
Software Requirements Specification

for

Dua-Khety: Hieroglyphics Detection and Classification

**Prepared by: Malak Sadek, Mohamed Badr El Din,
Ahmed El-Agha, Mohamed Ghoneim**

<The American University in Cairo>

<12th of December, 2017>

Table of Contents

Table of Contents	ii
Revision History	ii
1. Introduction.....	1
1.1 Purpose.....	1
1.2 Document Conventions.....	1
1.3 Intended Audience and Reading Suggestions.....	1
1.4 Product Scope	1
2. Overall Description	2
2.1 Product Perspective.....	2
2.2 Main Product Components	2
2.3 User Classes and Characteristics	3
2.4 Operating Environment.....	3
2.5 Design and Implementation Constraints	3
2.6 User Documentation	4
2.7 Assumptions and Dependencies	4
3. External Interface Requirements	4
3.1 User Interfaces	4
3.2 Hardware Interfaces	7
3.3 Software Interfaces	7
3.4 Communications Interfaces	8
4. System Features	8
4.1 Preprocessing Functions	8
4.2 Segmentation Functions.....	12
4.3 Feature Extraction Functions	14
4.4 Classification Functions.....	16
4.5 Optimization Functions.....	21
4.6 Social System Functions	23
4.7 Miscellaneous Functions.....	24
4.8 Use Case Tables for Functions	26
5. Other Nonfunctional Requirements	35
5.1 Performance Requirements.....	35
5.2 Safety Requirements	35
5.3 Security Requirements	35
5.4 Software Quality Attributes	35
5.5 Business Rules	35
6. Other Requirements	36
Appendix A: Glossary	36
Appendix B: Analysis Models	40
Appendix C: Sources & Useful Material	41

Revision History

Name	Date	Reason for Changes	Version

1. Introduction

1.1 Purpose

The purpose is to create a mobile application that is able to detect a subset of hieroglyphics in an image taken from the mobile's camera, improve the image's quality through various preprocessing operations, isolate the hieroglyphics using segmentation, identify and classify these hieroglyphics using machine learning techniques and classifiers, and finally present the user with the Gardiner number of the hieroglyph.

1.2 Document Conventions

All definitions are stated in the glossary at the end of the document. Firstly, all interfaces are stated. Then, a use case diagram outlines the entire system with its subsystems. Main functional requirements are stated one by one and including a short description, their sequence of events, and their requirements. All non-functional requirements are stated after that.

1.3 Intended Audience and Reading Suggestions

The application is intended for archeologists and scientists with an interest in deciphering hieroglyphics accurately and quickly, as well as tourists who want to find out more about the Egyptian culture on the go. This document is mainly intended for the Computer Science/Engineering professors at the American University in Cairo as the Software Requirements Specification document for the authors' thesis. However, it is also intended any person with interest in the product, concepts will be simplified as much as possible so that little technical background is needed to understand.

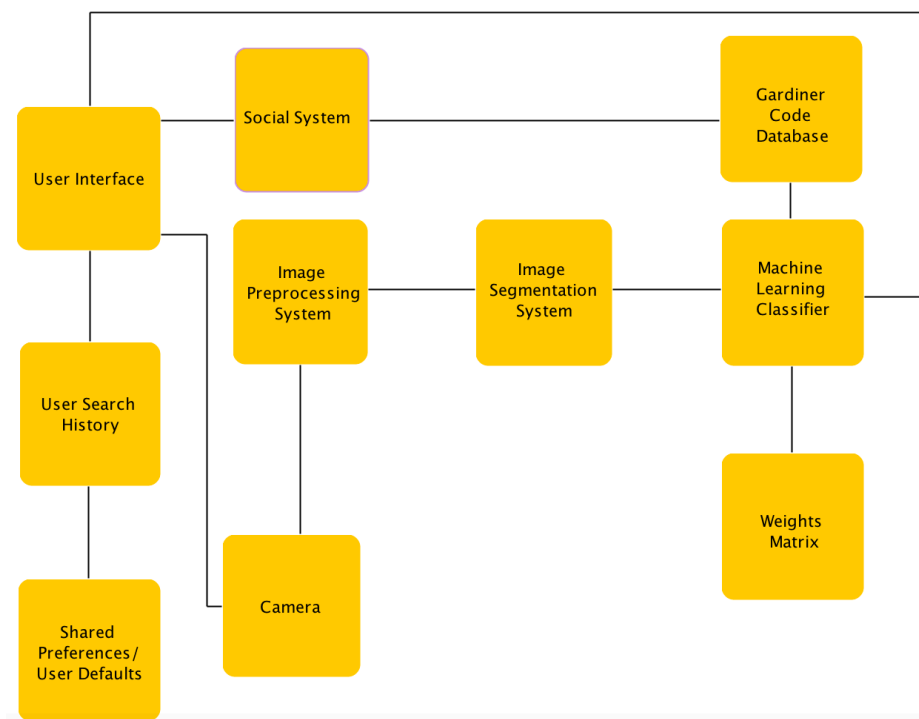
1.4 Product Scope

The software is a machine-learning based mobile application, designed as efficiently as possible with the aim of providing its users with the Gardiner number of a certain subset of hieroglyphics that they provide pictures of in the most practical and user-friendly way possible.

2. Overall Description

2.1 Product Perspective

The mobile application will be a self-contained product. A high-level overview of which systems will need to interact with each other and with databases is shown below.



2.2 Main Product Components

(General top level systems, the options and functions considered for each system are stated in section 4, see appendix for use case diagrams)

Image Preprocessing System

- Resize Image
- Binarize Image
- Filter Noise
- Use Wiener Filter
- Use Median Filter

Image Segmentation System

- Perform Edge Detection
- Use Canny Operator
- Perform Region-Based Segmentation
- Use Histogram
- Use Hough Transform

Machine Learning Classifier

- Classify Image

Social System

- Upload photo

- Comment on Photo
- Search for Information
- Add Information
- Check In
- Filter Content
- History:**
- Update User History
- Clear History
- Display History
- Miscellaneous:**
- Open Camera
- Submit Image
- Review Image
- Add to Training Data
- Add to Database
- Update Local Database
- Open Gallery
- Display Results
- Display Confidence
- Provide Transliteration

2.2 User Classes and Characteristics

There will be two user classes. There will be an opportunity to sign up and create an account if you are a professional, either a scientist or archeologist. This will unlock the ability to take pictures of individual hieroglyphics and submit them with a unique code (that can be used to identify hieroglyphics between users) to be added to the training data to further improve the system's accuracy, and once enough pictures are submitted for a certain class that the system does not currently classify, they can be added to the database to increase the number of hieroglyphs the system is able to classify. It will also unlock usage of the social system where users can post a picture along with its information and location and provide information about it for other people to search for or ask questions about it. The other user class is composed of users who use the application without signing up for single use. They will not be able to submit photos or access social features.

2.3 Operating Environment

The system must be able to operate on Android, and iOS to be fully accessible to users. If internet is unavailable, the user can still classify offline, however any social operations, providing transliterations from third parties, as well as more accurate classification are optionally available online. The Gardiner code database and users' search history will be saved locally on users' devices. Any databases needed for social system content will be stored online.

2.4 Design and Implementation Constraints

Currently, the system will only be implemented in English. There will be real-time constraints on the system, which will mainly be the maximum tolerated response time. This will be measured as the time between the point when the user takes a picture using the application, to the time when the user is

presented with the Gardiner number. This will be largely limited by the capabilities of the mobile device used. This will also be affected by decisions concerning segmentation and feature extraction and the classifier used (discussed in more details later on). There will be an interface to a 3rd party transliterator, possibly hieroglyphs.net developed by Paul Sciortino. A database will be utilized for storing corresponding Gardiner numbers for the subset of Hieroglyphics chosen, and one to store the content posted within the social system. Matrices of the weights for classification will also be locally kept on the device and a balance must be made between the space they take and their number for classification accuracy.

2.5 User Documentation

The users will be provided with an introductory tutorial the first time they open the application presented as a page with instructions on how to use all the features of the application, the subset of Hieroglyphics the application can detect, and contact information for the developers.

2.6 Assumptions and Dependencies

The application will only work on a certain subset of Hieroglyphics.

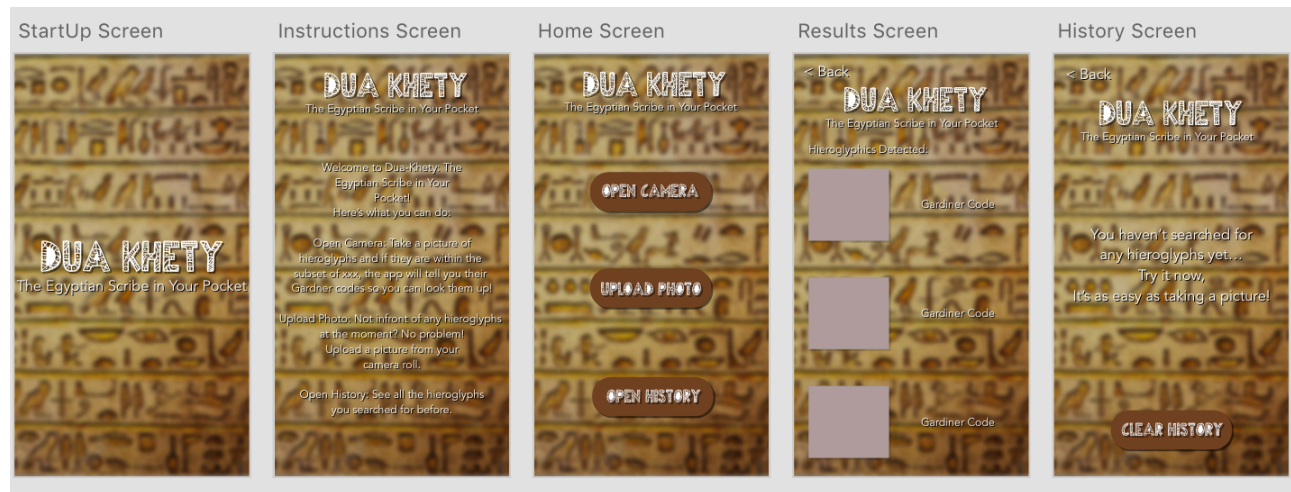
3. External Interface Requirements

3.1 User Interfaces

The application will have few user interfaces as most of the functionalities will be shielded from the user. They will be presented with a home screen from which they can open their history or the camera or upload a photo, after taking the picture they will be presented with a screen showing the Gardiner number of each Hieroglyphic and a button to return them to the home screen. Professional users will also have an interface to submit images for review and adding to the database or training set as well as an interface to access social features of the application.

The user interfaces will be:

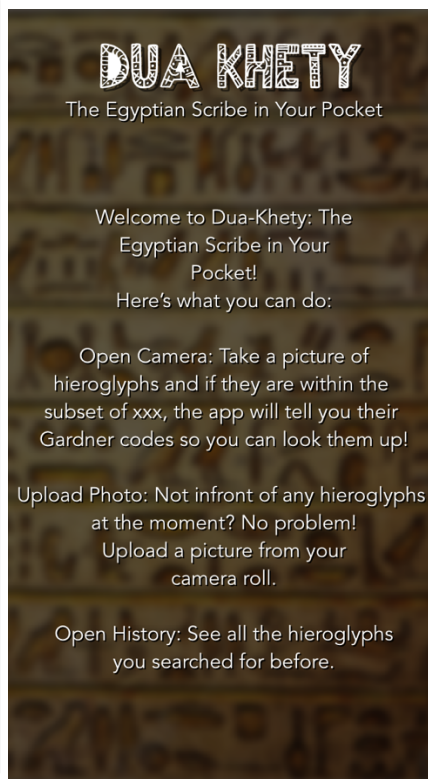
- Start Up Screen*
- Instructions Screen*
- Home Screen*
- History Screen*
- Interface to Camera*
- Results Screen*
- Submit Image Screen*
- Upload Photo Screen (Social System)*
- Search for Photo Screen*

-View/Comment on Photo Screen

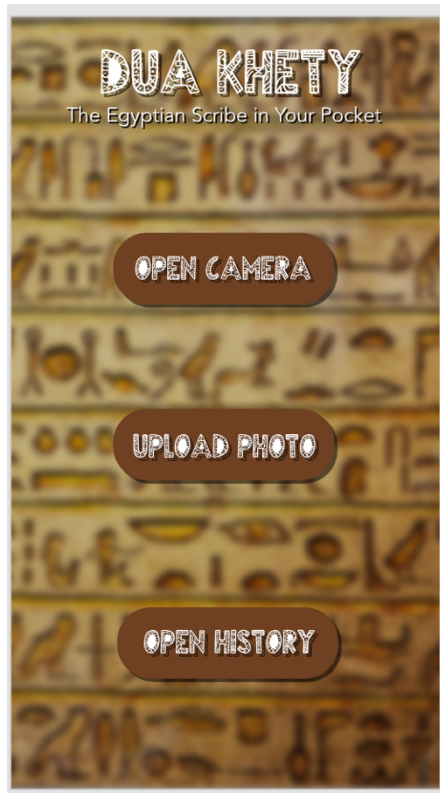
1. All screens



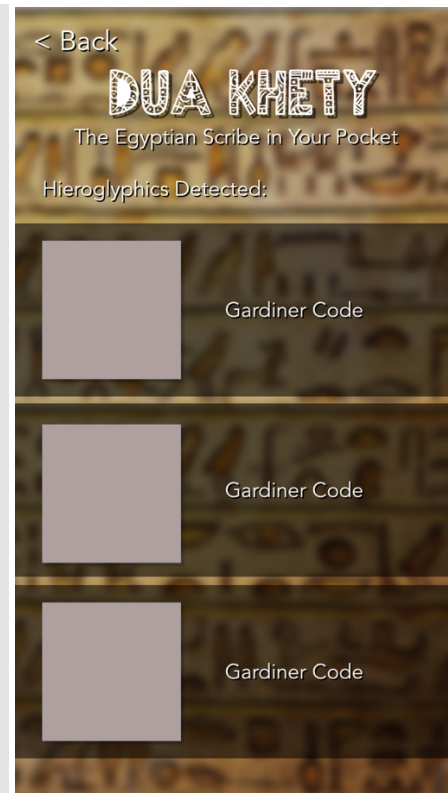
2. Start up Screen



3. Instructions Screen



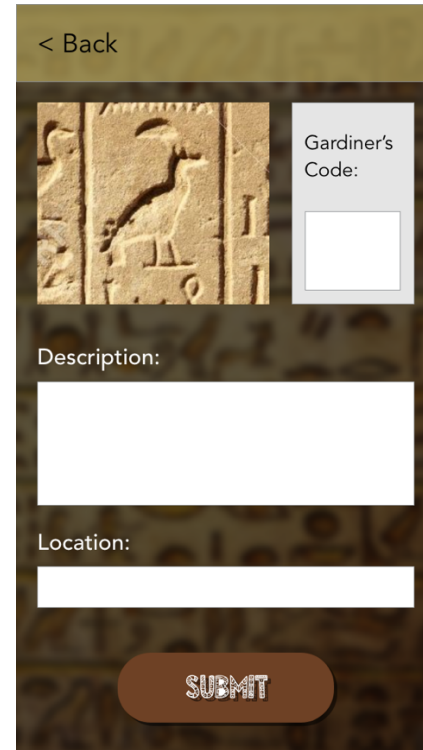
4. Home Screen



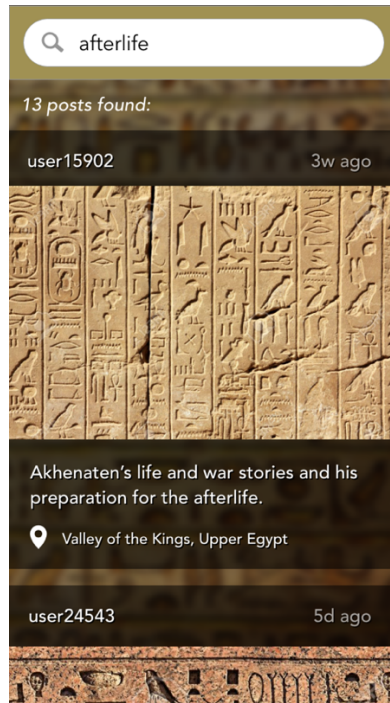
5. Results Screen



6. History Screen



7. Image Submission/Posting Screen



8. Searching for Information Screen

3.2 Hardware Interfaces

The system will only run on mobile devices. It will provide a means to open the camera from within the application, but will not need to communicate with the camera itself. The databases needed will also not require any hardware interfaces.

3.3 Software Interfaces

The local database will be created using SQLite and will seldom need to be updated, however the application will need to be updated whenever the database needs to be updated since it will be stored locally. An online database will be created with MySQL. Search history will be saved on the devices' User Defaults (iOS)/Shared Preferences (Android). An interface may also be needed to hieroglyph.net to access and extract information from it for transliteration. We will also interface to Google Maps to provide geographical locations for posts within the social system. And Firebase will be needed to safely manage user accounts.

3.4 Communications Interfaces

The system will only need to send emails to professional users when they create their accounts for verification, as well as upon changing their passwords or deleting their accounts. We might also add notifications for social features such as if someone commented on a photo they uploaded.

4. System Features

Below is a comprehensive list of all options that are being considered for use within the system, divided based upon the sub-systems. The final part of this section discusses the techniques the system is actually likely to use with justification.

4.1 Preprocessing Functions

All functions needed to perform the operations discussed below are available on either OpenCV, Scipy, or Matlab (image processing toolkit).

4.1.1 Resizing

It is important to make sure that all images are the of similar size, resolution, and distance (concerning both the training set and future images taken by the application). This can be done by rescaling the largest side of each image to a fixed length.

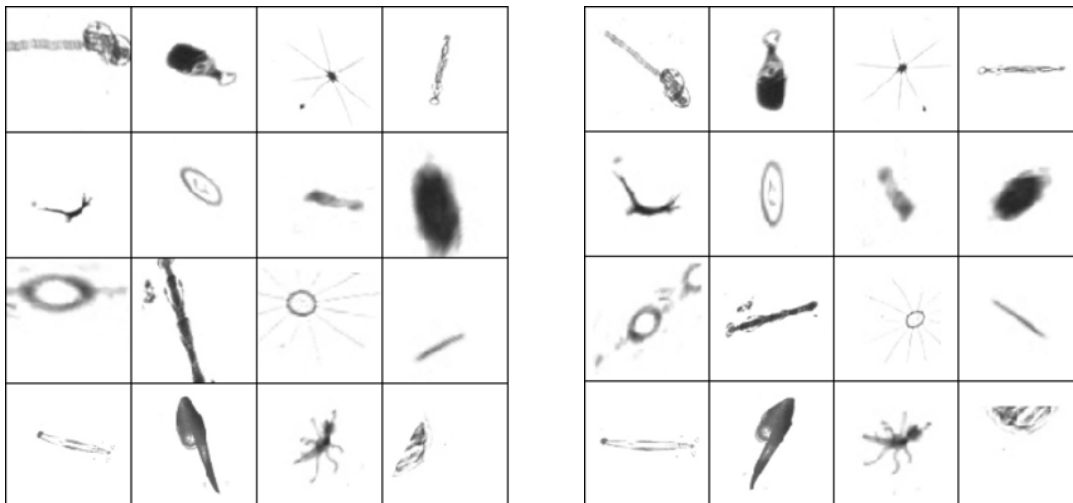
OpenCV and Scipy can both be used for this objective:

```
Resized_image = cv2.resize(image, (x,y))      //OpenCV
```

```
Resized image = scipy.misc.imresize(image, factor)      //Scipy
```

4.1.2 Augmentation

Data augmentation has been proven to increase performance in almost all cases. It refers to purposely making changes in the training dataset to increase its size and thus increase classifier accuracy. This is heavily dependent on the dataset being used. For example, an image of a chair should not be turned upside down, but can be flipped horizontally. Augmentation techniques also include shearing, stretching, translation, and rotation.



An example of data augmentation, source:

<https://datascience.stackexchange.com/questions/5224/how-to-prepare-augment-images-for-neural-network>

4.1.3 Noise Filtering

This refers to filtering out unnecessary data from images using different kinds of filters (low pass, high pass, etc.) to remove different types of noise from the images depending on what is needed. This can also be done using level set approach which makes use of curves cancelling each other out. OpenCV and Matlab also have functions that allow de-noising:

`Cv2.fastNIMeansDenoising()`

`Cv2.fastNIMeansDenoisingColored()`

`Cv2.fastNIMeansDenoisingMulti()`

`Cv2.fastNIMeansDenoisingColoredMulti()`

4.1.3.1 Weiner Filter

This kind of filter aims at removing noise caused by data signal corruptions. It uses a certain statistical equation to realize this.

$$G(u, v) = \frac{H^*(u, v) P_s(u, v)}{|H(u, v)|^2 P_s(u, v) + P_n(u, v)}$$

where

$H(u, v)$ = Degradation function

$H^*(u, v)$ = Complex conjugate of degradation function

$P_n(u, v)$ = Power Spectral Density of Noise

$P_s(u, v)$ = Power Spectral Density of un-degraded image

4.1.3.2 Mean Filter

The mean filter is a form of average filtering which sets pixel values to the average of the pixel values around them, eliminating all forms of grain noise. Each component of a pixel will fall under a certain mask, which is then averaged to form the value of a single pixel.

4.1.3.3 Blurring

Blurring can also be used as a rudimentary noise filtering technique. It is not favorable due to the loss of edge strength, leading to less accurate edge detection later on.

4.1.4 Thresholding

While noise filtering removes unnecessary data, thresholding removes data that is not noise, it contains actual information about the image, however it is not needed for the purposes of classification – a different kind of noise. This could be achieved by removing pixels who have attributes falling outside defined ranges (color values would be an example).



An example of an image before and after Thresholding. Source: [https://en.wikipedia.org/wiki/Thresholding_\(image_processing\)](https://en.wikipedia.org/wiki/Thresholding_(image_processing))

4.1.5 Binarization

This is the process of converting a color or grayscale image into a binary image of pure black and white and is achieved by firstly converting the image to grayscale (if it is not already) and then applying an appropriate level of thresholding to it.

4.1.6 Mean Image Subtraction

This works well if there are certain attributes that are not consistent throughout the image. For example, if a color is centered around a certain object or the light intensity in one part of the image is stronger than the rest. This will only be needed in cases where sunlight interfered with a part of the image, however most hieroglyphics and the mediums they are on have similar color schemes.

4.1.9 Whitening

This is sometimes referred to as normalization and it means turning the distribution of an image into a normal distribution. The pixels of an image can be normalized to fall on the same gradient with a define range centered around 0. Matlab has functions for performing this.

```
mat2gray()
```

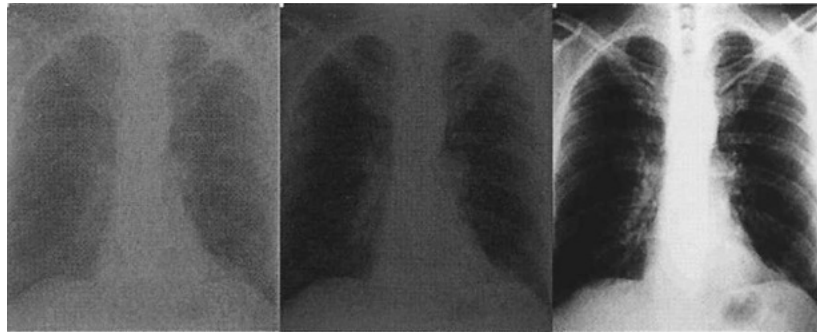
```
im2double()
```


4.1.10 Dimensionality Reduction

The data is transformed into a compressed version with less dimensions. The amount of loss of data from the image to achieve this is controllable. This would only be used as an optimization technique if needed.

4.1.11 Contrast Stretching

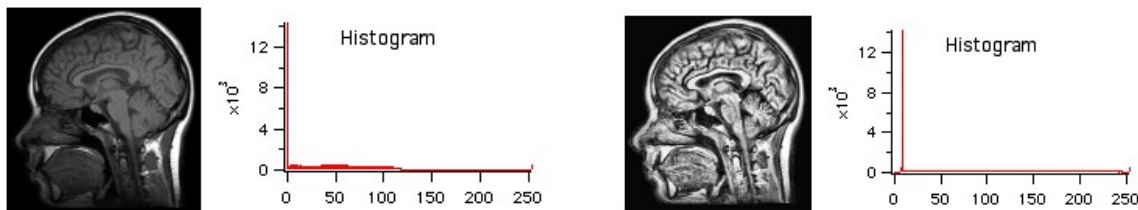
This is especially useful in our case as it targets images that are mostly homogenous. Different stretching techniques can be used to increase the narrow range of gray-levels in an image and increase the differences between them to be more noticeable.



Example of the effect of contrast stretching. Source: <http://what-when-how.com/embedded-image-processing-on-the-tms320c6000-dsp/contrast-stretching-image-processing/>

4.1.12 Histogram Modification

This can be used to supplement previously discussed methods, instead of using individual processes, histograms of the image can be altered directly and will reflect the characteristics of the image. For example, equalizing the image can achieve the same result as a combination of contrast stretching, normalization, and binarization in a single step as demonstrated below.



Source: <https://www.wavemetrics.com/products/igorpro/imageprocessing/imagettransforms/histmodification.htm>

The system will mainly use resizing to make sure images taken by users are the same dimensions as the training set images, noise filtering using Weiner filters and Mean filters as they are the most versatile filters available, and binarization and thresholding along with contrast stretching/histogram equalization, to optimize and homogenize images for optimal segmentation.

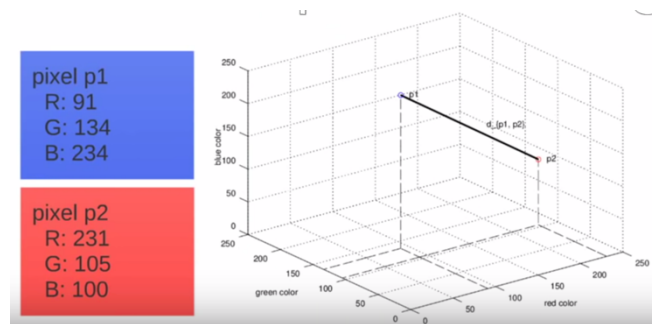
4.2 Segmentation Functions

4.2.1 Region-Based Segmentation

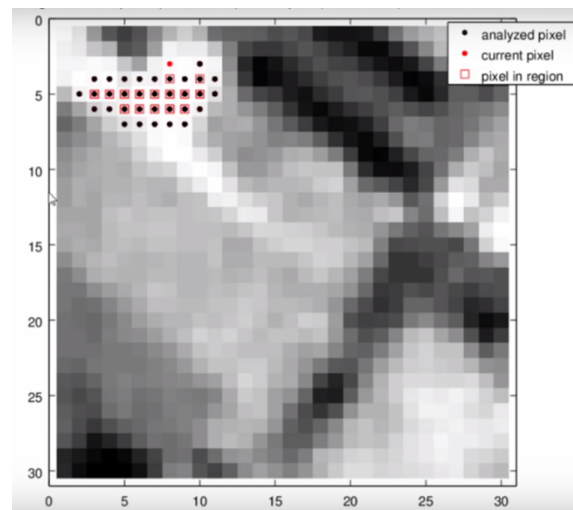
In this method, pixels are grouped based on the objects they are a part of. The bounds of a region can be determined using color and texture and then the edge is converted into a vector for further segmentation.

The simplest method to perform region-based segmentation is through gradient thresholding. Pixels are categorized depending on their intensity value. Where the assumption is made that pixels with similar values must belong to the same object/region.

Another method would be regional growth segmentation. The Euclidian distance is used to measure homogeneity between pixels. A homogeneity threshold is defined to determine whether two pixels are homogenous or not.



If a pixel is found to be homogenous to its neighbor, they are then merged together, hence the name regional growth.

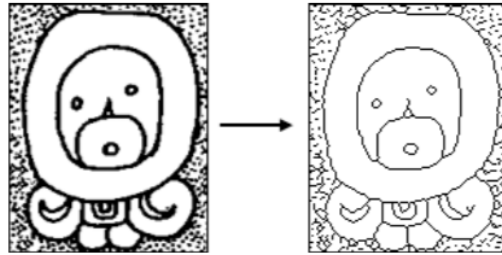


An example of finding the Euclidian distance and of binding homogenous pixels. Source:
<https://www.youtube.com/watch?v=VaR21S8ewCQ>

4.2.2 Edge-Based Segmentation

Here, the boundaries between regions alone are used to segment images. The Support Vector Machine (SVM) is commonly used for this and both its fixed and adaptive features can be utilized. Again, edges are detected by comparing neighboring pixel values. Thinning can also be used in Matlab to convert edges into one pixel thick lines.

```
bwmorph(image, 'thin')
```



An example of thinning. Source: Irving-Pease, Reading Maya Hieroglyphs: A Machine Vision Approach (2008)

The Sobel operator uses a kernel filter along with the maximum of the first derivative to detect edges as intensity will sharply change at those areas

The Canny operator extracts thinner and clearer edges. It begins with noise reduction using the Gaussian Convolution, it then uses the Sobel operator to initially detect edges. However, it adds a non-maxima suppression to get thin lines, where pixels that are not related edges are minimized. Broken and dashed edges are also removed. Scaling is also variable, while the Sobel operator sticks to 3X3 kernels.



Example of the Canny operator. Source:

<https://pdfs.semanticscholar.org/ee0a/414f7b4a7a16c8f8ef8c9437f59aeced4cef.pdf>

4.2.3 Segmentation Based on Clustering

In this technique, the image is converted to a histogram and then the pixels of the image are clustered. Python supports many cluster algorithms, most famously the K-Means algorithm.

```
Sklearn.cluster.KMeans()
```

Other examples of algorithms are the C-Means algorithm and the Seeded Growing Region (SRG) algorithm. This method can be useful if combined with histogram methods for image preprocessing to overall streamline the two systems together.

4.2.4 Model Based Segmentation

There are many pre-defined models for segmentation that can be directly used. The most prominent example is the Markov Random Field (MRF). It automatically detects regions based on inbuilt constraints and color pixels are then used as extra information leading to very accurate results.

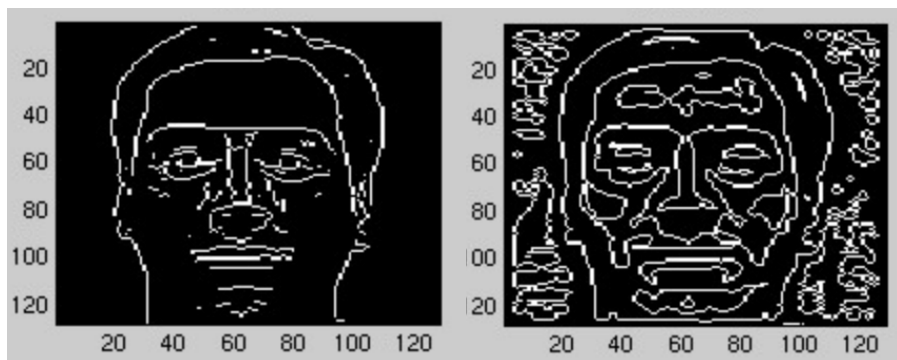
4.3 Feature Extraction Functions

4.3.1 Low-Level vs. High-Level Feature Extraction

Low-level extraction involves extracting more primitive features of an image, such as edges/lines, colors, etc. High-level extraction extracts objects based on processes more similar to how humans would detect objects, and therefore utilizes machine learning (a training system is needed). Instead of just edges or “attributes”, entire objects such as a chair, can be recognized and classified. High-level feature extraction would not need to be used for this project, since more primitive features, like edges, will be the main classification point as all the images will be of hieroglyphics. Additionally, since optimization is a target of the project, it would be best to not introduce extra machine learning overheads where they can be avoided.

4.3.1.1 Edge Detection

Edge detection is the most popular method for feature extraction. Edge-based segmentation techniques can then be combined with edge-detection methods to streamline the process. The Sobel operator or the Canny operator both provide appropriate results as discussed previously. Most edge detection is achieved through either the gradient method, or the Laplacian method. The gradient method uses the first derivative of the image, searching for maximums and minimums and assuming that they signify edges. The Laplacian method uses the second derivative of the image and searches for zero-crossings to signify edges.



Result of using the gradient method (left) versus the Laplacian method (right). Source: <http://www.owl.net.rice.edu/~elec539/Projects97/morphjrks/moreedge.html>

4.3.1.2 Corner Detection

This is mostly used in applications where motion is involved, and thus will not be used for this project. It defines corners as the “intersection of two edges”. There are many algorithms that perform corner detection, such as Moravec corner detection algorithm, the Harris & Stephens corner detection algorithms, and the Forstner corner detector.

4.3.1.3 Blob Detection

Blob detection is more general than edge detection and can be used as a precursor to it. It can detect different areas based on differences such as color or brightness. These are areas where all the pixels are almost constant compared to each other, but different from other regions. This can be achieved through convolution and therefore is an attractive method for this project when compared with edge detection. OpenCV has several functions for blob detection.

```
Detector = cv2.SimpleBlobDetector()
```

```
Keypoints = detector.detect(image) //detects blobs
```

4.3.1.4 Ridge Detection

This detects ridges instead of edges. While an edge is a line that differentiates between homogeneous areas, a ridge is a line that is itself very different from the neighboring areas, for example, zebra stripes. This is not needed for this projects as the edges of hieroglyphics are homogeneous and only need to be detected and traced, they do not hold any extra information themselves.

4.3.1.5 Template Matching

In this method, a template is used against small parts of the image. This is also used for edge detection. The approach used could either be feature-based or area-based. This is an attractive method because it is fast and has a high accuracy, however the templates would have to be saved on the device itself which would greatly increase the storage requirements. Therefore, this method will not be used.



Using a closed eye as a template. Source: <http://ieeexplore.ieee.org/document/7462419/>

4.3.1.6 Hough Transform

By computing global descriptions of features, this transform can isolate certain shapes in an image, such as circles. This will not be needed in this project as the outline of the hieroglyphics will be the main descriptor and not geometric shapes.

The system will mainly use edge detection using the Canny operator as it provides optimal results, in addition to Hough transform and histogram based methods for segmenting and extracting rows or columns of hieroglyphs and individual hieroglyphs.

4.4 Classification Functions



A map of different machine learning algorithms. Source:

<https://s3.amazonaws.com/MLMastery/MachineLearningAlgorithms.png?s=1pwodfvqbnbbu2ib2q11>

4.4.1 Linear Regression

This is one of the simplest algorithms to use. It is a linear model, meaning that if there is an output variable, the algorithm assumes that it has a linear relationship with the input variables based on a linear equation. This equation describes all the variables that would affect the output multiplied by a

scale factor that represent how much this variable will affect the output (where a factor of 0 means that this variable has no effect on the output).

4.4.1.1 Ordinary Least Squares Regression

This is used when the linear regression has more than one input variable (which is the case for this project). It treats the data as a matrix and estimates optimal values for the coefficients. However, it needs large amounts of memory and that all the data be available at all times. The plus side is that it minimizes the the distances between each data line and the regression line squared, making it accurate.

4.4.1.2 Gradient Descent

This method assigns random values for the coefficients and a learning rate is used as a scale factor. The process is repeated reducing the sum squared error each time, again offering high accuracy but it requires an unpractically large amount of memory as it needs a very large dataset.

4.4.1.3 Regularization

This works on reducing both the error and the complexity of the model (the number of coefficients used in the equation). There is Lasso and Ridge variations of regularization, both modify the Ordinary Least Squares method.

4.4.2 Logistic Regression

Instead of a linear function, this uses a logistic function. Additionally, it limits the number of possible values unlike linear regression, which can have an infinite number of outcomes (negative, or very large values). It is mainly used when the response variables have discrete categories and are not continuous. Essentially, it is a “binary classifier algorithm” that “outputs the probability of the input belonging to a certain label”.

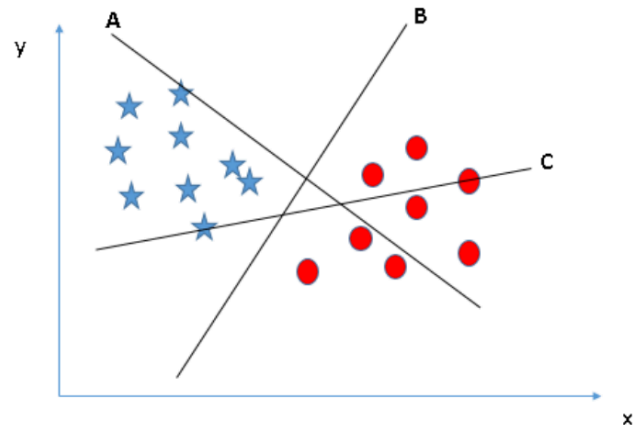
4.4.3 Decision Tree

Essentially, this method uses a binary tree, where the leaves are an output variable which is used to make a prediction. To reach a certain leaf, several split points are passed that are traversed based on categorizing input variables at each root node. This can either be stored in a graphical manner, or as a set of rules. A criterion for stopping is needed, usually a cap on the number of training instances for each leaf. Trees can be optimized through ‘pruning’, which is essentially observing the effect of removing each node on the accuracy of the prediction and removing nodes if they reduce the cost function without decreased accuracy.

4.4.4 Support Vector Machines

Mainly used for classification, SVMs search for the hyper-plane that best differentiates between two classes. Each feature is placed as a coordinate (support vector) and then the most appropriate hyper-plane is selected, prioritizing error rate in classification and ignoring outlier coordinates, also using kernels when needed.

An example of an SVM, where hyper-plane B is the optimal segregation plane. Source:



<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

4.4.5 Naive Bayes

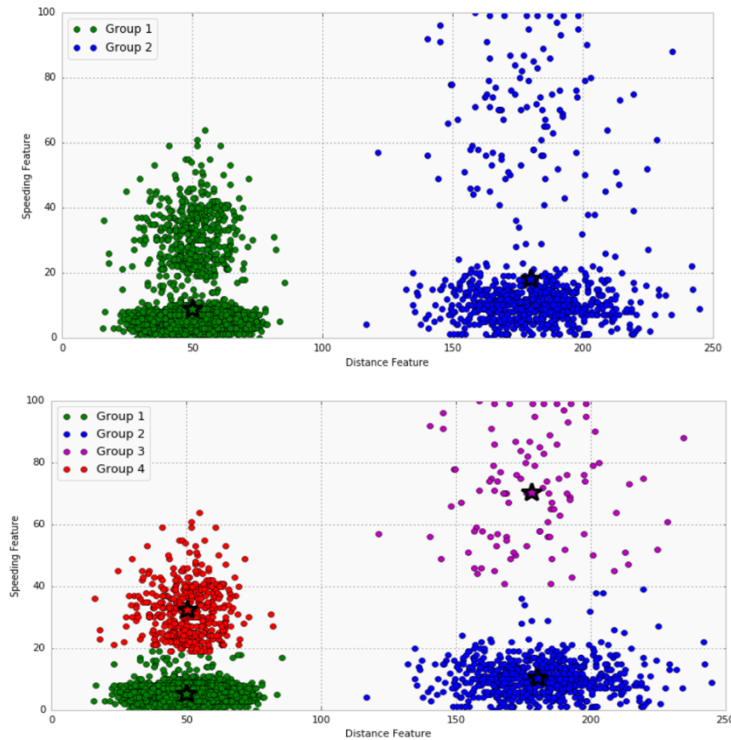
This technique is useful for large data sets and offers simplicity. It assumes that all features contributing to classifying a class are independent of each other and contribute to classifying this class in a certain category. By creating frequency and likelihood tables and then applying a Naive Bayesian equation, the probability of classification of a class can be calculated based on various features. While it is not the most accurate classifier, it is useful for real-time applications as it is very fast.

4.4.6 K-Nearest Neighbors

No learning is required, this technique simply uses all of the data set, requiring both space and consistency (low tolerance for outliers or data containing errors). It searches the data set for “the K most similar instances” based on the Euclidean, Hamming, Manhattan, or Minkowski distances between new and existing points. The output is the category with the “highest K most similar instances”. This requires heavy preprocessing of the dataset to make it as uniform as possible, and requires a lot of space.

4.4.7 K-Means Clustering

This method is useful for when data is not categorized. It works on grouping the data (where the number of groups is determined by K). It assigns each individual point to one of the K groups that can either be defined or be left to form organically.



K-Means Clustering with a) 2 groups and b) 4 groups. Source: <https://www.datascience.com/blog/k-means-clustering>

4.4.8 Random Forest

Multiple trees are created (the number of trees is directly proportional to the accuracy). Each tree follows the decision tree technique (each following a set of rules). Each tree randomly selects a set of features from the total pool of possible features and calculates nodes using split points. Each tree stores its predicted outcome and votes are calculated, the outcome with the highest vote count is considered as the final prediction, offering higher accuracy rates than using single decision trees.

4.4.9 Convolution Neural Networks

Convolution can be applied to images using a specific kernel to obtain a feature map. This removes “distracting information”, essentially transforming the inputs before feeding them into the algorithm. By using convolutional nets, the kernel will improve over time at knowing what to filter and what to leave to produce the optimal feature map.

Using convolution to retain only the shape of clothing to classify them and removing irrelevant color and pattern information. Source: <http://timdettmers.com/2015-03/26/convolution-deep-learning/>



4.4.10 Dimensionality Reduction Algorithms

With optimization in mind, this technique represents data sets having varying dimensions in a much more compact manner, ensuring that similar information is represented as efficiently as possible, reducing the storage space required. Missing values, as well as values with low variance and high correlation regarding each other are dropped (or grouped together), since they do not offer much in terms of accuracy and are therefore reducing the efficiency of the procedure.

4.4.10.1 Backward Feature Elimination

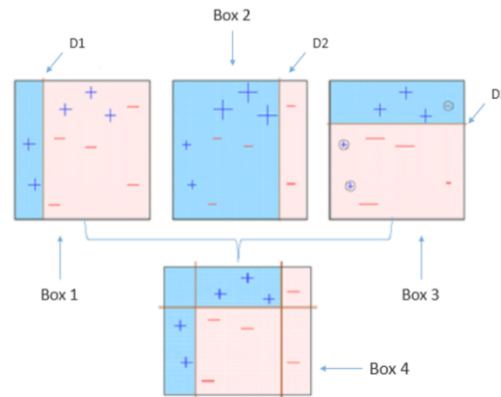
The sum of square of error is calculated while removing each variable, the variable that produced the smallest increase in the error are removed, decreasing the number of variables by one on each iteration. This is repeated as many times as possible. The reverse process can also take place (adding one variable at a time and noting its improvement on performance, taking the variable causing the highest improvement. This is called Forward Feature Selection).

4.4.10.2 Principal Component Analysis

Principal components are obtained out of linear combinations of the original variables, while trying to capture the largest amount of variation possible. Each principle component must be as orthogonal as possible to the others (have the largest amount of variance from them and within themselves).

4.4.11 Gradient Boosting Algorithms

This technique aims to strengthen learners by combining many classification rules that have weak implications on their own together to achieve a high overall prediction accuracy. It uses many different algorithms, giving equal weights to them initially, and increasing those weights with each error generated by an algorithm, finally combining their outputs to create a strong learner. The most common boosters are: AdaBoost, Gradient Tree Boosting, and XGBoost.



An example of using AdaBoost, several incorrect attempts at classifying “+”s and “-”s are combined to produce an accurate final classification. Source:

<https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/>

The exact classifier used by the system is still unknown, however the strongest candidates are linear regression, image descriptors, K-Nearest Neighbor, Convolutional Neural Nets, Support Vector Machines, and Principle Component Analysis.

4.5 Optimization Functions

4.5.1 Mobile GPUs

Presently, both iOS and Android devices contain embedded GPUs (Graphical Processing Units). Apple devices mainly use PowerVR, while Android most commonly uses Qualcomm’s Adreno and ARM’s Mali GPUs. While its main purpose is for graphics application, the GPU can be very helpful in boosting performance for ‘heavy’ algorithms and computations, such as those required for image processing, and the classifiers on the mobile device (GPU compute cores), speeding up processes and withstanding larger loads without crashing the device. “In the last years, high-end consumer graphics cards were more and more used to speed up the CNN training”. The newly released CUDA neural network library targets GPUs for mobile devices and supports smaller neural networks, also including a digital signal processor on board.

Smartphone	GPU model
Apple iPhone 7	PowerVR Series7XT Plus
Apple iPhone 7 Plus	PowerVR Series7XT Plus
Samsung Galaxy S8 - EMEA	Qualcomm Adreno 540
Samsung Galaxy S8 - USA and China	ARM Mali-G71 MP20
Google Pixel	Qualcomm Adreno 530
Google Pixel XL	Qualcomm Adreno 530
Sony Xperia XZ Premium	Qualcomm Adreno 540
LG G6	Qualcomm Adreno 530
HTC U11	Qualcomm Adreno 540
Huawei P10	ARM Mali-G71 MP8

The GPU models used in most recent mobile devices. Source: <https://deviceatlas.com/blog/most-used-smartphone-gpu-2017>

4.5.2 Support for Machine Learning Algorithms on Mobile Devices

Machine learning algorithms are already utilized within many applications on mobile devices. For example, some music players use algorithms to learn a user's preference and taste. Classifiers, such as SVM, can be designed on a device with little memory/power/etc. constraints, such as a laptop or a desktop computer, trained beforehand on said device, and then simply transferred onto the mobile device that evaluates it. This is particularly useful for larger network sizes that require more hardware capabilities or else will take longer computation time. There are currently five major frameworks for deep learning on mobile devices: Caffe, Torch, Theano, Deeplearning4j, and Tensorflow. They all utilize GPUs on the mobile devices.

4.5.3 Database Size

The larger the training set, the better the accuracy of results, therefore a large amount of data is needed. This requires large storage capabilities, as well as processing power to label the data (which is why this is usually not done on the mobile device itself, rather given to it ready). After training, only a small subset of labelled data is used for fine tuning. Less important weights can be simply ignored or de-duplicated. Certain features can also be grouped together if they are found to be correlational or non-orthogonal, reducing the number of variables taken into consideration.

4.5.4 Approximate Computing for Machine Learning:

This technique focuses on finding out the maximum degree of approximation that can still produce acceptable results. There are many methods for achieving this. Loop perforation discards certain iterations of calculations taking place inside a loop, reduced precision computation represents data in a smaller amount of bits (leading to less hardware load but reduced precision), and relaxed synchronization reduces the overheads of synchronizing across multiple cores to increase performance.

4.5.5 Optimizing Code

This is an ongoing challenge throughout development of the product. The logic used, as well as the number of libraries, APIs, etc. will all affect the efficiency of the application and must be taken into consideration.

4.5.6 Optimizing Images

There are currently several PNG optimizers that can reduce and compress image sizes, this will be particularly useful for the images stored in the database containing the Gardiner numbers of the chosen subset of hieroglyphics, as well as for the images stored in the users' history. These optimizers remove unneeded metadata and perform several types of reductions to reduce the size of the images without noticeably sacrificing the quality.

4.6 Social System Functions

4.6.1 Upload photo

The user will be able to upload photos along with certain required information (4.6.4 and 4.6.5), and this will be filtered (4.6.6) and then added to the online database. This can then be searched for by other users by using certain keywords to find it, or based on its location.

4.6.2 Comment on Photo

After searching for a photo and locating it, users can comment on photos as encouragement, or to ask questions about it. These comments will also be filtered (4.6.6).

4.6.3 Search for Information

Users can search using keywords or locations for photos posted by other users and can view them and comment on them (4.6.2).

4.6.4 Add Information

While posting a photo (4.6.1), users must add certain information about it, such as more details about where they found it, its translation or Gardiner's code if they know it, etc. This information will be stored in the database along with the photo.

4.6.5 Check In

While posting a photo (4.6.1), users must also add the location of the photo so that others can search for this location, or be presented with photos near them.

4.6.6 Filter Content

Any content added to the social system, whether photos or comments, will be filtered against profanity or verbal abuse to any other user.

4.7 Miscellaneous Functions

4.7.1 History

A history of the users' previous searches will be saved locally on the mobile device using Shared Preferences for the Android operating system and User Defaults for the iOS operating system and displayed on demand. This is stored locally and requires no internet access to view. It contains no sensitive information and therefore has no security requirements. It can be cleared by the user.

4.7.2 Camera

The application will be able to open the camera internally to allow users' to take a picture of hieroglyphics, or to access the photo gallery and give the application a previously taken picture of hieroglyphics. These images (whether taken at the present time or taken from the photo gallery) will then be parsed to the application to perform the necessary steps as discussed above.

4.7.3 Submit Image for Review

Professional users will have the opportunity to submit an image of a single hieroglyph along with its Gardiner code to be reviewed and then added to the training set for that hieroglyph to improve its validation accuracy (4.7.4), if the hieroglyph is one that the system does not currently classify, once enough images of it are submitted, it can be incorporated into the database of Gardiner codes and classified in the future (4.7.5).

4.7.4 Add Image to Training Data

An image submitted by a professional user can be added to the training data of a certain Gardiner code to increase validation accuracy for that subset, but must be verified by an expert panel first.

4.7.5 Add Image to Database

Once enough images have been submitted for a new subset of hieroglyphics that are not currently being classified by the system, they can be added to the Gardiner code database and their weights matrix can be added to devices so that they can be detected and classified, increasing the number of recognizable hieroglyphs by the system.

4.7.6 Display Results

At the end of the process, users will be given the Gardiner's code of the hieroglyphics that they have uploaded an image of if possible, or else they will be told that the hieroglyph is undetectable, either due to the fact that the system does not classify, or the image's quality is poor, this will be accompanied with a confidence level and a transliteration (if possible and an internet connection is present).

4.7.7 Display Confidence

When results are being displayed (4.7.6), a confidence level is also displayed describing how reliable the classification is, if the confidence level falls below a certain amount, this means the image has not been properly classified either due to the fact that the system does not classify, or the image's quality is poor.

4.7.8 Provide Transliteration

If there is an internet connection, the system might interface to a 3rd party transliterator and provide the user with a transliteration for the Gardiner's codes of the hieroglyphics in the picture taken by them.

4.8 Use Case Tables for Top Level Functions

4.8.1 Obtain Picture

<i>Feature ID</i>	<i>001</i>
<i>Feature Name</i>	Obtain Picture
<i>Main Actors</i>	End users
<i>Pre-Conditions</i>	User is on the application, but has not chosen or taken an image
<i>Post-Conditions</i>	User has selected or taken an image
<i>Normal Flow</i>	Open main menu -> open camera -> taken an image -> parse image to the application -> Preprocess Image (4.8.2)
<i>Alternative Flow</i>	Open main menu -> open gallery -> select a previously taken image -> parse image to the application -> Preprocess Image (4.8.2)

4.8.2 Preprocess Image

<i>Feature ID</i>	<i>002</i>
<i>Feature Name</i>	Preprocess Image
<i>Main Actors</i>	Image Preprocessing System
<i>Pre-Conditions</i>	Photo is unprocessed
<i>Post-Conditions</i>	Photo has received the needed preprocessing operations based on its state and is in its optimal form for the next stage
<i>Normal Flow</i>	Identify needed preprocessing operations -> perform them -> Segment Image (4.8.3)
<i>Alternative Flow</i>	None

4.8.3 Segment Image**Feature ID****003**

Feature Name	Segment Image
Main Actors	Image Segmentation System
Pre-Conditions	Image is preprocessed and contains multiple hieroglyphics
Post-Conditions	Each hieroglyphic is segmented individually and is in the agreed format to undergo classification
Normal Flow	Segment each hieroglyphic based on ordering scheme discussed -> Perform needed segmentation techniques to obtain the agreed image format for classification
Alternative Flow	None

4.8.4 Classify Image**Feature ID****004**

Feature Name	Classify Image
Main Actors	Machine Learning Classifier
Pre-Conditions	Each hieroglyph is segmented, its relevant features are obtained
Post-Conditions	The classifier has used the extracted features to make a prediction about the hieroglyph from the data set (weights database)
Normal Flow	Evaluate features -> predict hieroglyph from dataset (weights database) -> Present Image (4.8.5)
Alternative Flow	None

4.8.5 Present Image**Feature ID****005**

Feature Name	Present Image
Main Actors	User Interface
Pre-Conditions	Each hieroglyph has been predicted
Post-Conditions	The hieroglyphs are presented to the user with their name corresponding Gardiner number (from the Gardiner number database)
Normal Flow	Search for image in Gardiner number database -> Present image to user -> Add to History (4.8.6)
Alternative Flow	Search for image in Gardiner number database -> Inform user that hieroglyph is not recognized

4.8.6 Add to History**Feature ID****006**

Feature Name	Add to History
Main Actors	User Search History
Pre-Conditions	User has obtained the Gardiner numbers for the hieroglyphics they have taken pictures of
Post-Conditions	The picture taken and Gardiner numbers present are stored in Shared Preferences/User Defaults
Normal Flow	Finish search -> Add image and Gardiner numbers to history
Alternative Flow	None

4.8.7 Open History**Feature ID****007**

Feature Name	Open History
Main Actors	End users
Pre-Conditions	User is on the main menu page
Post-Conditions	User has their search history displayed
Normal Flow	Open main menu -> Open History
Alternative Flow	Open main menu -> Open History -> Clear History (4.8.8)

4.8.8 Clear History**Feature ID****008**

Feature Name	Clear History
Main Actors	End users
Pre-Conditions	User has their search history displayed
Post-Conditions	User has their search history deleted (now empty)
Normal Flow	Open History -> Clear History
Alternative Flow	None

4.8.9 View Results

Feature ID	009
Feature Name	View Results
Main Actors	End users
Pre-Conditions	User has selected or taken an image
Post-Conditions	User is shown the image with its symbols classified and transliteration (if possible) and confidence level are provided.
Normal Flow	Classify hieroglyphs -> Interface to transliterator -> Show results
Alternative Flow	None

4.8.10 Choose & Submit Image to Database

Feature ID	010
Feature Name	Choose & Submit Image to Database
Main Actors	Professional Users
Pre-Conditions	User has selected or taken an image
Post-Conditions	User has submitted the image for review
Normal Flow	Open main menu -> Open camera -> Take a photo -> Parse photo to application -> Preprocess Image -> Send for review
Alternative Flow	Open main menu -> Open gallery -> Choose a previously taken photo -> Parse photo to application -> Preprocess Image -> Send for review

4.8.11 Review Images

Feature ID	011
Feature Name	Review Images
Main Actors	Review Panelist
Pre-Conditions	Professional user has submitted an image
Post-Conditions	Image either rejected or admitted into training data set or Gardiner code database
Normal Flow	Retrieve image pending review -> Approve photo to dataset or database
Alternative Flow	Retrieve image pending review -> Reject photo to dataset or database

4.8.12 Post Images

Feature ID	012
Feature Name	Post Images
Main Actors	Professional Users
Pre-Conditions	User has not taken or chosen an image
Post-Conditions	Image is posted and added to the social content database
Normal Flow	Open Camera -> Take a photo -> Parse photo to application -> Upload to social content database
Alternative Flow	Open gallery -> Choose a previously taken photo -> Parse photo to application -> Upload to social content database ne

4.8.13 Comment on Posts**Feature ID****013**

Feature Name	Comment on Posts
Main Actors	Professional Users
Pre-Conditions	User searched for a photo and obtained a result
Post-Conditions	User's comment is posted
Normal Flow	View photo -> Write comment -> Post comment -> Add comment to social content database
Alternative Flow	None

4.8.14 Search for Posts**Feature ID****014**

Feature Name	Search for Posts
Main Actors	Professional Users
Pre-Conditions	User is in the social section of the application
Post-Conditions	User finds posts relevant to keyword(s) they entered or location they are in
Normal Flow	Enter keyword or choose search by location -> Search database -> Return results
Alternative Flow	None

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The software should be highly available due to the fact that it is not limited by availability of internet.
- It should be as reliable and accurate as possible while not sacrificing size and speed, the option to improve accuracy by using the internet will allow us to reach a balance between both. A confidence level will also be displayed in case the user took a blurry or unsuitable image.
- It should be as optimized, compact and efficient as possible, while not sacrificing reliability and accuracy.
- The maximum tolerated response time should be realistic, we will extract this from similar applications like Google translate.

5.2 Safety Requirements

While there is no danger to the user, using flash to take pictures for the application might damage certain artifacts, the system will display a warning for users to stay alert about this. Additionally, the system will have a method in place to filter all content posted using the social features of the application against profanity or any unsuitable content to protect the mental wellbeing of users.

5.3 Security Requirements

Privacy and security are very high within the system due to the fact that there is no private or sensitive information being shared with the application. There will be no payment information collected and users can sign up with fictitious usernames. The system will use Firebase to handle accounts and passwords, which is very secure and will make no attempt to track users if a check in option is added in the social features.

5.4 Software Quality Attributes

- The system will be portable and will work on both Android and iOS.
- Maintenance costs should be reasonable, within 80% of the system's development costs.
- Robustness is very important, databases, and any stored information should never be corrupted or lost upon crashes or system failures (99.99% success rate).
- The software should be very easy to use, requiring minimal help prompts during first session in order to become familiar with it.

5.5 Business Rules

- Only Dua-Khety administrators/developers can access the databases. A review panel must accept professional user submissions. All content posted on the social system must be filtered first.

6. Other Requirements

Weight matrices and a database of Gardiner numbers will be required in order to correctly classify the selected subset of Hieroglyphics. The project should be as modular as possible to promote re-use.

Appendix A: Glossary

Bag of Words:

This machine learning technique is used to represent data that is textual. It measures the presence of a vocabulary of known words in a document, discarding any information concerning the order or location of the words, hence being referred to as a “bag” of words. It produces document vectors of the number of occurrences of each word in the selected vocabulary.

Descriptors:





Essentially, these are representations of elements or variables used by the machine learning algorithm. A descriptor vector for an image is one that would not be changed even after the image had been scaled, rotated, etc.

Feature Extraction:

In the scope of machine learning, a feature is an observable property of an element in a data set. A set of these features are chosen to enable classification algorithms to function properly. They should be orthogonal of each other and discriminating of the element they represent. Essentially, before classification or pattern detection/recognition can take place, the data set must be converted into a set of features. Selecting and extracting the features that lead to the highest classification accuracy is a matter of experimentation and combination of different techniques.

Gardiner Number:

Sir Alan H. Gardiner organized and classified hieroglyphics into categories, where each symbol has a corresponding unique identification code referred to as its Gardiner number.

A. Man and his Occupations	B. Woman and her Occupations	C. Anthropomorphic Deities	Gardiner Number	Hieroglyph	Description of Glyph	Details
D. Parts of the Human Body	E. Mammals	F. Parts of Mammals	A1		Seated man	Det. of man, names; Pronoun1st sing. <i>i, wi, ink, kwi.</i> "I," "me," "my."
G. Birds	H. Parts of Birds	I. Amphibious Animals, Reptiles, etc.	A2		Man with hand to mouth	Det. of eat, drink, speak, think.
K. Fish and Parts of Fish	L. Invertebrates and Lesser Animals	M. Trees and Plants	A3		Man sitting on heel	Det. of sit.
N. Sky, Earth, Water	O. Buildings, Parts of Buildings, etc.	P. Ships and Parts of Ships	A4		Man with arms raised	Det. adoration, hide

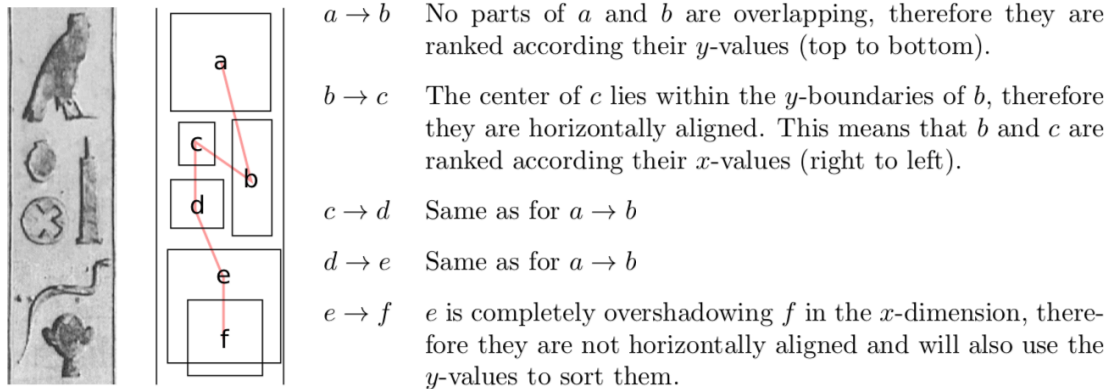
Some of the categories of hieroglyphics, and some of the Gardiner numbers of the first category.

Source: <http://www.egyptianhieroglyphs.net/gardiners-sign-list/>

GPU:

The Graphics Processing Unit is a processor that is solely dedicated to improve the performance of graphics intensive applications. Recently, it has also been used to improve the performance of computationally heavy processes such as machine learning algorithms.

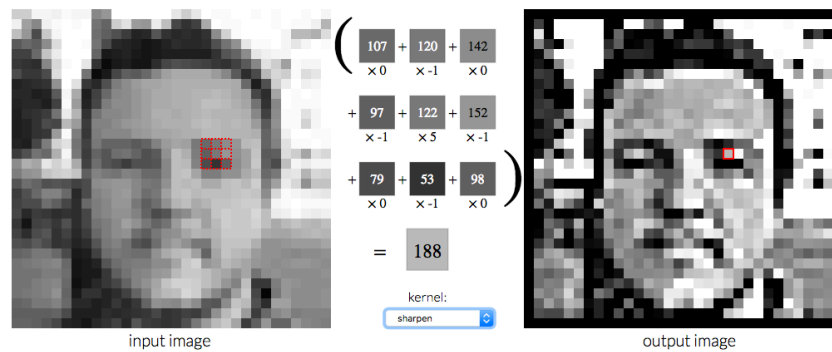
Hieroglyphic Ordering:



Source: Franken & Gemet, Automatic Egyptian Hieroglyphic Recognition by Retrieving Images as Texts

Kernel Filter:

This is a small matrix that is used in image processing to perform a certain preprocessing operation to an image, such as sharpening, edge detection, etc. It does so by convoluting the target image with the kernel.



An example of image sharpening through using a filter kernel. Source: <http://setosa.io/ev/image-kernels/>

Machine Learning:

This is the science of training computers using data to act on their own judgment without explicitly being programmed to behave a certain way. It also allows computers to make predications based on previous data ‘studied’ with varying techniques and accuracies.

Matlab:

Matlab (which is short for Matrix Laboratory) is a software that enables users to perform various matrix manipulations, algorithms, digital signal processing, plotting, etc. in multiple programming languages (including its own).

OpenCV:

OpenCV (which is short for Open Source Computer Vision) is a library that supports deep learning frameworks Caffee and TensorFlow (discussed in section 4) and provides real-time computer vision and digital signal processing programming functions.

Scipy:

This is a Python collection of open-source software that includes many packages used for multiple mathematics and science application. Its most famous packages are NumPy, Matplotlib, IPython, and Sympy.

Sum of Square of Error (SSR):

This is a measure of the differences between a certain estimation model and the actual input data. As the name suggests, it calculates the sum of the squares of deviances between each point of empirical data and the corresponding point on the estimation model. It is commonly used to determine the optimality of a certain parameter for selection.

SVM:

This is short for Support Vector Machines and is a supervised machine learning model (meaning that it uses labelled training sets, as opposed to unsupervised models that use unlabeled training sets) that is given a training set where each element is marked as belonging to a certain category. Given new data, the model can then categorize it itself.

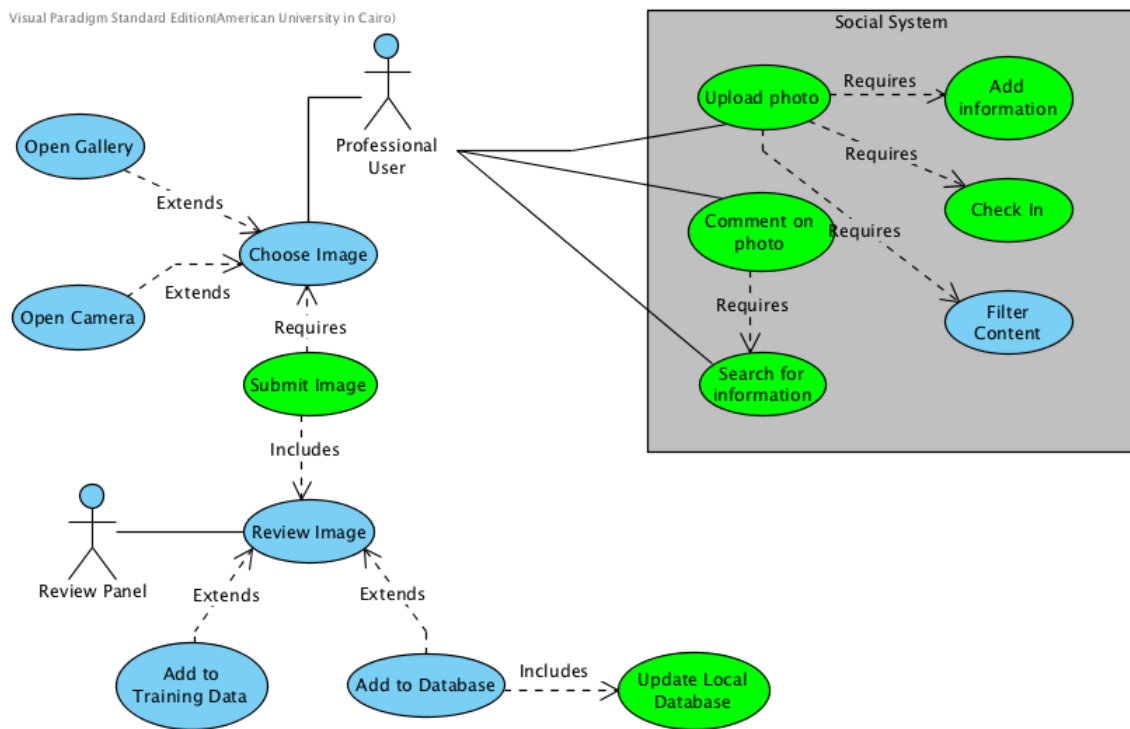
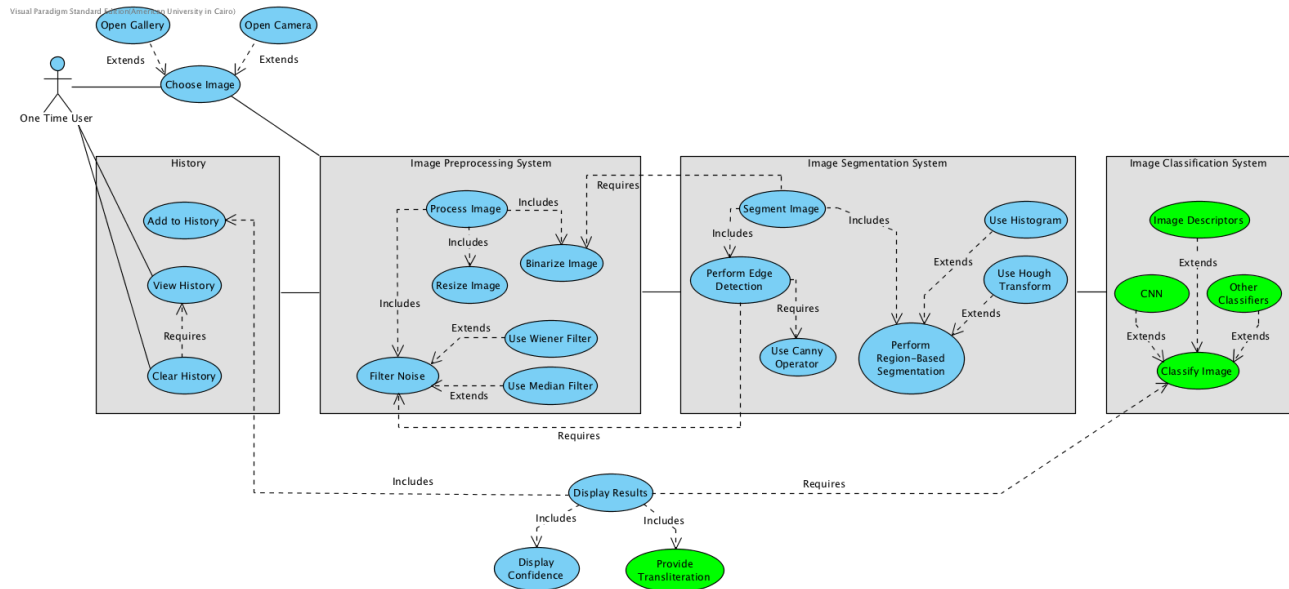
User Defaults/Shared Preferences:

In iOS, User Defaults is where key-value pairs are stored. These are “persistent across launches of [the] app.” Similarly, Shared Preferences are used in Android for defining a set of preferences that remain in a consistent state throughout multiple uses of the app.

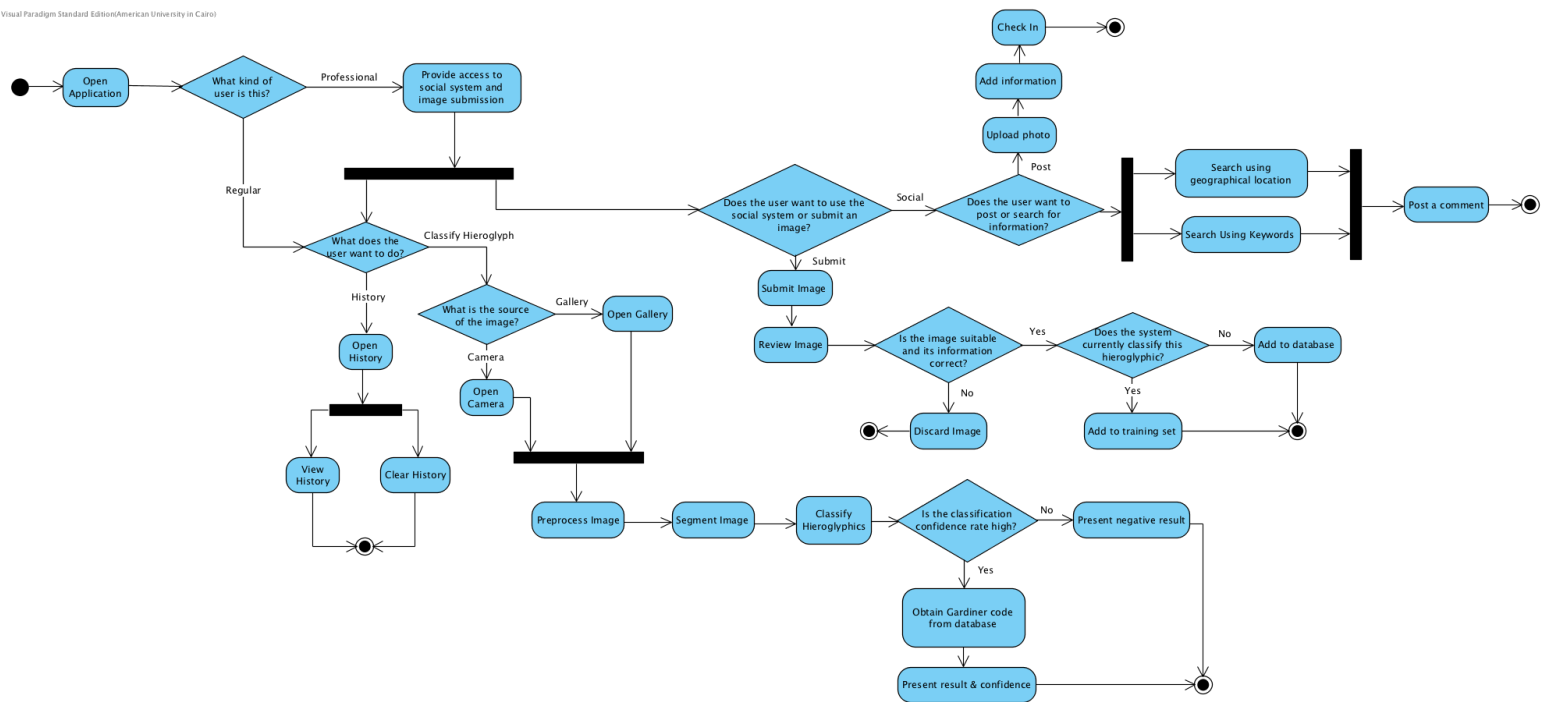
Weights:

In machine learning, the weight of a node, is the strength of influence that that node has on the outcome of the prediction/classification. All nodes initially have equal weights, and then during the training process each node’s weight is adjusted based on the relationship between it and its corresponding element in the output.

Appendix B: Analysis Models



Visual Paradigm Standard Edition/American University in Cairo



Activity Diagram Covering Decision Flow for the Entire System

Appendix C: Sources & Useful Material

Pre-Processing:

<http://www.rroij.com/open-access/an-overview-on-image-processing-techniques.php?aid=47175>

<https://www.quora.com/What-are-some-ways-of-pre-processing-images-before-applying-convolutional-neural-networks-for-the-task-of-image-classification>

<https://datascience.stackexchange.com/questions/5224/how-to-prepare-augment-images-for-neural-network>

<https://math.berkeley.edu/~sethian/2006/Applications/ImageProcessing/noiseremoval.html>

https://docs.opencv.org/3.2.0/d5/d69/tutorial_py_non_local_means.html

<https://www.mathworks.com/matlabcentral/answers/109517-image-normalization-in-the-range-0-to-1>

<http://felixniklas.com/imageprocessing/binarization>

Segmentation:

<http://cdn.intechopen.com/pdfs/11405.pdf><http://mmi.tudelft.nl/pub/leon/AGChitu.pdf>

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.412.1479&rep=rep1&type=pdf>

<https://pdfs.semanticscholar.org/ee0a/414f7b4a7a16c8f8ef8c9437f59aeced4cef.pdf>

http://homes.di.unimi.it/ferrari/ImgProc2011_12/EI2011_12_16_segmentation_double.pdf

<https://www.youtube.com/watch?v=VaR21S8ewCQ>

<https://www.mathworks.com/matlabcentral/fileexchange/27291-thinning-image?focused=5152062&tab=function>

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

Feature Extraction:

<https://stackoverflow.com/questions/26590705/difference-between-low-level-and-high-level-feature-detection-extraction>

https://en.wikipedia.org/wiki/Corner_detection

https://en.wikipedia.org/wiki/Blob_detection

<https://www.learnopencv.com/blob-detection-using-opencv-python-c/>

<https://dsp.stackexchange.com/questions/1714/best-way-of-segmenting-veins-in-leaves>

<https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm>

Classification:

<https://machinelearningmastery.com/linear-regression-for-machine-learning/>

<https://stackoverflow.com/questions/12146914/what-is-the-difference-between-linear-regression-and-logistic-regression>

<https://machinelearningmastery.com/logistic-regression-for-machine-learning/>

<https://stackoverflow.com/questions/12146914/what-is-the-difference-between-linear-regression-and-logistic-regression>

<https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/>

<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

<https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>

<https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/>

<https://www.datascience.com/blog/k-means-clustering>

<http://timdettmers.com/2015/03/26/convolution-deep-learning/>

<https://www.analyticsvidhya.com/blog/2015/07/dimension-reduction-methods/>

<https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/>

Optimization:

<http://www.embedded-computing.com/embedded-computing-design/understand-the-mobile-graphics-processing-unit>

<https://deviceatlas.com/blog/most-used-smartphone-gpu-2017>

https://www.nst.ei.tum.de/fileadmin/w00bqs/www/publications/as/2015WS-HS-Deep_learning_mobile_platforms.pdf

http://www.ijera.com/papers/Vol3_issue6/EW36913917.pdf

<http://machinethink.net/blog/machine-learning-device-or-cloud/>

<https://docs.unity3d.com/Manual/MobileOptimizationPracticalGuide.html>

<https://www.appticles.com/blog/2016/04/websites-are-couchpotatoes-images-are-donuts-and-you-have-to-do-something/>

<https://eclipsesource.com/blogs/2013/08/12/optimizing-images-for-mobile-native-and-web-apps/>

Glossary:

<https://machinelearningmastery.com/gentle-introduction-bag-words-model/>

<https://www.quora.com/What-is-mean-by-weight-in-machine-learning>

<https://developer.android.com/reference/android/content/SharedPreferences.html>

<https://developer.apple.com/documentation/foundation/userdefaults>

https://en.wikipedia.org/wiki/Supervised_learning

https://en.wikipedia.org/wiki/Support_vector_machine

https://en.wikipedia.org/wiki/Residual_sum_of_squares

<https://www.scipy.org/>

<https://en.wikipedia.org/wiki/OpenCV>

<https://en.wikipedia.org/wiki/MATLAB>

<https://www.coursera.org/learn/machine-learning/lecture/Ujm7v/what-is-machine-learning>

[https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/feature_extr.htm#i1005920

<https://www.igi-global.com/dictionary/feature-extraction/10960>

[https://en.wikipedia.org/wiki/Feature_\(machine_learning\)](https://en.wikipedia.org/wiki/Feature_(machine_learning))

<https://arxiv.org/abs/1709.01666>

<https://cs.stackexchange.com/questions/51373/what-is-the-difference-between-features-and-descriptors-in-computer-vision>

<https://machinelearningmastery.com/gentle-introduction-bag-words-model/>