification results is that using the lexicon language model does not improve the classification rate at all, in fact the classification rate becomes worse. This was unexpected and after some research it turned out that the Egyptians had many small words consisting out of only one or two glyphs. It was because of these small words that the language model was able to come up with a multiple of possible words for each hieroglyph and had to chose between them. The preference for small words is amplified by the word occurrence, since single glyph-words dominate on this domain. For example, the word "I" occurred over 3600 times in the dataset, while the average word occurrence is only 12. Although after extensive testing with different parameter settings on the test set, it was possible to improve the Computer Vision output with an average of 1.3%. But this resulted in a set of parameters specifically fine-tuned for the test set. The results show that the language model based on *n*-grams language model does have a positive effect on the performance, a small, but consistent 1% improvement in the classification rate.

Another important observation is that the Self-Similarity descriptor in combination with the RANSAC matching scheme performs much worse than the other methods. A possible explanation could be that the Self-similarity descriptor does not cope well with the spatial information, and applying the RANSAC algorithm would cause wrong patches to be matched to each other and confuse the comparison.

The difference between the automatically annotated hieroglyphs and the manually annotated is more than just the positioning, in some cases parts that do belong to the hieroglyph are blend in with the background because the system though it was noise. And as mentioned before, the amount of correctly annotated hieroglyphs is 85.5% with the automatic detection, meaning that 14.5% of the images cannot be correctly classified because these images do not contain any hieroglyph. The results for the automatic dataset are shown in figure 7.3. This is the most important test since it shows how many of the hieroglyphs are correctly classified given a raw picture containing hundreds of hieroglyphs.

With the prospect of running the hieroglyph recognition on smart phones it should be able to classify the images in a reasonable amount of time. In figure 7.4 the classification rate is shown against the average time it takes to classify one hieroglyph when using a 2.93GHz machine. The times are purely measured on the classification part and exclude the locating process as well as the addition of the language model. Although the best performance is achieved by combining a HOG descriptor with the Spatial comparison method (green triangle), it takes more than a minute to compute a single hieroglyph. Needless to say that this combination is not suitable to run on a mobile phone. The best choice is the HOG descriptor combined with the Single matching scheme (red triangle), which is roughly 8000 times faster at the cost of only 2% in classification rate.

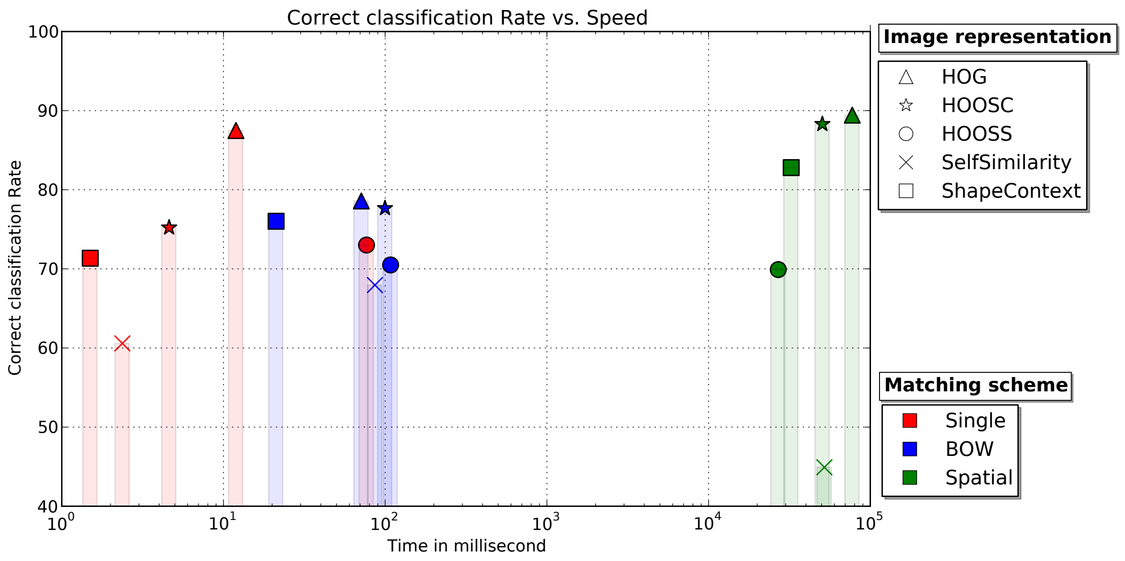


Figure 7.4: Correct classification rate (%) plotted against the average processing time in milliseconds.

Discussion 8

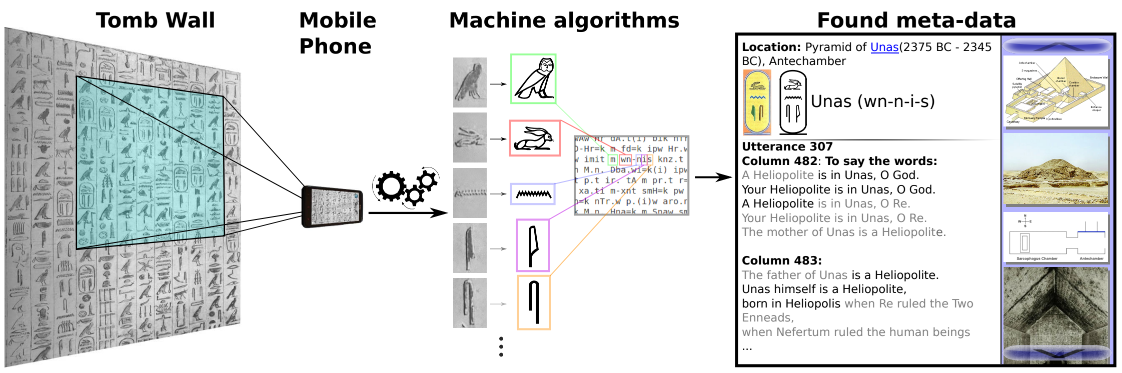
With this study I show the effect of different Computer Vision techniques on a interesting domain. I provide a unique dataset of ancient Egyptian hieroglyphs, containing 3993 annotated images of hieroglyphs. And with it the ability to easily annotate more hieroglyphs. This study shows the obstacles and dilemmas when facing the task of recognize ancient Egyptian hieroglyphs, and how to tackle them. I demonstrate that it is possible to detect and recognize the ancient Egyptian hieroglyphs with a reasonable classification rate by experimenting with different techniques, some of which can be applied on a smart-phone App.

Some of the methods used in this study have a common ground, for example the HOOSC descriptor makes use of the same capabilities than the ShapeContext algorithm and the HOG algorithm, which have also been tested individually.

Together with Jan van Gemert, we submitted a short paper titled "Automatic Egyptian Hieroglyph Recognition by Retrieving Images as Texts" to the ACM multimedia conference 2013 (see appendix A). By now I can happily announce that the paper has been accepted. However, some questions arose from the reviewers, such as the difference between Maya hieroglyphs and the Egyptian hieroglyphs. The questions is a reference to how the HOG descriptor can outperform the HOOSC descriptor for the Egyptian dataset (only slightly), but not on the Maya dataset. The reason for this might lie in the fact that the Maya hieroglyphs are much more detailed, while the Egyptian hieroglyphs are mostly made from a few strong lines. Therefore the 3x3 grid of the HOG descriptor is sufficient in capturing the essence of a patch. Another explanation might be that the authors of the HOOSC were using images of a much higher resolution than the 50x75 used for the Egyptian hieroglyphs. The reason for choosing the 50x75 resolution is because it was found sufficient for classifying the hieroglyphs.

At the moment, the classification system is unable to classify an input image as being "non-hieroglyphic" and can only classify it with a known hieroglyphic label. While this is of no concern when using the manually annotated dataset, it could become a problem when using the automatic detection system. Therefore, when focusing more on using the automatic detection system, it has to be able to differentiate between hieroglyphic images and background images.

A part of my thesis was to prove that it would be possible to create an application (App.) for smart phones which enables the users to gain background information and translation about ancient Egyptian texts by taking a picture of it. This would allow anyone with with a smart phone to unveil the mysteries of the ancient texts. I submitted the idea of creating such App (named Tomb Reader) to the AMSIA (Amsterdam Science and Innovation Award). As part of my proposal to the AMSIA I created a figure which illustrates the general idea of this App (see figure 8.1).



Location: Pyramid of Unas(2375 BC - 2345 BC), Antechamber

Utterance 307
Column 482: **To say the words:**
A Heliopolite is in Unas, O God.
Your Heliopolite is in Unas, O God.
A Heliopolite is in Unas, O Re.
Your Heliopolite is in Unas, O Re.
The mother of Unas is a Heliopolite.

Column 483:
The father of Unas is a Heliopolite.
Unas himself is a Heliopolite,
born in Heliopolis when Re ruled the Two Enneads,
when Nefertum ruled the human beings
...

Figure 8.1: From a picture of a hieroglyphic text, the App will locate and classify each individual glyph. The found glyphs will be matched to known and already translated texts. If it finds a match it will display various information about the text, including translation, time period and possibly related photographs.

I was among the 9 finalist from a total of 93 proposals. After a two and a half minute pitch in front of the jury I was awarded with a honourable mention. For the full proposal see appendix B.

In order to realise such App there is still some work to be done. The dataset must be expanded to include photographs other than those found in the pyramid of Unas to familiarise the classification algorithm with the different writing styles. Aside from expanding the visual dataset, a whole new dataset must be created which holds the Egyptian texts with translation and other background information. On top of that an algorithm has to be implemented that can retrieve the Egyptian text based on the found hieroglyphs, it has to be able to retrieve the correct text even though only a small section of the whole text is photographed.

After speaking with members of the Allard Pierson Museum, they were interested in such application and provided me with the ability to test it in the museum. An Approval letter from the Heritage lab of the Allard Pierson Museum is included in appendix C.

With this study I hope to give the readers some insights in the task of recognising ancient Egyptian hieroglyphs. The achieved classification rates should be sufficient to implement the second phase of the smart-phone App.

Acknowledgements

I would first like to thank my supervisor Jan van Gemert for guiding me through the difficult times of my thesis research. Our mutual interest in recognizing the ancient Egyptian hieroglyphs resulted in an accepted paper to the ACM multimedia 2013. So once again, an extreme thank you Jan for being the best supervisor I could ever hope for. In addition I would like to thank Sezer Karaoglu for providing me with the code for text recognition algorithm that ultimately led to the hieroglyph detection system in this thesis. I am particularly grateful for the assistance given by Prof. dr. Olaf Kaper (Leiden University) for introducing me to the field of Egyptology and recommending me the book "The pyramid of Unas" by A. Piankoff. His suggestions laid the foundation of this thesis. I wish to acknowledge the help provided by Jolanda van Iperen who helped writing my thesis and supported me along the way. Thanks also goes out to Robrecht Jurriaans for helping me to prepare my pitch about Tomb Reader for the Amsterdam Science and Innovation Award, his suggestions made me revise the entire pitch for the better.

Bibliography

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24 (4), 2002.

[2] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[3] D.C. Cireşan, U. Meier, L.M Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010.

[4] D.C Cireşan, U. Meier, J. Masci, L.M Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two*, pages 1237–1242. AAAI Press, 2011.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[6] T. Dunning. *Statistical identification of language.* Citeseer, 1994.

[7] A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[8] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR*, pages 2963–2970, 2010.

[9] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[10] Y. Frauela, O. Quesadab, and E. Bribiescaa. Detection of a polymorphic mesoamerican symbol using a rule-based approach. *Pattern Recognition*, 39, 2006.

[11] A. Gardiner. *Egyptian Grammar.* Griffith Institute, 1957. ISBN 0900416351.

[12] J. van Gemert. Exploiting photographic style for category-level image classification by generalizing the spatial pyramid. In *ICMR*, 2011.

[13] P. Isola, J. Xiao, A. Torralba, and A. Oliva. What makes an image memorable? In *CVPR*, 2011.

[14] R. Johnson, E. Hendriks, J. Berezhnoy, E. Brevdo, S. Hughes, I. Daubechies, J. Li, E. Postma, and J. Wang. Image processing for artist identification. *Signal Processing Magazine, IEEE*, 25(4):37–48, 2008.

[15] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4), 1987.

[16] S. Karaoglu, J. van Gemert, and T. Gevers. Object reading: Text recognition for object recognition. In *ECCV-IFCVCR Workshop*, 2012.

[17] H.Y. Kim and J.H. Kim. Hierarchical random graph representation of handwritten characters and its application to hangul recognition. *Pattern Recognition*, 34(2):187–201, 2001.

[18] F. Kimura, K. Takashina, S. Tsuruoka, and Y. Miyake. Modified quadratic discriminant functions and the application to chinese character recognition. *Pattern Analysis and Machine Intelligence*, (1):149–153, 1987.

[19] H.W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2), 2006.

[20] L. Ladicky, C. Russell, P. Kohli, and P. Torr. Graph cut based inference with co-occurrence statistics. In *Computer Vision–ECCV 2010*, pages 239–253. Springer, 2010.

[21] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[22] M. Levoy et al. The digital Michelangelo project: 3D scanning of large statues. In *Computer graphics and interactive techniques*, 2000.

[23] C.L. Liu, F. Yin, Q.F. Wang, and D.H. Wang. Icdar 2011 chinese handwriting recognition competition. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1464–1469. IEEE, 2011.

[24] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14, 1967.

[25] V. Magdin and H. Jiang. Discriminative training of n-gram language models for speech recognition via linear programming. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 305–310. IEEE, 2009.

[26] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Computer VisionECCV 2002*, pages 128–142. Springer, 2002.

[27] A. Piankoff. *The Pyramid of Unas.* Princeton University Press, 1968.

[28] C.A. Rahman, W. Badawy, and A. Radmanesh. A real time vehicle's license plate recognition system. In *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.*, pages 163–166. IEEE, 2003.

[29] E. Roman-Rangel, C. Pallan, J.M. Odobez, and D. Gatica-Perez. Retrieving ancient maya glyphs with shape context. In *ICCV Workshop*, 2009.

[30] E. Roman-Rangel, C. Pallan Gayol, J.M. Odobez, and D. Gatica-Perez. Searching the past: an improved shape descriptor to retrieve maya hieroglyphs. In *ACM MM*, 2011.

[31] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *CVPR*, 2007.

[32] R. Smith. An overview of the tesseract ocr engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. IEEE, 2007.

[33] R.L. Solso, P.F. Barbuto, and C.L. Juel. Bigram and trigram frequencies and versatilities in the english language. *Behavior Research Methods & Instrumentation*, 11(5):475–484, 1979.

[34] X. Tong and D.A Evans. A statistical approach to automatic ocr error correction in context. In *Proceedings of the fourth workshop on very large corpora*, pages 88–100, 1996.

[35] P. Viola and M.J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

Paper accepted in the ACM Multimedia conference 2013 A

Automatic Egyptian Hieroglyph Recognition by Retrieving Images as Texts

Morris Franken and Jan C. van Gemert
Intelligent Systems Lab Amsterdam (ISLA), University of Amsterdam
Science Park 904 Amsterdam, The Netherlands

ABSTRACT

In this paper we propose an approach for automatically recognizing ancient Egyptian hieroglyph from photographs. To this end we first manually annotated and segmented a large collection of nearly 4,000 hieroglyphs. In our automatic approach we localize and segment each individual hieroglyph, determine the reading order and subsequently evaluate 5 visual descriptors in 3 different matching schemes to evaluate visual hieroglyph recognition. In addition to visual-only cues, we use a corpus of Egyptian texts to learn language models that help re-rank the visual output.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.4 [**Image Processing and Computer Vision**]: Features Measurement—*Feature representation, size and shape*

Keywords

Egyptian, Hieroglyphs, Automatic Recognition

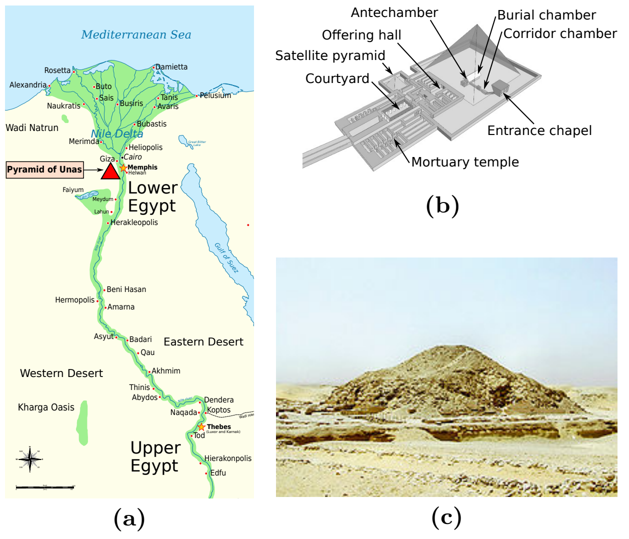


Figure 1: The pyramid of Unas. (a) Location at red triangle. (b) Schematic reconstruction. (c) Current state. *Images courtesy of Wikipedia, creative commons license.*

1. INTRODUCTION

The ancient Egyptian hieroglyphs have always been a mysterious writing system as their meaning was completely lost in the 4th century AD. The discovery of the Rosetta stone in 1799 allowed researchers to investigate the hieroglyphs, but it wasn't until 1822 when Jean-François Champollion discovered that these hieroglyphs don't resemble a word for each symbol, but each hieroglyph resembles a sound and multiple hieroglyphs form a word. The ability to understand hieroglyphs has uncovered much of the history, customs and culture of Egypt's ancient past.

In this paper we present a system that is able to automatically recognize ancient Egyptian hieroglyphs from photographs. As illustrated in fig 4, a single photograph contains several, often overlapping, hieroglyphs without proper segmentation or a-priori reading order. Moreover, the passing of 4 millennia has introduced noise, and broken or destroyed the original symbols. These conditions present severe challenges to automatic hieroglyph recognition. Automatic hieroglyph recognition is useful for archaeology scholars, the interested amateur, cultural heritage preservation or as a smart-phone App for a tourist or museum visitor.

The paper has 3 contributions. First, we introduce a new hieroglyph dataset where we manually segmented and labeled 3993 hieroglyphs of 10 photographs from the pyramid of Unas. This pyramid is built in the fifth dynasty as a burial place for the Pharaoh *Unas* and is located just south of the city of Giza, see fig 1. We chose a single pyramid to avoid issues with different dialectic writing styles. Second, we show how to automatically locate, segment and recognize hieroglyphs based on visual information. The third contribution is a multimodal extension to the visual analysis with statistical language models from hieroglyph texts. In fig 2 we show a schematic of our approach.

2. RELATED WORK

Multimedia tools have aided preservation, analysis and study of cultural, historical and artistic content. For ex-

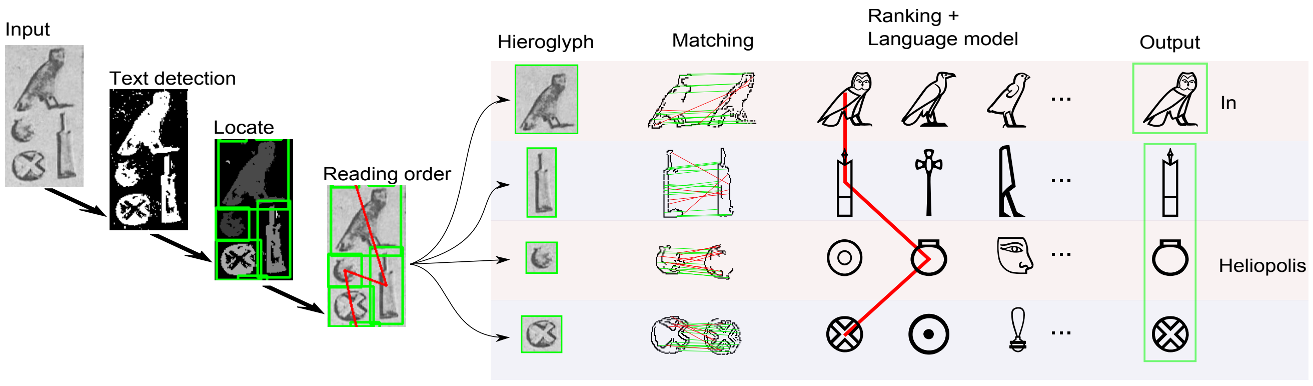


Figure 2: Pipeline for hieroglyph recognition. The 3rd output hieroglyph from the top is corrected by the language model in order to find the word 'Heliopolis' (birth-city of Unas).

ample, the digital Michelangelo project [12] created high quality 3D models of Michelangelo's sculptures and architecture. Furthermore, wavelet analysis of brush strokes in paintings can reveal artist identity [7], image composition has shown to aid category labeling [17] and photographs can be classified as memorable or not [6]. In this paper we follow in these footsteps, and propose a new dataset, and a multimodal (visual and textual) approach for automatic Egyptian hieroglyphs recognition.

Related work on automatic hieroglyph recognition focuses on Mesoamerican culture, and in particular on the ancient Maya hieroglyphs [5, 14, 15]. To this end, the HOOSC descriptor was developed [15], which is a combination of HOG [2] and the Shape-Context [1]. Such descriptors can be used for direct matching [14] or with a bag-of-words (BOW) approach [15]. Other work extracts detailed line segments for Maya hieroglyph matching [5]. In all these works the hieroglyphs are typically manually extracted and individually digitized. In contrast, our photographs consists of noisy plates, which each typically contain around 400 hieroglyphs (see fig 4). Moreover, the Maya culture used a considerable different type of hieroglyphs and we therefore evaluate the HOOSC and other descriptors on Egyptian hieroglyphs.

Current work on automatic scene text detection and recognition [4, 9, 10, 11] are typically hand-tuned to specific western or asian (e.g. Chinese or Hangul) characters which are quite different from Egyptian hieroglyphs. In our work, we will draw inspiration from text detection to localize the hieroglyphs and use generic image descriptors for the recognition.

3. HIEROGLYPH RECOGNITION

Our approach has a visual component where the reading order is determined, hieroglyphs are localized, pre-processed and visually matched. The top-N visual matches are subsequently input to a textual component that re-ranks the hieroglyphs according to a statistical language model trained on other texts. In fig 2 we show a schematic of our approach.

3.1 Localization and Pre-Processing

We localize hieroglyphs with a saliency-based text-detection algorithm [9] (software kindly provided by the authors). This algorithm does not make assumptions on the shape of the text, as done e.g. in [4]. The used algorithm creates a saliency map based on second order curvature statistics and subsequently uses conditional dilation from border-seeds to create a binary text/non-text mask. We group the noisy masks by connected components after a dilation. The output of the localization is an unordered list of bounding boxes (BBs) fitted around each glyph. We experimentally evaluate the localization performance in section 4.

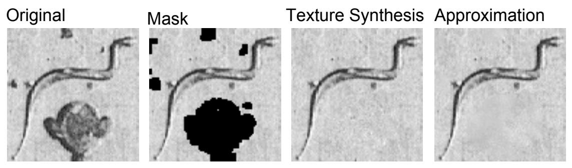


Figure 3: Removing unconnected hieroglyphs.

From the unordered list the reading order is determined to facilitate textual language modeling. The reading order in Egyptian hieroglyphs is either from left to right, right to left or from top to bottom. The only indication of the correct reading order is that glyphs will face the beginning of a line, and top to bottom is indicated by columns separators. Multiple horizontal hieroglyphs in a column should be read as a single line. For the pyramid of Unas the reading order is top-down, from right to left. We sort the unorderd list accordingly to determine the sequence in which the hieroglyphs should be read.

The hieroglyphs BBs are often overlapping or they are in contact with a 'cartouche' (a frame around the name of a royalty). To generate individual hieroglyph images suitable for matching, we filled the overlapping parts with background texture by a non-parametric texture synthesis method [3]. This approach works well, however it is rather slow. We therefore implemented a faster approximation. For each pixel to generate we randomly sample from a search window around the closest filled-in pixel. If the sampled pixel is not part of the foreground mask it is kept, otherwise the process is repeated with a larger window size. After texture synthesis is complete, the final background is smoothed to reduce noise. Our approximation is in the order of 300 times faster, the results of both methods are shown in fig 3. As a final step the patches are extended to 50x75 while retaining their discriminative height/width ratio where the background is again texture synthesized if necessary.

3.2 Image Descriptors

We evaluate five image descriptors, two based on shape, one based on appearance, and the other two on a shape-appearance combination. The Shape-Context (SC) [1] is a shape descriptor originally designed for recognizing handwritten symbols and therefore interesting for hieroglyphs. The SC constructs a spatial log-polar histogram that counts the frequency of edge pixels. Similar to the SC, the Self-Similarity [16] (SS) descriptor was designed for shape matching. Where the SC counts edge pixels, the SS computes the pixel correlation between the central cell to the other cells in a log-polar shaped region. The SC and the SS are shape descriptors, but the hieroglyphs may also benefit from appearance matching. The Histogram of Oriented Gradients (HOG) [2] is a popular appearance descriptor that computes histograms of edge gradient orientations. A combination of shape and appearance can be achieved by merging the HOG and the SC. This combination is called the HOOSC [15] and was developed for Maya hieroglyph matching. The HOOSC replaces the edge pixels in the SC with a HOG. As the fifth descriptor we add a straightforward combination of the SS with HOG which we dub HOOSS. For this combination, the correlations in SS between pixels are replaced with similarities of HOGs. All similarities between K-dimensional descriptors i and j are computed with the χ^2-distance, $\chi^2(i,j) = \sum_1^K \frac{(i_k - j_k)^2}{i_k + j_k}$.

3.3 Visual Matching

To recognize a hieroglyph we compare it to labeled patches in the training set. To this end, we evaluate three common image patch matching schemes. The first method is simply using a single centered descriptor for a full patch. The second approach uses interest points with the popular bag-of-words framework as also used for Maya hieroglyphs [15]. The third approach has also been used for Maya glyphs [14] and uses pair-wise patch matching using interest points with spatial verification. This method starts with a variant of the Hungarian method [8] to assign each descriptor in one patch to a descriptor in the other patch. After obtaining matches, the final matching score s is obtained by fitting an affine transformation between the patches by using Ransac and is computed as $s = m * \sum_{(p_1,p_2)}^{P} \chi^2(p_1,p_2)/|P|^2$, where m is the number of matches, and (p_1,p_2) are matched pairs in the set of Ransac inliers P.

3.4 Language Modeling

The visual ranking is based on a single hieroglyph and does not take the neighboring hieroglyphs into account. The neighborhood can give valuable information since hieroglyphs often co-occur to form words and sentences. To take the context into account we employ language models of hieroglyph occurrences to improve the visual ranking. We compare two strategies: (1) a lexicon lookup and (2) N-grams which represent statistics of N-neighboring hieroglyphs.

The lexicon-based approach tries to match N consequative hieroglyphs to existing words in a dictionary. For each hieroglyph i, we look at the top K=10 results $(v_{i1}, v_{i2}, \ldots, v_{iK})$ of the visual ranking. We re-rank each hieroglyph i by word length $|w|$ and occurrence probability $P(w)$ as $(p(w)+\lambda_w) \cdot |w| \cdot \prod_{j=1}^{K}(v_{i1}/v_{ij})^2$, where w is any exact word match that is present in the Carthesian $N \times K$ space of possible words where N is equal to the largest word in the corpus. To reduce the influence of non-existing words we use a standard Laplace smoothing term λ_w, which we set to 1/20. In section 4 we give the details of the used corpus. We found the best non-linear weighting of visual scores (in this case v^2) on a small hold-out set.

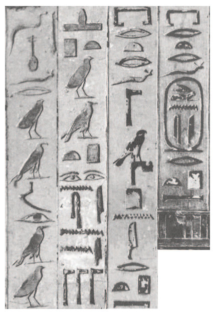

...
he shall not write with his little finger. How beautiful is indeed the sight, how good indeed to see, so say they, so say the gods, (when) this god ascends to heaven, (when) Unas ascends to heaven while his power is over him
...

Figure 4: Part of a plate taken from the north wall of the antechamber (column 476) and its translation.

The N-gram approach uses the probability of hieroglyph sub-sequences of length N occurring in a row. We re-rank each hieroglyph i with $\prod_{i=1}^{N}\prod_{j=1}^{K}(v_{i1}/v_{ij})^3 \cdot (p(w) + \lambda_n)$, where w is a hieroglyph sequence of length $N = 3$. To reduce the influence of non-existing N-grams we use a Laplace smoothing term λ_n of 1/2000. Again, we found the best non-linear weighting of visual scores (v^3) on a hold-out set.

4. EXPERIMENTS

We evaluate all descriptors, matching techniques and language models on our new hieroglyph dataset.

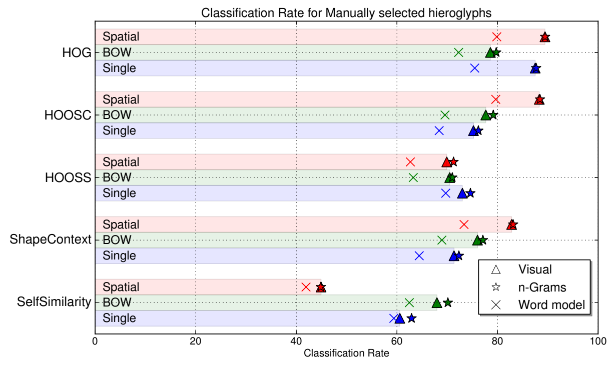
4.1 Dataset

The dataset consist of two sets: one being photographs of hieroglyphic texts, the other being the textual corpus that is used for the language model. The visual set consists of 10 plate photographs with hieroglyphs [13], as illustrated in fig 4. These photographs contain 161 unique hieroglyphs with a total of 3993. We manually segmented all individual hieroglyphs with a bounding box and annotated them with their label. To record the label we use a form of transliteration which transforms the hieroglyphs into a character. Many transliteration variations exist among Egyptologists, which are rarely compatible with each other. In this research we chose to use the transliteration used by the open-source JSesh project[1] which also gave rise to the textual set of the database, containing 158 pyramid texts (with a total size of 640KB of text). This textual set is used to train the language models and does not contain any texts from the pyramid of Unas.

4.2 Implementation Details

To reduce differences between descriptors due to parameter setting we keep parameters as equal as possible over the five variants. For HOG we use 8 angle bins, and 4x4 spatial binning. The HOG inside HOOSC and HOOSS also use 8 angle bins. All log-polar histograms have 3 rings, and 8 segments. For the bag-of-words matching we found that a vocabulary size of 200 visual words works well. In the spatial matching we use 500 Ransac iterations. The interest

[1]`http://jsesh.qenherkhopeshef.org/`



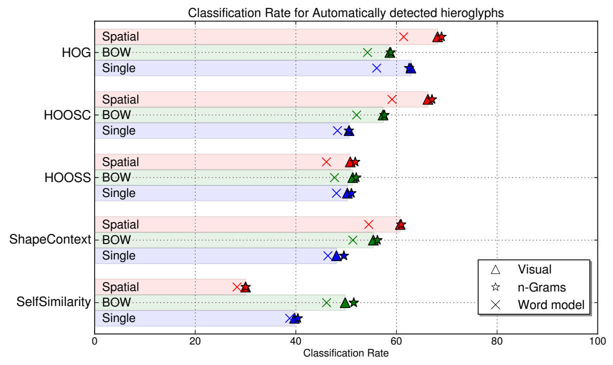


Figure 5: Results for manually cut-out hieroglyphs. The average score is $74 \pm 1\%$.

Figure 6: Results for automatically detected hieroglyphs. The average score is 53 ± 5.

points in the BOW and in the spatial matching are based on the contour [14, 15] of a Canny edge detector.

To simulate taking a single photograph, we use one plate for testing and the other plates for training. We repeat the experiment to obtain standard deviations.

4.3 Results

We give results for manually cut-out hieroglyphs recognition in fig 5 and for our automatic detection approach in fig 6. The automatic detection method finds 83% of all manually annotated hieroglyphs, and 85.5% of the detections are correct according to the Pascal VOC overlap criteria. The matching performance trend between the automatic and the manual annotated hieroglyphs is similar, although the single-descriptor HOG seems slightly more sensitive to localization errors.

From the 5 descriptors, the HOOSC and the HOG are the best performers. HOOSC is best on manually annotated hieroglyphs whereas HOG is more robust for automatically detected regions. This seems to indicate that flexibility in spatial structure is important, given the reduced performance of single descriptor HOG on the automatically detected glyphs.

Between the three matching schemes, the spatial-matching performs best. Only for the Self-Similarity the BOW is better. Generally the Self-Similarity does not perform as well on this dataset, which could be attributed to the lack of discriminative features such as color. The slightly better performance of the spatial matching scheme, however, is a factor of 8,000 times slower compared to a single descriptor and 1,000 times slower than the BOW.

Language modeling with a lexicon decreases results on average with 5%. This is due to a bias for smaller words and the lack of word-separators. The N-grams always improves results, albeit slightly, with on average 1%.

5. CONCLUSION

We presented a new Egyptian hieroglyph set with a thorough evaluation of five popular image descriptors, three common matching schemes and two types of language modeling.

6. REFERENCES

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *TPAMI*, 24(4), 2002.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[3] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[4] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR*, pages 2963–2970, 2010.

[5] Y. Frauela, O. Quesadab, and E. Bribiescaa. Detection of a polymorphic mesoamerican symbol using a rule-based approach. *Pattern Recognition*, 39, 2006.

[6] P. Isola, J. Xiao, A. Torralba, and A. Oliva. What makes an image memorable? In *CVPR*, 2011.

[7] R. Johnson, E. Hendriks, J. Berezhnoy, E. Brevdo, S. Hughes, I. Daubechies, J. Li, E. Postma, and J. Wang. Image processing for artist identification. *Signal Processing Magazine, IEEE*, 25(4):37–48, 2008.

[8] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4), 1987.

[9] S. Karaoglu, J. van Gemert, and T. Gevers. Object reading: Text recognition for object recognition. In *ECCV-IFCVCR Workshop*, 2012.

[10] S. Karaoglu, J. van Gemert, and T. Gevers. Con-text: Text detection using background connectivity for fine-grained object classification. In *ACM-MM*, 2013.

[11] F. Kimura, K. Takashina, S. Tsuruoka, and Y. Miyake. Modified quadratic discriminant functions and the application to chinese character recognition. *PAMI*, (1):149–153, 1987.

[12] M. Levoy et al. The digital Michelangelo project: 3D scanning of large statues. In *Computer graphics and interactive techniques*, 2000.

[13] A. Piankoff. *The Pyramid of Unas*. Princeton University Press, 1968.

[14] E. Roman-Rangel, C. Pallan, J.-M. Odobez, and D. Gatica-Perez. Retrieving ancient maya glyphs with shape context. In *ICCV Workshop*, 2009.

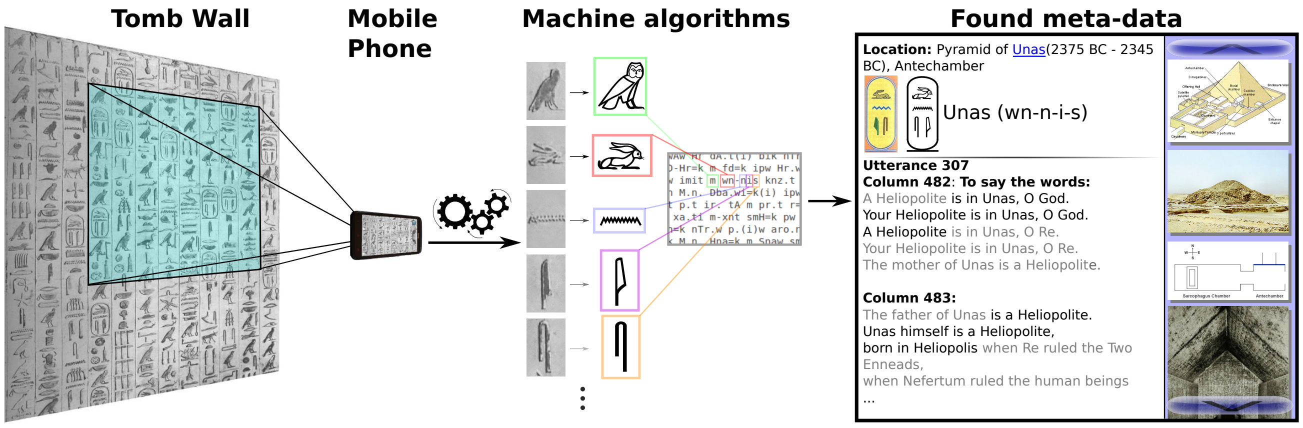
[15] E. Roman-Rangel, C. Pallan Gayol, J.-M. Odobez, and D. Gatica-Perez. Searching the past: an improved shape descriptor to retrieve maya hieroglyphs. In *ACM MM*, 2011.

[16] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *CVPR*, 2007.

[17] J. van Gemert. Exploiting photographic style for category-level image classification by generalizing the spatial pyramid. In *ICMR*, 2011.

App Proposal to the Amsterdam Science and Innovation Award B

Hieroglyph Recognition on a Smart Phone



Location: Pyramid of Unas(2375 BC - 2345 BC), Antechamber

Utterance 307
Column 482: **To say the words:**
A Heliopolite is in Unas, O God.
Your Heliopolite is in Unas, O God.
A Heliopolite is in Unas, O Re.
Your Heliopolite is in Unas, O Re.
The mother of Unas is a Heliopolite.

Column 483:
The father of Unas is a Heliopolite.
Unas himself is a Heliopolite,
born in Heliopolis when Re ruled the Two Enneads,
when Nefertum ruled the human beings
...

Figure 1: From a picture of a hieroglyphic text, the App will locate and classify each individual glyph. Database matching with known and already translated texts yield the translation and any desired amount of meta-data.

Main idea

The ancient Egyptian hieroglyphs have always been a mysterious writing system as their meaning was completely lost in the 4th century. Despite the discovery of the Rosetta stone in 1799, it wasn't until 1822 when Jean-François Champollion discovered that these hieroglyphs don't resemble a word for each symbol, but each hieroglyph resembles a sound and multiple hieroglyphs form a word. To this day there are only a few people who are capable of reading the ancient texts. With current technology, however, it is possible recognize these symbols such that anyone can unveil the mysteries of the ancient writings. The main idea is illustrated in figure 1.

Current state

For my master thesis I developed a system that can automatically locate and identify Egyptian hieroglyphs. The system is trained on more than 3000 images of hieroglyphs and achieved a classification rate of 89% on manually annotated hieroglyphs. The algorithm starts by locating the hieroglyphs on an input image using an image saliency-based algorithm, the hieroglyphs are cut out and compared with the images in the train set. Each comparison between two images yields a similarity-score on which the hieroglyphs are ranked. Furthermore, the probability of certain hieroglyphs occurring together is weighted and is used to refine the classification results. The entire process is illustrated in figure 2.

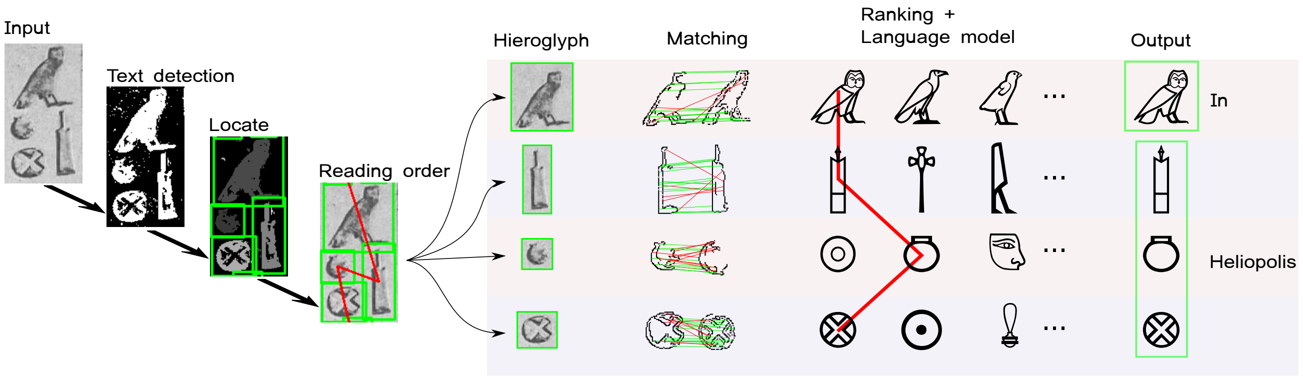


Figure 2: Using detection, recognition and occurrence statistics to find the word 'Heliopolis' (birth-place of Unas).

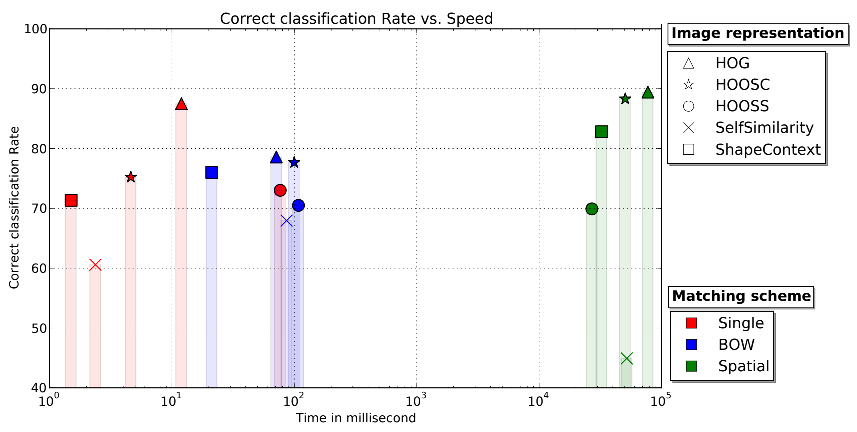
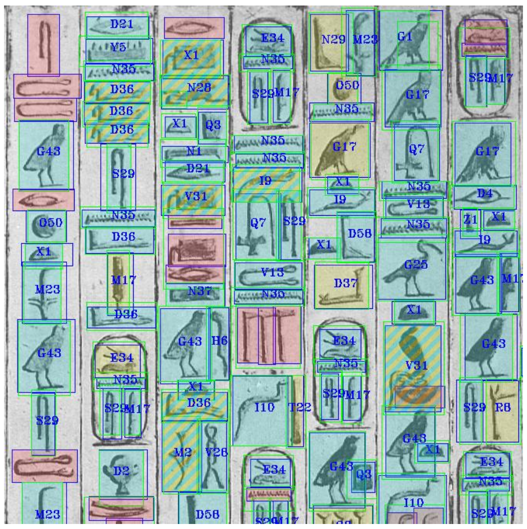


Figure 3: Correct classification rate (%) plotted against the average processing time in milliseconds.

Five different image representations were used in combination with three different matching schemes in order to evaluate the scalability for a mobile App. In figure 3 the average time it takes for a desktop computer to classify one hieroglyph is plotted against the classification rate . Although the best performance is achieved by combining a HOG descriptor with the Spatial comparison method (green triangle), it takes more than a minute to compute just one hieroglyph. Needless to say that this combination is not suitable to run on a mobile phone. The best choice is the HOG descriptor combined with the Single comparison method (red triangle), which is roughly 8000 times faster at the cost of only 2% in classification rate.

The figure on the right illustrates the final classification result for the HOG descriptor with the Single comparison method denoted by the red triangle in figure 3.
The superimposed colors are identified below.

🟥	Failed to detect the hieroglyph.
🟨	Successful detection of the hieroglyph, but failed to classify it correctly.
▨	Successful detection but misclassified because the hieroglyph is not present in the trainings set.
🟦	Successful detection and classification.



Towards the final App

The App is dependent on the training images and the available meta-data. The meta-data and the translations have to be put into a database, which should be portable, and accessible on a mobile phone. The training images should ideally be obtained from the locations where the App will be used. In practice though, I expect that images obtained through a (digital) library are of sufficient quality, as long as copyright law is respected. All training hieroglyphs need to be individually annotated with an identifier. For this, I have software readily available.

The remaining practical steps towards a fully-functional App are the linking of the meta-data and the consolidation of the research in a professional and high-quality smart phone App. This involves creating an intuitive user interface and compiling versions for iPhone, Android and possibly Windows phone. Moreover, the App can be primed with a GPS location of a pyramid or on a specific time period of the Egyptian dynasties.

In conclusion, my research has shown that a computer can recognize hieroglyphs, which makes it possible for tourists or museum visitors to obtain a personalized and interactive tool to explore the mysteries of the ancient Egyptian texts just by pointing a smart phone.

C Approval letter from Heritage lab of the Allard Pierson Museum

UNIVERSITY OF AMSTERDAM
ERFGOEDLAB

18 April 2013

Als coördinator van het UvA ErfgoedLab kan ik stellen dat er steeds meer technologische ontwikkelingen worden gebruikt om de ervaring van de museum bezoekers te verbeteren. Denk hierbij aan de recent geinstalleerde Etruscanning in het Allard Pierson Museum waarbij bezoekers zich wanen door een virtuele tombe en met behulp van handgebaren kunnen bewegen en informatie opvragen. Met behulp van digitale toepassingen is het goed mogelijk de interactie tussen bezoeker en museumobject nieuwe lagen van interactie te laten ontsluiten. Het UvA ErfgoedLab is hier als experimentele ruimte voor de presentatie van erfgoed zeer in geïnteresseerd. We verkennen meerdere toepassingen om dit nieuwe gereedschap in te zetten.

Daarom spreek ik namens het UvA ErfgoedLab mijn interesse in het "*Hieroglyph recognition application for smart phones*" project uit. Deze toepassing kan ons helpen om bezoekers meer interactief te betrekken tijdens een museum bezoek door hen de mogelijkheid te geven stukken oud-Egyptische teksten te vertalen. Ik zie zeker mogelijkheden voor een pilotproject in het UvA ErfgoedLab.

Met vriendelijke groet,

[illegible]

J.L. Bolten,

Coördinator van het UvA ErfgoedLab