Geo-information Science and Remote Sensing

Thesis Report GIRS-2019-40

IMAGE RECOGNITION FOR IRRIGATION MANAGEMENT IN COSTA RICA

Thedmer Postma

October 8, 2019

WAGENINGEN
UNIVERSITY & RESEARCH

Image Recognition for Irrigation Management in Costa Rica

Thedmer Postma

Registration number 92 08 09 667 050

Supervisors:

dr. ir. Sytze de Bruin

A thesis submitted in partial fulfilment of the degree of Master of Science
at Wageningen University and Research Centre,
The Netherlands.

October 8, 2019
Wageningen, The Netherlands

Thesis code number: GRS-80436
Thesis Report: GIRS-2019-40
Wageningen University and Research Centre
Laboratory of Geo-Information Science and Remote Sensing

Abstract

The Costa Rican national service for groundwater, irrigation and drainage (SENARA) is aiming to improve its irrigation management structure and processes. This research explored how Geographical Information Systems (GIS) in combination with Information and Communication Technologies (ICT) can be applied for creating a spatial database infrastructure based on an Android application for the accounting and analysis of water used for irrigation. A system was created for the automated reading and storing of water consumption values from water meters. An Android application was developed for taking and uploading photos of a water meter's dial to a server for determining the indicated water consumption via image processing. Two water consumption values were read from the dial: the amount of water assigned to the user for irrigation indicated by a pointer and the user's total consumption, indicated by a cumulative tally counter. Image segmentation was used for the detection and reading of the pointer value, achieving readings with a mean absolute error of $2m^3$ (1 percent relative to the scale). The tally counter was read via digit recognition based on Fourier descriptors achieving an accuracy of 76.6 percent. Additionally, a spatial database was set up for the storing of the water consumption data. The created system offers potential for the accounting of the water consumption of single users and performing regional analysis on water consumption.

Acknowledgements

First of all, I want to thank SENARA and Eduardo Soto in particular for allowing me to work on an interesting project and providing me with all the data and background information I needed.

Additionally, I would like to express my gratitude towards my supervisor Sytze de Bruin for his help, patience, guidance and feedback during the entire process of writing this thesis; It was much appreciated.

Finally, I would like to thank Diego Marcos Gonzalez for his advise on digit recognition. Your suggestion of the Fourier transform turned out to be an indispensable part of this research.

Contents

1. Introduction

1.1 Context and Background

Climate change is increasingly affecting agricultural practices in Costa Rica as Central America is one of the focal points of climate change in the tropical zones (Solano et al. 2012). Future scenarios indicate it to be very likely that precipitation will either decrease or change dramatically. Some predictions show that the traditional dry season will become a lot more wet while the wet season would see a strong decrease in precipitation. Drying trends are already noticeable in the entire region of middle America and the Caribbean (Neelin et al. 2006). In Costa Rica, this is most noticeable in the North-Western part of the country where there is already an increase in the severity and frequency of droughts (Birkel & Demuth 2006).

With this future scenario in mind, careful and efficient water management will be indispensable to keep a productive agricultural sector. The agricultural sector in Costa Rica is diverse and consists of a mix of large and small farms (Babcock et al. 2016). Small farmers typically are family businesses and rent land from big farms. They either raise cattle or grow crops that largely end up on the local market. Large farms, on the other hand, focus on the cultivation of cash crops (e.g, melons, sugar cane and rice) or breeding cattle. They typically employ multiple laborers and engineers. Large farms as well as some smaller farms use irrigation systems. Based on locational factors, a mix of rainfall, groundwater and river water is used as a water source for these irrigation systems.

SENARA

In Costa Rica, several authorities and associations are responsible for different parts of the water distribution process. In total there are 13 actors involved with water governance, seven of them with the design and implementation and six with the regulation of water distribution practices (OECD 2012). One of these actors is the Servicio Nacional de Aguas Subterráneas, Riego y Avenamiento (SENARA), the Costa Rican national service for groundwater, irrigation, and drainage. SENARA is responsible for providing water to the agricultural sector for irrigation (de Bruin et al. 2017).

SENARA was founded in the year 1983 after a reorganization within the Costa Rican government that aimed to create a central institution to promote agricultural development by the establishment and operation of irrigation and drainage systems (Alvarado 2003). SENARA focuses on irrigation, drainage, flood prevention, and groundwater research and conservation. Some of SENARA's tasks of particular interest for this research are:

- Developing and executing a policy of water use and distribution for agricultural purposes while keeping in mind the land use and natural resources of the area.

- Developing and maintaining irrigation, drainage and flood control systems.
- Optimizing the use and distribution of water for irrigation to increase and diversify agricultural production.

SENARA manages two irrigation programs, the "Distrito de Riego Arenal Tempisque" (DRAT) and the Small Irrigation and Drainage Areas Program (PARD). DRAT is the largest, covering around 28.000 ha with the "Laguna de Arenal" as it's source of water. DRAT is executed throughout the national territory and, with an area of around 3500 ha of irrigation service is considerably smaller

1.2 Problem Definition

Currently, the government's hydrological information management in Costa Rica is not optimal. Data are dispersed, inconsistent, outdated or inaccessible (Córdoba et al. 2016). This is especially unfortunate as in the current situation, the available data is not used to its full potential for planning and risk assessment. However, efforts are made to improve this situation. SENARA, for one, aspires to modernize its practices through the use of geospatially-enabled technology, like Geographical Information Systems (GIS). At present their implementation of GIS technologies is limited to the presentation of projects only. However, intentions exists to explore the possibilities that GIS technologies offer for the improvement of its practices (de Bruin et al. 2017).

The overall aim of this thesis is to explore how GIS in combination with Information and Communication Technologies (ICT) can help SENARA improve their irrigation management structure and processes. As GIS are capable of handling large amounts of spatial and remote sensing data, integrating databases and statistics as well as creating high-quality maps for visualizations and analysis, they have proved extremely useful for facilitating irrigation management tasks (Acharya et al. 2014). This is supported by Soto-Garcia et al. (2013), who investigated the influence of implementation of ICT and GIS on a local irrigation management system in Cartagena, Spain, and reported improvements for equitable water allocation and traceability and transparency of water consumption.

Many examples of irrigation management systems exist in which GIS is used for data gathering, processing, analysis, and visualization. For example, Todorovic & Steduto (2003) developed a GIS-based irrigation management system at the beginning of the 2000s for local governments and irrigation consortia in Apulia, Italy. The application itself consisted of an irrigation module that enabled the evaluation of irrigation scenarios under different soil, climatic and management conditions. Saravanan et al. (2019) used GIS to calculate the irrigation needs of the region of Pallumbadi in Tamil Nadu, India. Rey et al. (2016) used GIS to estimate the economic benefits of outdoor irrigated production in the UK. Valverde-Arias et al. (2018) generated a drought risk map using GIS tools, enabling the assessment of the vulnerability of rice crops in the Babahoyo Canton in Ecuador.

Besides GIS, the implementation of ICT techniques in mobile applications for hydro informatics also offers a wide range of opportunities. Within the European Union, the latest ICT developments have been recognized as tools to improve inter-agency cooperation and data quality within organizations concerned with the collection and provision of environmental data

(Jonoski et al. 2013).

De Bruin et al. (2017) explored a few of the possible ways in which GIS and ICT could benefit SENARA. They developed a mobile application that allowed users to report technical problems with the hydrological net, such as the breaking of a tube. Executives of SENARA expressed enthusiasm about the mobile application and great interest in extending the use of mobile apps for more of their activities.

As mentioned earlier, SENARA is responsible for the distribution of water for farmers to irrigate their crops. Maintenance and operation of the irrigation systems are performed by Agricultural User's Associations (SUA), whose members may be owners or tenants of land, under the supervision of SENARA. At the moment there are 76 SUA's that have the main responsibility of taking care of the collective use of public water and the building and maintenance of irrigation systems (FAO 2015). The water meters are visited by SUA staff regularly to set the amount of water for the farmers to use in the upcoming period. Water is allocated to users, who get billed for each m^3 of water consumed. Each user has a water meter that regulates and registers the amount of water consumed from the hydrological network. An example of the water meters used by SENARA is shown in 1.1.

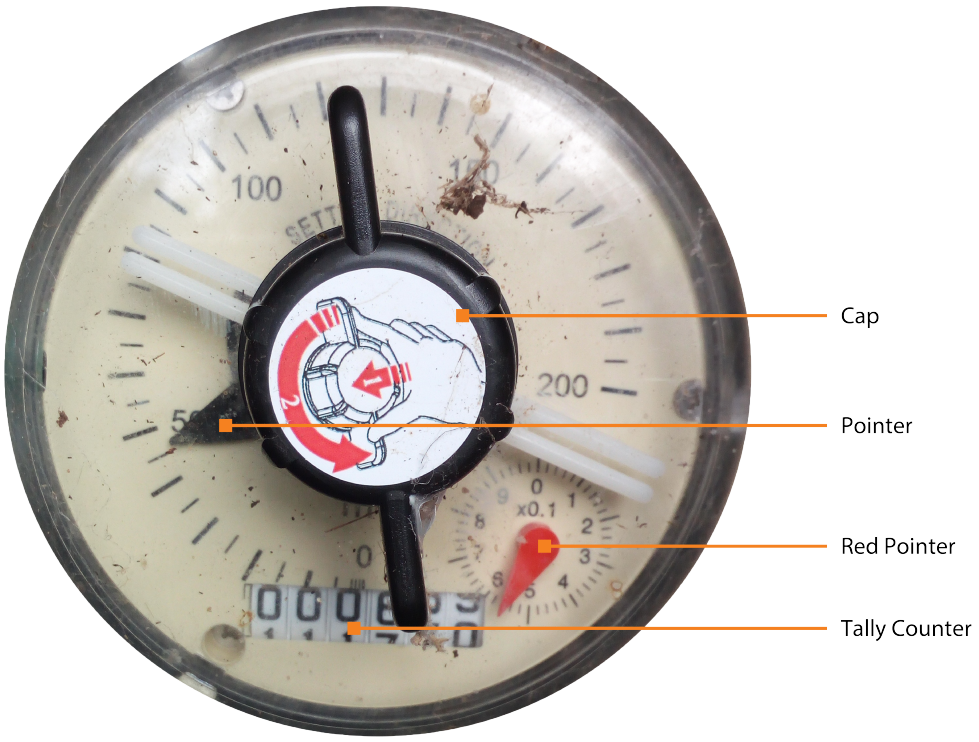


Figure 1.1: Water meter used by SENARA, with the names of the parts relevant to this research

The process of water allocation works as follows: the amount of water allocated for irrigation is indicated by the black pointer. In the example provided here, it is set to 46 m3. By turning the black cap with the two handles, the amount of water available for consumption can be modified. The number on the counter on the lower part of the meter is the cumulative amount of water consumed at this station. The red pointer indicates whether water is flowing through the meter.

At the moment, there is no automated system for the processing and storage of data on water usage. SENARA is interested in finding a way to gather and store this data in a way that makes the management process more efficient. This thesis focuses on the development of a mobile phone application for reading the water consumption of individual water meters and the storage of these data in a central database. This will enable monitoring of water consumption and visualization of how water consumption is distributed over irrigation systems.

The easiest solution would be to create an application where the SUA members can directly input the water consumption values shown on the water meters to be stored into a central database. However, as mentioned earlier, the SUA's are not a part of SENARA and made up of local users. Because of this, it is preferable to develop a system that independently determines water consumption values to prevent manipulations and improve transparency and verifiability of water consumption. Soto-Garcia et al. (2013) found that the transparency offered by a central tool for water allocation reduced the number of conflicts between farmers, mainly because it was possible to check every step of the water allocation process.

By gathering and processing water consumption data through an application, opportunities arise for analysis of the hydrological practices in the region. The incorporation of GIS could offer SENARA new and unexplored opportunities to improve the functioning of their organization. GIS is now only being used to present projects. Readily available data provided by the app offer possibilities to also improve decision making and planning activities for SENARA.

1.3 Research Objectives and Research Questions

SENARA wants to implement a spatial database infrastructure supported by an App for accounting and analysis of irrigation water. The main objective of this thesis is to contribute to SENARA's goal of creating a spatial database infrastructure based on an Android Application for the accounting and analysis of the water used for irrigation. To do this the following research questions will be answered.

1. Which functionality is promising for a simple Android application that allows the automated reading of water meters?
2. What accuracy is achieved to read pointer values indicating water gifts?
3. With what accuracy can the digits of the cumulative tally counters be read?
4. How can the application be used within the context of a spatial database infrastructure?

2. Theoretical Background

This chapter provides a brief overview of literature covering the use of image processing techniques for the estimation of gauge values as well as digit recognition. Next, a theoretical background is provided for the methods that were applied during the thesis research.

2.1 Review

Automated digit recognition has been a topic that has been studied extensively throughout the years, with successful results. Perhaps one of the most well-known achievements on this topic comes from LeCun et al. (1990) who designed a system of neural networks to identify handwritten digits. It performed very well with an error rate of only 1 percent. Approaches for digit recognition based on boundary features using so-called Fourier descriptors achieved similar results. Shridhar & Badreldin (1984) used a combination of Fourier and topological descriptors to classify handwritten digits achieving an accuracy of 99 percent. In a review of five different Fourier descriptors representations, Lu et al. (1993) reported accuracies ranging from 98.6 percent to 99.8 percent.

Digital Image processing – which as the name suggests refers to the processing of digital images using a digital computer (Gonzalez & Woods 2002, pp.23) – has been successfully implemented for the reading of dial and gauge values in several cases. Ye et al. (2013) used a combination of binarization, image thinning and the Hough transform to determine the value of a pointer type pressure gauge. Jaffery & Dubey (2012) created a sliding window algorithm to extract pointer values based on canny edge detection and feature matching. Yao et al. (2014) successfully implemented a combination of Hough circle detection methods and centroid detection methods to create a pointer recognition algorithm for pressure gauge pointers.

For this thesis project, the goal is to combine both the pointer detection and the digit recognition into a single image processing algorithm. The aforementioned examples show it is possible to achieve good results for the automated reading of the water meters. However, one considerable difference between this project and the examples is that the environmental properties of the water meters are unstable. Depending on the time of day, weather type and the general condition of the water meter the inputs for the image processing algorithm can vary greatly. To deal with this in the best way, a robust preprocessing process is needed.

2.2 Image Segmentation

Image segmentation is one of the most important components of the image processing done in this thesis project. In this section, a brief explanation will be given for a better understanding

of the methodology and results.

In the context of image processing, segmentation is used to divide an image into regions based on the intensity values of the image's pixels. When segmentation is performed with the goal of object detection, the segmented regions represent potential objects. Segmentation is generally performed based on discontinuity or similarity of the intensity values. Segmentation based on discontinuity divides an image based on sharp changes of intensity, for example, edges obtained via edge detection. Segmentation based on similarity is most often done via thresholding, popular because of its speed, simplicity and intuitive approach (Gonzalez & Woods 1992, pp.760-774). As thresholding is the segmentation technique used in this thesis research, the coming section provides an extensive explanation of it.

Thresholding

The basic principle of thresholding is best explained using an example. Figure 2.1 represents the density plot of a grayscale image $f(x, y)$ containing a light object and a dark background.

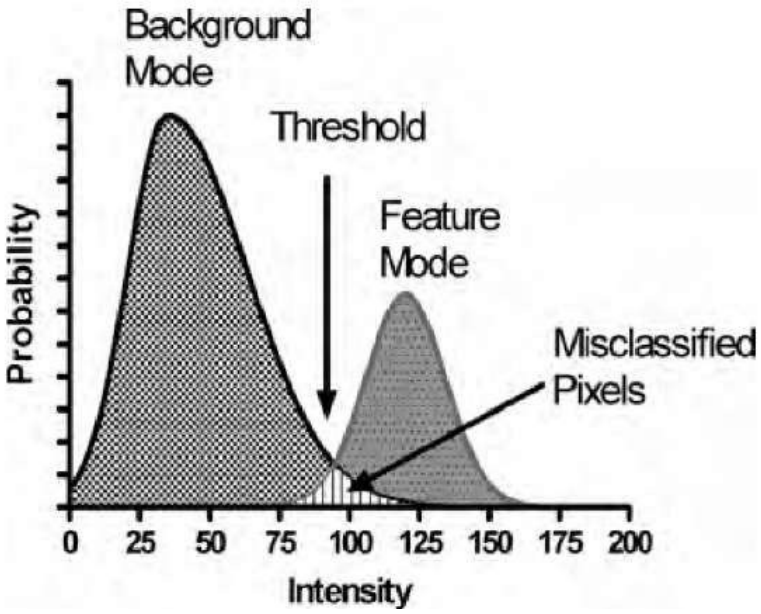


Figure 2.1: Example of object detection using image thresholding on the bimodal density plot of an image (Gonzalez & Woods 1992)

The given density plot is bi-modal and as a result, the object in the image can be segmented by selecting a threshold value T that separates the two modes. The result would be an image where any pixel that fits $f(x, y) > T$ gets labeled as a pixel belonging to the object, while pixels that fit $f(x, y) \leq T$ get labeled as background pixels. The thresholding operation can be portrayed as tests against a function T, defined as

$$T = T\left[x,\, y,\, p(x, y),\, i(x, y)\right] \tag{2.1}$$

where $p(x, y)$ represents a local property of the pixel, e.g. the average gray value within a specified neighborhood around (x, y), and $i(x, y)$ represents the value, for example the gray value, of the image $f(x, y)$ at location (x, y). The result of a thresholding operation, the

segmented image $g(x, y)$, can be defined as

$$g(x,y) = \left\{ \begin{array}{ll} 1 & if\ f(x,y)\ >\ T \\ 0 & if\ f(x,y)\ \leq\ T \end{array} \right\} \tag{2.2}$$

where the pixel label 1 indicates object pixels and pixel label 0 indicates background pixels, meaning that image $g(x, y)$ is a binary image (Gonzalez & Woods 1992, pp.596). The threshold T is called global when it is based solely on $i(x, y)$ and local when T is based on both $i(x, y)$ and $p(x, y)$.

Determining Threshold Values

Most images encountered in real-life situations are more complex than the simple bi-modal example shown in figure 2.1, and determining the best value for threshold T can be a challenging task. Luckily a wide range of automated methods have been developed to determine the best threshold value. One of the most famous threshold determination methods for bi-modal images is Otsu Thresholding (Otsu 1979). This method sets a threshold value based on the properties of the histogram of an image, minimizing the inter-class variance in between the object and the background pixel class.

Li & Lee (1993) applied a similar approach to determine the best threshold value by minimizing the cross-entropy between the input image and it's segmented version. This method gives better results than Otsu's threshold when the object and background classes have unequal sizes and highly unequal variance. Li & Tam (1998) incorporated the minimum cross-entropy thresholding method into a fast iterative algorithm, creating a valid alternative for Otsu's thresholding algorithm. For detailed explanations of the workings of the Otsu algorithm and the minimum cross-entropy algorithm, see the work of Otsu (1979), Li & Lee (1993) and Li & Tam (1998).

2.3 Color Space

Image segmentation is performed based on the intensity values of an image. The values for the intensity depend on the color space chosen for the image. Colors refer to the part of the electromagnetic spectrum that is visible to humans and color space is a notation in which the colors are specified (Tkalcic & Tasic 2003). Many different color spaces are used to represent color used for a wide range of applications (e.g., color printing, monitors and digital cameras).

Image processing results can vary greatly depending on the color space chosen for analysis. In the scope of this research, various color spaces have been used to achieve the best results for object detection via image thresholding. In the coming section, a short explanation will be provided for each color representation that is used in this thesis.

Red, Green and Blue (RGB)

The RGB color space is based on the way humans perceive color. The human retina forms all colors based on a combination of the three primary colors, red, green and blue. The RGB color space stores the red, green and blue image information into separate channels creating a three-dimensional array, with the values for each channel ranging from 0 to 255. When combining the information of each channel the color can be recreated for each pixel based on

the red, green and blue value of that pixel (Liu & Mason 2016). A drawback of the RGB color space is the high correlation between the individual channels as well as its non-uniformity which refers to the low correlation between the difference of two colors and the corresponding distance in the color space (Tkalcic & Tasic 2003).

In image processing, an RGB image can be converted to a single one-dimensional grayscale image representing the intensity of every pixel (Brahmbhatt 2013, pp.26). Because of the compression into a single channel, loss of information is inevitable. However, the goal of the transformation to grayscale images is to keep as much of the information from the original RGB image (Čadík 2008). For this research grayscale images were produced using the following color transformation (OpenCV 2019)

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{2.3}$$

Where R, G and B represent the channels of the RGB color space, each multiplied by a constant to determine the gray value Y of the image.

Hue, Saturation and Intensity (HSI)

The HSI color space takes a more intuitive approach, as it separates a pixel's color information from a pixel's brightness information. The image is represented as a three-dimensional array, with the pixels color information stored in the first two channels of the array, which are named hue and saturation. Hue can be characterized as the pure color of a pixel, based on a circular color wheel representation. Saturation indicates the amount of white light that is mixed with the base color, in other words, the purity of the color (Solomon & Breckon 2011). Vibrant colors have a high saturation while pastel colors have a low saturation (Carper et al. 1990). The intensity value of the color space represents the brightness of a pixel. Using the HSI color space offers the advantage of separating the color information from the brightness of an image. This is especially useful in the case of image segmentation if a specific object needs to be segmented based solely on its color or overall brightness. There are several variations of the HSI color space, with great variety in the way the values for hue, saturation, and intensity are calculated. In the scope of this thesis, the HSV and HSL transformations are used.

HSV (Hue, Saturation, Value)

For the HSV (Hue, Saturation, Value) transformation the intensity value V of a pixel is calculated as follows based on the RGB values of the pixel (OpenCV 2019):

$$V = \max(R, G, B). \tag{2.4}$$

Subsequently, the saturation S of the pixel is calculated in the following way

$$S = \begin{cases} \dfrac{V - \min(R,G,B)}{V} & if\ V \neq 0 \\ 0 & if\ V = 0 \end{cases} \tag{2.5}$$

and the hue H is calculated as follows

$$H = \begin{cases} \frac{60(G-B)}{V-\min(R,G,B)} & if\ V\ =\ R \\ \frac{120+60(B-R)}{V-\min(R,G,B)} & if\ V\ =\ G \\ \frac{240+60(R-G)}{V-\min(R,G,B)} & if\ V\ =\ B \end{cases} \tag{2.6}$$

with $H = H + 360$ if $H < 0$.

HSL (Hue, Saturation, Lightness)

In the case of the HSL (Hue, Saturation, Lightness) transformation, intensity value L (lightness) is calculated in the following way (OpenCV 2019)

$$\begin{aligned} V_{max} &= \max(R,G,B) \\ V_{min} &= \min(R,G,B) \\ L &= \frac{V_{max}+V_{min}}{2}. \end{aligned} \tag{2.7}$$

The formula for saturation value S is as follows

$$S = \begin{cases} \frac{V_{max}-V_{min}}{V_{max}+V_{min}} & if\ L\ <\ 0.5 \\ \frac{V_{max}-V_{min}}{2-(V_{max}+V_{min})} & if\ L\ \geq\ 0.5 \end{cases} \tag{2.8}$$

The calculation for the hue H for the HSL is comparable to the calculation observed for the HSV transformation, however in this case saturation S replaces the $V - \min(R,G,B)$.

The most important reason for using both the HSV and HSL transformation in this thesis is how the intensity is represented. As the intensity values for the HSL are based both on the highest and lowest intensity RGB channel, instead of just the highest intensity RGB channel, the corresponding lightness value is more nuanced, as it takes into account color differences. For example, assume two pixels: pixel 1 with the RGB values $(1, 1, 255)$ and pixel 2 with RGB values $(255, 255, 255)$. Both pixels would receive an intensity value of 255 in the HSV color space, but a different value in the HSL color space (128 for pixel 1 and 255 for pixel 2).

YUV

The YUV color space was originally used for analog color TV systems as it minimized the bandwidth by converting the original RGB values and was compatible with the older black and white television systems. The YUV consists of a luminance component (Y) and two color components (U and V) and offers the advantage of separating the color information from the

luminance (Al-Tairi et al. 2014). The YUV values are derived from the RGB values in the following way

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} +0.257 & +0.504 & +0.098 \\ -0.148 & -0.291 & +0.439 \\ +0.439 & -0.368 & -0.071 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} \tag{2.9}$$

In certain cases, the YUV color space gives better results for image segmentation than the HSI color space, as was demonstrated by Al-Tairi et al. (2014). In the scope of this thesis, the YUV color space was used in the scenarios where segmentation based the HSI color space did not offer satisfactory results.

2.4 Fourier Descriptors

As mentioned before, the result of image segmentation is a binary image containing object regions which can be classified based on their features. Which features to select for classification depends largely on the type of classification. For the digit classification in this thesis, the shape of the region's boundary is the feature used. Fourier descriptors are a way to represent a boundary's information. To explain the way these Fourier descriptors are derived, it is important to begin by introducing the Fourier series and Fourier transform.

Fourier series and transform

The Fourier series is based on the fact that all periodic functions can be decomposed into a sum of sines and/or cosines of different frequencies multiplied by a coefficient. This sum is called the Fourier series. The Fourier transform applies the same principle to a non-periodic function with a finite area under its curve. This function can be expressed as the integral of sines and/or cosines multiplied by a weighting function. One of the most important attributes of functions expressed in the Fourier transform or Series is that their original information can be entirely recovered using an inverse function (Gonzalez & Woods 2002, pp.222-223).

Fourier descriptors

The Fourier transform can also be used to extract boundary characteristics from an object region, called Fourier descriptors. The following section provides a brief explanation of how the Fourier descriptors are calculated.

The boundary of a region can be represented as an array of point coordinates $b(k)$

$$b(k) = [x(k), y(k)] \quad for \quad k = 0, 1, 2, 3..., K \ - \ 1. \tag{2.10}$$

where K represents the length of the boundary array, k indicates the coordinate point in the array and $x(k)$ represents the horizontal location and $y(k)$ the vertical position of the boundary pixel on the image. The coordinates are ordered in such a way that they follow the image region clockwise. Each point coordinate $x(k), y(k)$ in the array $b(k)$ can be represented as the complex number $s(k)$

$$s(k) = x(k) + jy(k). \tag{2.11}$$

Complex Numbers

A complex number is a number that is constituted of a real part consisting of a real number and an imaginary part consisting of a real number and the imaginary number j, which is equal to $\sqrt{-1}$. Complex numbers can be viewed as points in a plane called the complex plane, where the real part of the complex number can be plotted on the horizontal axis and the imaginary part on the vertical axis. For the complex number $s(k)$, $x(k)$ is the real part and $jy(k)$ the imaginary part, which is represented in coordinate system of the complex plane by the coordinates $(x(k), jy(k))$.

Calculating the Fourier descriptors

The advantage that the transition to complex numbers offers is the possibility to apply the Fourier transform on a one-dimensional problem instead of a two-dimensional problem without any loss of boundary information. The Fourier transform is applied on the boundary function $s(k)$ in the following way

$$a(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K} \quad for \quad u = 0, 1, 2, 3..., K\ -\ 1 \tag{2.12}$$

for $u = 1, 2, 3...K - 1$. The resulting array $a(u)$ contains the Fourier descriptors for the boundary function $s(k)$. Furthermore using the inverse of the Fourier transform as follows

$$s(k) = \sum_{u\,=\,0}^{K-1} a(u) e^{j2\pi uk/K} \quad for \quad k = 0, 1, 2, 3..., K\ -\ 1 \tag{2.13}$$

the original boundary function $s(k)$ can be restored without any loss of information. A useful attribute of Fourier descriptors is that the low frequencies contain coarse boundary shape information while the high frequencies contain the fine detail shape information. In practice, this means that it is possible to reconstruct the general shape of a region's boundary by only using a subset of descriptors of the low frequencies (Gonzalez & Woods 1992, pp.658).

Scaling and Rotation

For the classification of digits of the tally counter, it is important for the descriptors to be sensitive to rotation and symmetry, to prevent misclassification between digits with similar shapes like six and nine or two and five. To prevent this, the original boundary function $s(k)$ is adjusted based on the centroid of the contour (x_c, y_c) in the following way

$$s(k) = (x(k) - x_c) + j(y(k) - y_c). \tag{2.14}$$

where the centroid (x_c, y_c) is calculated as follows (Zhang et al. 2001)

$$x_c = \frac{1}{K} \sum_{t\,=\,0}^{K-1} x(t) \qquad y_c = \frac{1}{K} \sum_{t\,=\,0}^{K-1} y(t) \tag{2.15}$$

By performing this alteration to the boundary, the coordinates are transformed based on their position relative to the center of the digit region, resulting in a set of Fourier descriptors sensitive to rotational and symmetrical differences (Lu et al. 1993).

As the size of the digits are not important during the classification, it is important that the Fourier descriptors only reflect the shape of the boundary and not the size. To achieve this, the first Fourier descriptor, which represents the direct current (DC) component is removed from the set of Fourier descriptors, as the information represented stored the DC component only describes the position of the region and not the shape of it. Next, for the Fourier descriptors to be insensitive to the size of the contour, all descriptors are normalized by dividing them by the lowest frequency descriptor left after the removal of the DC. (Zhang et al. 2001), in the following way

$$a(u) = \frac{|FD_2|}{|FD_1|}, \frac{|FD_3|}{|FD_1|} \ldots \frac{|FD_N|}{|FD_1|} \tag{2.16}$$

Where FD_1 represents the lowest frequency component after the DC is removed, and FD_N represents the remaining Fourier descriptors in a selection of N Fourier descriptors.

Example of shape retrieval via Fourier descriptors

To illustrate the shape retrieval using a subset of Fourier descriptors an example is given using the boundary of the digit in figure 2.2, which was taken from the tally counter of a water meter in the dataset.

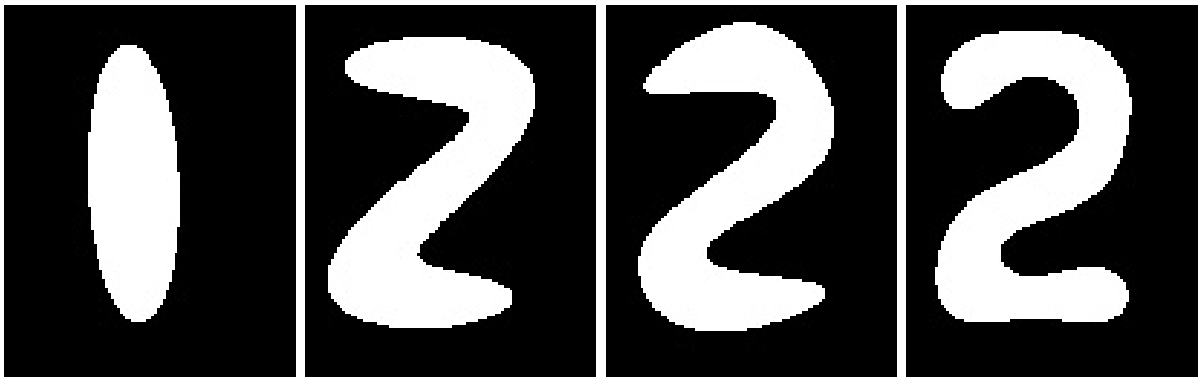
Figure 2.2: Image of digit to serve as example

The first step before calculating the Fourier descriptors is to extract the boundary information of the object. By performing a simple thresholding operation on the example image, an object region is segmented (figure 2.3a) from which the boundary pixels can be selected (figure 2.3b).

By calculating the Fourier descriptors for the region boundary it is possible to retrieve information of the boundary and recreate the region shape using the inverse Fourier transform. Figure 2.4 demonstrates the recreated shapes based on just a few Fourier descriptors. As shown in figure 2.4d, only 20 Fourier descriptors provide enough shape information for determining that the original object region belongs to the digit 2. The total number of descriptors for this boundary was 428, so in this case, only 4,7 percent of the descriptors were used to get a satisfactory result.

(a) *(b)*

Figure 2.3: (a) Original segmented digit region. (b) Boundary of the segmented digit region.



(a) *(b)* *(c)* *(d)*

Figure 2.4: Recreated digit regions using 2 (a), 6 (b), 10 (c) and 20 (d) Fourier descriptors

2.5 K-Nearest Neighbors Classification

To predict the digit class of a region based on its Fourier descriptors, a classification method is needed. In this thesis project, the k-Nearest Neighbor (k-NN) classification was used. This is a simple but effective classification method, first introduced at the beginning of the 1950's (Fix & Hodges Jr 1951). Popular because of its nonlinear decision boundary, easily tuned single integer parameter and low chance of overfitting, k-NN is deemed a reliable method for good classification performances in practical applications (Goldberger et al. 2005), (Denoeux 1995). The k-NN classification method assigns an unknown sample to a class based on the majority class in a set of k nearest neighbors in the feature space of a training set, as demonstrated in figure 2.5. The feature space refers to the m-dimensions in which the data points of a training set are based, where the number of features of the data point determines the number of dimensions m. For example, a training dataset where each data point consists of 10 Fourier descriptors has a 10-dimensional feature space. There are various ways in which the distance metric for the feature space can be calculated, however, most commonly the Euclidean distance is used (Müller et al. 2016, pp.46), (Chirici et al. 2016). It is possible to add a weighting factor to the neighbors in the k-NN classification process based on the distance. In such a case the

inverse of the distance to the unknown sample is used as a weight for each selected neighbor point.

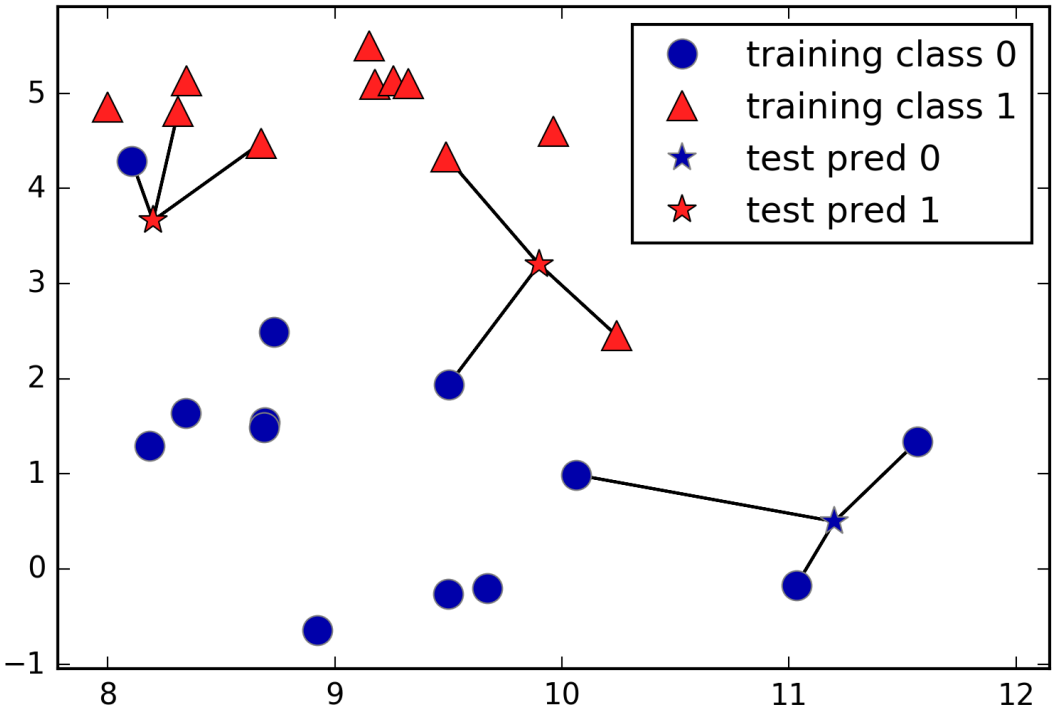


Figure 2.5: Examples of k-NN classifications using the three nearest neighbors(Müller et al. 2016, pp.38)

3. Methods

3.1 Data

Water Meter Images

SENARA provided a data set of 32 images of water meters as used for the registration of water use in the Small Irrigation and Drainage Areas Program. The images were acquired on the 12th of February of 2019 between 8 AM and 1 PM using a Huawei TAG-L23 smartphone. The image dimensions are 3104 by 3104 pixels. An algorithm was developed to read both the pointer value and the tally counter value of the water meter from those images.

Geographical Data

Furthermore, SENARA provided geo-information of an irrigation network in the Paraíso and Alvarado cantons in the province of Cartago, Costa Rica, that makes use of the water meters. The data consists of point data representing separate water meters, as well as the irrigation network to which these water meters are connected. The map in figure 3.1 shows the irrigation network as well as the locations of the water meters.

Each point has the following data:

1. Latitude and Longitude
2. Altitude
3. User of the farm
4. Sector in which the water meter is located

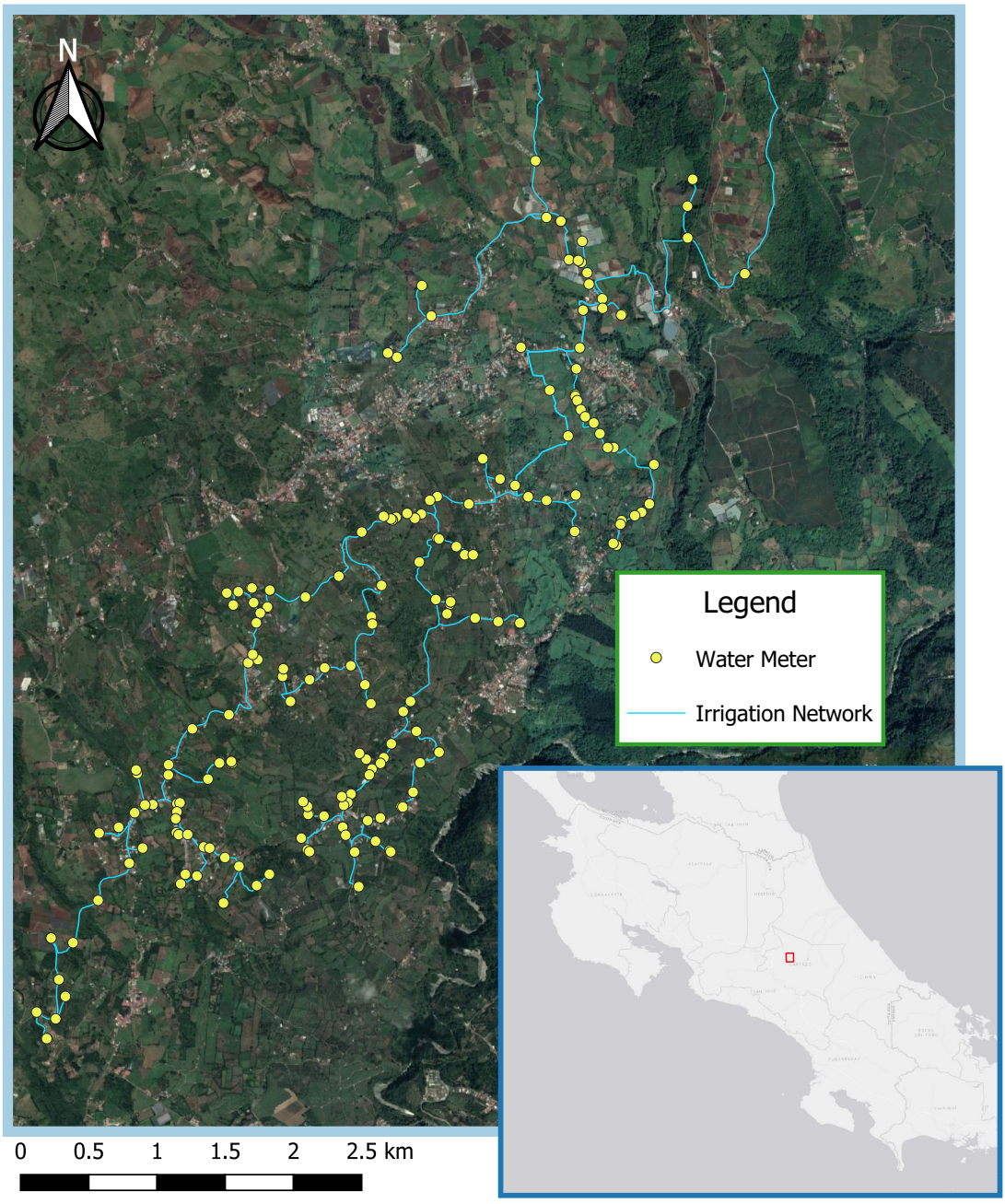


Figure 3.1: Locations of the water meters for the irrigation network found in the Paraíso and Alvarado cantons in the province of Cartago, Costa Rica.

3.2 Application Development

For the development of a system for the automated reading of water meters an architecture consisting of two main parts was created, the application and the backend. The following sections provide a brief explanation for both parts.

3.2.1 Application

An application was developed intended for smartphone devices with the and Android operating system, currently the most popular operating system worldwide (IDC 2019). Android has its own open-source integrated development environment (Android Studio) as well as many online resources to help with the application development (*Android Studio* 2019). Furthermore, Android Studio provides a local application testing environment which simulates a virtual android device with nearly the same capabilities of a physical device, called the Android Emulator (*Android Emulator* 2019).

The goal for the mobile application is to serve as an instrument for the members of the SUA to store water consumption data straight from the field. To facilitate operation and minimize user intervention, the water consumption values are determined via an image processing algorithm taking place on the backend. The main functionality the application provides is taking a picture and sending it to the server for image processing. Upon completion of the image processing, the application allows the user to check the result before it stores the information into the database. Furthermore, the application provides the user's device location for the backend to determine the location of the nearest water meter. This location is then displayed on a map, after which the user can select other water meters on the map in case the wrong water meter was selected based on the user's location. For the map functionality, the application makes use of the Google Maps API (*Maps SDK for Android* 2019), which allows developers to integrate interactive maps into their android applications.

3.2.2 Backend

Database

The database contains the locations of the water meters and is used to store the consumption data of each user. Upon completion of the image processing, the water consumption data is stored in this database, where it is assigned to a user based on the location of the user's device. For this task, the database needs to handle geographic objects, such as the point coordinates of the water meter locations. To facilitate this, a PostgreSQL object-relational database system (PostgreSQL 2019) is used with the PostGIS extension to handle the geographical component of the data (PostGIS 2019).

Framework

The backend connects the database, application and image processing algorithm. For this project, the backend functionality was designed using a python based web development framework named Django. Django is modeled in such a way that data handling, request routing logic, and the user interface are all separated. As a result, a Django web-application is built up out of separate modules that can each be altered without affecting the other modules in the framework (Holovaty & Kaplan-Moss 2009, pp.1-4). Django offers a built-in functionality for

PostgreSQL databases as well as the PostGIS extension, the only prerequisite being that the python package psycopg2 is installed. This package contains a database adapter used to query the database. Additionally, the Django framework offers a simple and easy to use administration interface (Holovaty & Kaplan-Moss 2009, pp.68-69) and the possibility to create custom commands for the administrator (*Writing custom django-admin commands* 2019), allowing to quickly extract and analyze data stored in the database.

To facilitate the exchange of data between the backend and the mobile application a Representational State Transfer (REST) web service was implemented. REST uses standard practices of the world wide web to facilitate the exchange of data to clients via a standard set of HTTP commands (Rodriguez 2008). REST web services are integrated into the Django framework via a separate toolkit called the Django Rest Framework (DRF) (Elman & Lavin 2014).

3.3 Image Processing

Overview

To help to answer the second and third research questions, an image processing algorithm was implemented. Figure 3.2 shows a diagram representing the workings of the image processing algorithm. Each blue square represents a module in the algorithm, the yellow diamonds represent the moments at which the algorithm produces an intermediate result. If the algorithm fails to obtain a result it raises a critical error causing the algorithm to stop for the thirst three decision moments, as these indicate a failure to properly segment the dial of the water meter. At the final stage of the algorithm, the values of both the pointer and the tally counter are determined. Unlike the earlier modules, when the algorithm fails to determine a result it does not stop the algorithm. Instead, the result is assigned a null value. The idea behind this is that a failure to segment the pointer does not automatically mean the digit values from the tally counter will not be successfully segmented and classified and vice versa. Only in cases where the algorithm fails to determine both the pointer value and the tally counter value, an error is raised and the user will be asked to restart the image processing with a new image. When only one of the result values was determined, the user is expected to manually enter the correct input.

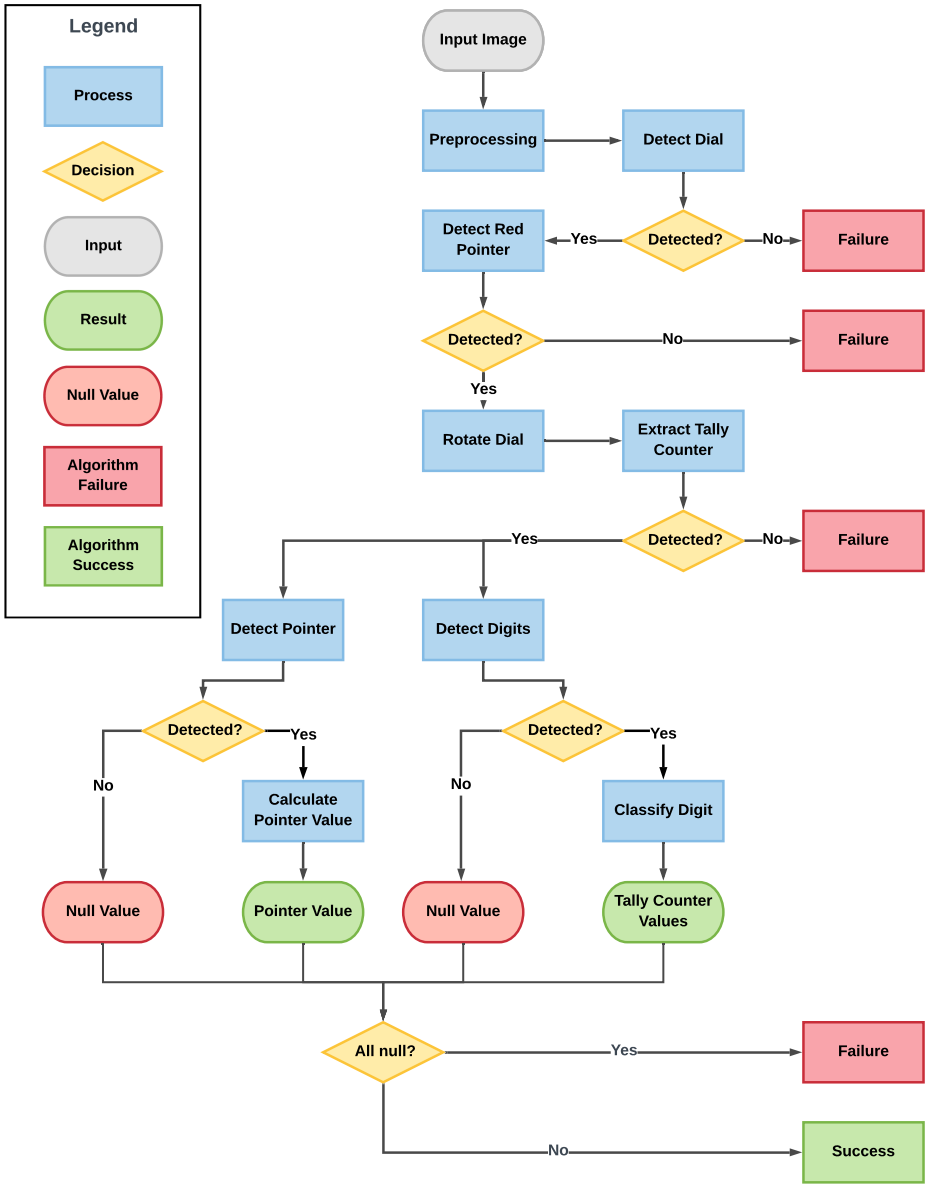


Figure 3.2: Overview of the image processing algorithm for reading the pointer and the tally counter

Preprocessing

One of the challenging parts of developing this application was handling the variation in environmental conditions. As the water meters are photographed under natural lighting conditions weather and shadowing have a strong effect on the quality of the images. In the test data set, differences in contrast as well as overexposed areas on the dial caused by the reflection of sunlight on the cover of the dial were noticeable.

Gamma Correction

To deal with the differences in contrast, an adaptive version of the gamma correction, proposed by Rahman et al. (2016), was applied. To explain how this method works a short explanation of the traditional gamma correction is needed, which is given by

$$I_{out} = c\, I_{in}^{\gamma} \tag{3.1}$$

where I_{out} and I_{in} are the input and output image intensities. Parameters c and γ control the shape of the transformation curve, c is intended to brighten or darken the intensity values while γ is the exponent of a power transform. One of the benefits of using adaptive gamma correction is the avoidance of the need to manually set the values for the c and γ parameters. Instead, the best-suited intensity transformation function for the input image is determined, depending on the contrast and brightness of the image. For a detailed explanation of the adaptive gamma correction algorithm see Rahman et al. (2016).

Contrast Limited Adaptive Histogram Equalization

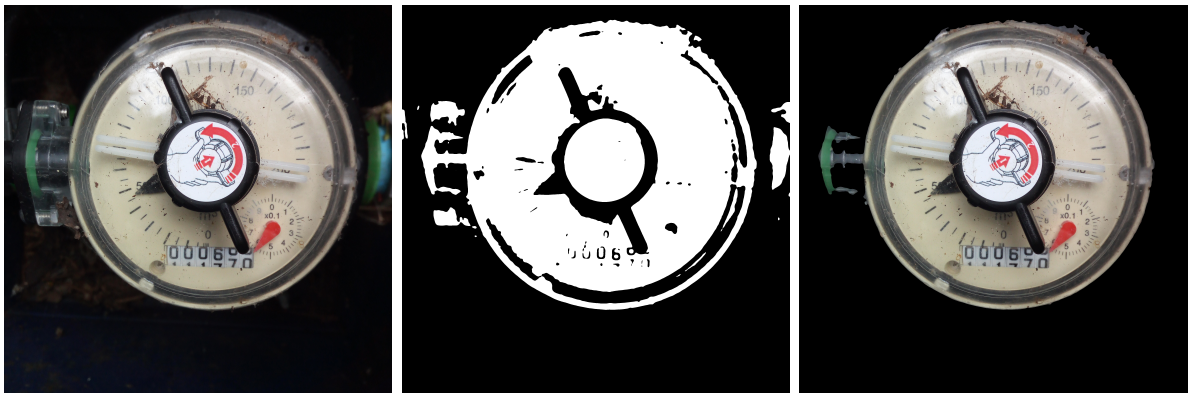
To correct for local inequalities in the imagery, Contrast Limited Adaptive Histogram Equalization (CLAHE) was applied (Zuiderveld 1994). CLAHE optimizes the image intensity values based on local properties. The original image gets split up into separate smaller regions, after which local histogram statistics are calculated for each of the smaller regions. To prevent noise the contrast enhancement for homogeneous areas is limited. This is done by setting a maximum number of pixels per histogram bin. The total number of pixels cut off over all the bins is then redistributed equally to keep the total histogram count equal. The maximum number of pixels allowed per bin is called the clip limit, which is defined as a multiple of the histogram averages.

3.3.1 Dial Extraction

Dial Detection

The first task of the algorithm is to locate and extract the surface of the dial. Attempts were made to detect the dials using the Hough Circle Transform (Ioannou et al. 1999). However, the results obtained were not satisfactory. Because of the variation of the angle at which the images of the test data set were taken, the outline of the dials was not always exactly a circle but more oval-shaped. Furthermore as the size of the dial in the image differs based on the distance at which the image was taken, the Hough Circle Transform needed to be run various times until the best match was found. This process is rather slow, which is not ideal for an application requiring instantaneous feedback.

Therefore, a different approach was chosen, namely object detection using image segmentation. One characteristic that all the images have in common is a very dark background. With this in mind, the first step for the extraction of the dial is removing the background by applying a mask to the lower lightness values of the image. In practice, this means that the pixel values of the input image are set to zero in case they lie below the mean lightness value. Subsequently, Otsu's Threshold is used to create a binary image of the grayscale version of the image, after which the largest region in the binary image was extracted. This contour encompasses the dial, as shown in the example in figure 3.3b.



(a) *(b)* *(c)*

Figure 3.3: The algorithm receives an input image (a) from which potential dial regions are segmented using Otsu thresholding (b). The largest region in this binary image is selected as the dial (c).

At this point the dial is almost completely separated from the background, however, the resulting image is not yet suitable for further analysis. For more efficient computation, it is required that all the irrelevant areas of the image are removed for the image to only contain the dial itself. In image processing, this is called cropping. After cropping is performed the result should produce an image that has a height and width equal to the diameter of the dial and the center of the dial exactly in the middle of the image.

The center of the dial is detected using a similar method used for extracting the dial itself. However, focus now shifts to extracting the black cap, see figure 1.1. In the binary images created using the Otsu threshold (figure 3.3b, the cap outlines are classified as background pixels because of their low-intensity values (caps are black). As the cap is surrounded by the dial's surface having high-intensity values, it is possible to extract the contour of the cap by inverting the binary values of the dial's region, and then select the largest region out of these inverted binary values. These steps are illustrated in figure 3.4a & 3.4b. To get a more accurate representation of the center of the cap, protrusions like the handlebars and the pointer are removed. This is done by calculating the average distance between each point in the boundary of the cap region and the center of mass of the cap region. Next, the average distance is used to draw a circle around the center of the cap region after which any parts of the cap region that fall outside of this circle are removed, as shown in figure 3.4c. The center of mass of this newly produced region representing the cap is then selected as the center of the dial, while the radius of the cap r_{cap} is calculated by taking the average distance between the boundary of the corrected cap region and its center of mass. Based on the a priori knowledge about the ratio between the radius of the cap and the radius of the whole dial, r_{dial} is calculated by:

$$r_{dial} = r_{cap} \cdot 2.55 \tag{3.2}$$

Finally, a circle with radius r_{dial} drawn around the center of the cap is selected as the dial's face, as shown in 3.4d. In case the algorithm fails to detect a dial, an error is raised and the algorithm stops.

(a) (b) (c) (d)

Figure 3.4: By inverting the binary values obtained via Otsu thresholding a binary image (a) is created in which the cap is the largest region (b). After correcting for protrusions like the pointer and the cap's handles (c) the center of mass of the cap region is taken as the center of the dial. The radius of the dial is calculated based on the radius of the cap, as the proportions between the dial's features are known. Next, the dial is cut out by extracting the pixels within a circle with the radius of the dial drawn around the center of the cap (d).

Correctional Rotation

For gauging the values of the water meter, the algorithm must receive input images with approximately similar locations of the dials' features. The tally counter, for example, needs to be situated at the central lower part of the input image for the algorithm to work properly. For this reason, it is necessary to check the dial's position in the input image and rotate it to be in the desired position. To perform this check, the position of the red pointer is checked relative to the center of the tally. The image is then rotated to position the red pointer at an angle of 43 degrees relative to the center of the cap and the center bottom of the image. In case the algorithm fails to detect the red pointer, the rotation can not be performed and the algorithm raises an error, stopping the image processing.

Tally Counter Detection

Detection of the area of the tally counter starts by selecting the image area where the tally counter is expected, namely the lower part of the dial. Thereafter object regions are segmented using thresholding on the U channel of the YUV color space. Via experimentation, it was established that this channel was the best suited for thresholding, as it offered the sharpest contrast between the color of the tally counter and the dial's face. After the segmentation of the object regions, unwanted pixels such as pixels belonging to the cap handles are removed from the regions by masking out pixels with low values in the Value channel. To check whether the detected regions match the shape and size of the tally counter the expected height and width of the tally counter are calculated, based on the size of the dial. Within the object regions, the region shape that best fits the expected shape is selected. When a positive match

for the tally counter is encountered, a rectangular area encompassing the region, as shown in figure 3.5c, is cut out from the image and divided into six equally sized parts, each containing one digit. At this point, the digit segmentation and recognition comes into play. In case a tally region is not detected, the algorithm raises an error stopping the image processing as it suggests an incorrect input image of the dial.

(a) *(b)* *(c)*

Figure 3.5: First, the area in which the tally counter is expected (a) is extracted. By thresholding within the U channel of the YUV color space and removing of pixels with a low Value in the HSV channel space a binary image is created containing object regions, one of which belongs to the tally counter (b). The pixels of the object region that best matches the shape and size of a tally counter (c) are extracted and used for further processing

Determining the Pointer and Tally Counter Values

After completing the previous steps, the algorithm determines the values of the tally counter and the black pointer. These values are retrieved independently from each other and failure in detecting one does not imply a general failure. Instead, partial failures return a null value, as explained at the beginning of this chapter. The following sections describe the methodology for determining both the pointer value and the value of the tally counter.

3.3.2 Pointer Reading

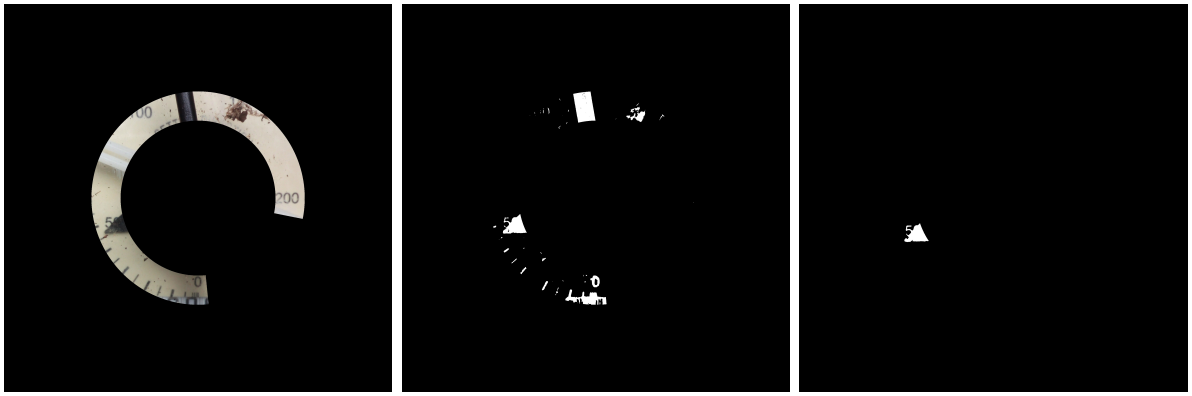
Pointer Detection using Image Segmentation

Based on the diameter and the center of the dial, the area in which the pointer is expected to be is cut out, as shown in figure 3.6a. In this section, this area will be referred to as the area of interest. A binary image, as shown in figure 3.6b, is created by thresholding the red (R) channel of the RGB color space. The red channel of the RGB color space was chosen for segmentation as experimentation showed that its color properties performed the best in separating the pointer region from other elements of the dial during experimentation.

The binary image obtained as a result of the thresholding contains object regions representing a potential pointer. Out of these, the three largest regions are chosen as potential pointer regions for further analysis. The first step is to check whether the potential pointer regions intersect with the inner border of the area of interest. In practice, this means that if a region does not contain pixels on the inner border of the area of interest it is excluded from the set of potential pointer regions. The second step is the removal of any region deemed either too small or too big. Based on the known proportions of the dial, the size limit for a pointer was set at 4 percent of the size of the area of interest. The third and final step is to check the region's spatial distribution over the area of interest. As the pointer is triangular and pointing away from the dial's center, most of the pointer's mass is located close to the dial's center.

With this in mind, all image regions that have a majority of their mass located towards the outer limit of the area of interest are excluded from the set of potential pointer regions.

In case no suitable pointer regions are found via the selection process, the process is repeated using the U channel of the YUV color space for image segmentation. As the YUV color space separates the brightness from the color information, it gives better results in cases where the RGB values are influenced by overexposure. However, the U channel is less successful in separating the pointer from unwanted elements of the dial, which is the reason the Red channel is used primarily.



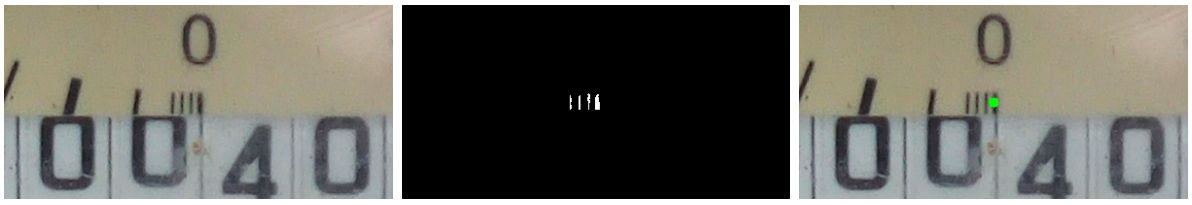
(a) *(b)* *(c)*

Figure 3.6: First the area in which the pointer is expected is cut out from the image (a), after which a binary containing object regions is produced via thresholding (b). The object region that fits the known properties of a pointer the best is selected as the region corresponding to the pointer (c).

Calculating the Pointer Value

The angle between the center of mass of the pointer region, the center of the dial and the zero point determines the value of the water meter pointer. When manually reading the pointer, the tip of the pointer is taken as the indicator to read the value. Unfortunately, for the pointer regions obtained via thresholding, the tip of the pointer is not always clearly distinguishable. To overcome this, a more robust approach is chosen by taking the centroid of the pointer region. As figure 3.8 shows, a line drawn from the center of the dial to the tip of the pointer goes through the center of the pointer's region. This means the same angle is obtained when using the center of mass instead of the tip of the pointer.

The marker indicating the zero point is positioned at the center of the tally counter, as seen in figure 1.1. To get the most precise coordinates for the zero point, the area above the middle of the tally counter was cut out. By performing thresholding on the Value channel of the HSV color space for this area, a binary image is created containing regions representing the markers found around the zero point, as shown in figure 3.7b. As the zero marker is the thickest of all the markers, the region with the largest area is selected and its center of mass is used as the coordinate of the zero point.



(a) (b) (c)

Figure 3.7: The marker representing the zero point of the dial is located above the center of the tally counter, along with three other markers (a). By thresholding the Value channel of the HSV color space, a binary image containing object regions belonging to the markers is created. The marker representing the zero point is the thickest and therefore the center of mass of the largest object region is selected as the coordinate for the zero point (c).

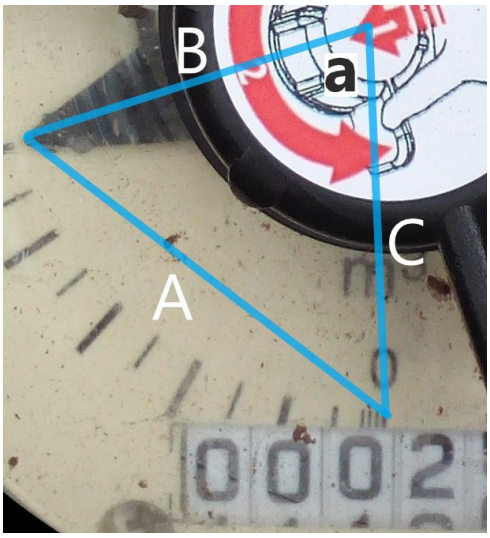


Figure 3.8: Triangle representation of the coordinates used for calculating the pointer value

Now that all the relevant image coordinates for the calculation of the pointer value are known, the angle between the zero value and the pointer is calculated. The law of cosines, describing the relation between the three sides of a triangle and one of the angles of the triangle, is used to calculate the angle between the zero point and the pointer. As demonstrated in figure 3.8 the three coordinates are interpreted as the points of a triangle with sides A, B, and C and an unknown angle a. The lengths of the sides are calculated by taking the Euclidean distance between the points of the triangle. Based on the lengths of the sides the unknown angle a is calculated as follows

$$a = \cos^{-1}\left(\frac{A^2 - B^2 - C^2}{-2BC}\right) \cdot \frac{180}{\pi} \tag{3.3}$$

The volume of water indicated by the pointer is calculated by dividing the value indicated at 180 degrees (130 m^3) by the number of degrees (180), resulting in a change of value of $0.72m^3$ per rotated degree. Pointer Value V is calculated accordingly

$$\begin{aligned} V &= a \cdot 0.72m^3 \quad if \quad x_{pointer} \leq x_{center} \\ V &= 130 + (180 - a) \cdot 0.72m^3 \quad if \quad x_{pointer} > x_{center} \end{aligned} \tag{3.4}$$

where a is the calculated angle in degrees, $x_{pointer}$ represents the horizontal position of the pointer's center of mass and x_{center} represents the horizontal position of the center of the dial. It is important to note that the angle at which the dial is photographed has to be perpendicular to the dial's face for this calculation to be correct.

Evaluation of the Pointer Value Readings

To measure the accuracy of the test results of the pointer value reading algorithm, the error is used between the value determined by the algorithm and the reference value determined by visually reading the pointer value. To evaluate the performance of the reading method the mean absolute error (MAE) is calculated, which is given by

$$MAE = \frac{\sum_{i=1}^{n} |e^i|}{n} \tag{3.5}$$

where n represents the number of measurements and $|e^i|$ is the absolute error between the reference value and the predicted value of the i^{th} measurement (Willmott & Matsuura 2005).

3.3.3 Tally Counter Reading

For the reading of the tally counter, a digit value must be determined for each of the six slots of the tally counter. The tally counter detected during the dial extraction is divided into six equal parts, serving as inputs for the digit recognition. Each input image is supposed to represent a single digit, however, in some cases, there are two equally sized partial digits present, as shown in figure 3.9. The digit recognition algorithm must be able to handle such cases as well.

Figure 3.9: Occurrence of two equally sized partial digits

Digit Segmentation

First, the input image is converted into a grayscale image and resized to a width of 250 pixels while maintaining the aspect ratio. Next, the image is preprocessed by smoothing the image to remove noise after which a contrast limited adaptive histogram equalization is performed

to get rid of local inequalities. After this, the minimum cross-entropy thresholding method proposed by Li & Tam (1998) is applied to create a binary image containing the potential digit regions. Regions that are considered too small or big are removed from the set of potential digits. Furthermore, regions that have a center of mass located outside of the central 80 percent of the image width are excluded. If there is more than one region left at this point, the possibility of two digit regions is addressed by checking for any overlap in the vertical position of the two largest regions. As the digits on the tally counter are always positioned on top of each other no vertical overlap should be possible. If there is no overlap, the assumption is made that there are two digit regions. In such cases, the two digit regions are returned to be classified separately. Cases where there is vertical overlap indicate that the digit region is split up, e.g. by dirt particles on the dial, causing two separate regions for the same digit. In such cases, the largest region is returned as the digit region for classification.

Digit Classification using Fourier Descriptors

The classification of the digits is a challenging task as the digits on the tally counter are often cut off or partially covered. This means that a digit can have multiple different types of appearances. To successfully classify the digits, the classification process should take into account that the digit classes 0 to 9 appear in many different ways, including partial display of the numbers. To classify the digits a classifying system was created that performs template matching based on digit templates.

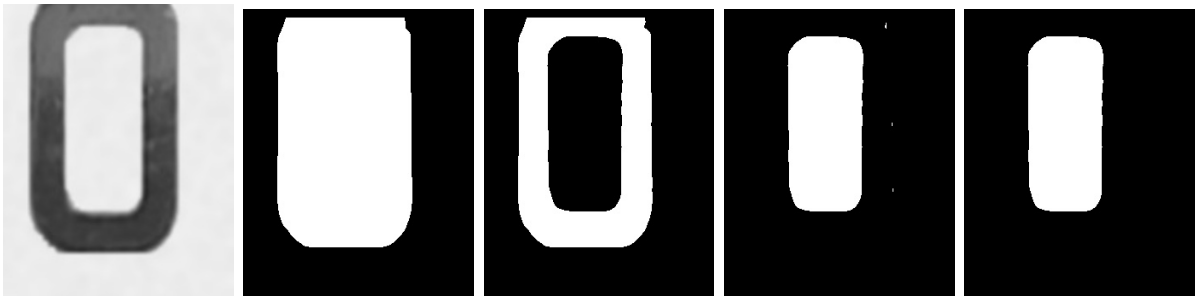
Digit Templates

The digit templates are created by placing a cutout of a digit on an artificially created background. To account for as much potential digit shapes the placement of the digit is randomized by picking a random starting coordinate. To simulate the occurrence of partial digits, as shown in figure 3.9, the starting coordinate can be outside of the actual size of the background, leading to part of the digit cutout being omitted from the digit template. A selection of examples illustrating the digit templates can be found in image 3.10. Using this method a set of 10.000 digit templates was created.

Figure 3.10: Examples of the randomly created templates

Template Features

The next step is to calculate the Fourier descriptors for digit regions segmented from each digit template as well as determining the number of holes this digit region contains. These are the features to be used for the classification of the unknown digit regions. The addition of the number of holes helps to distinguish between zeros and eights, as their overall shapes may otherwise be confused. To determine the number of holes within a digit region, the binary values within the digit region are extracted, as shown in figure 3.11c. Next, the binary values are inverted, meaning that background regions are now considered object regions and vice versa, resulting in a binary image with object regions representing the holes found within the digit region, as shown in figure 3.11d. Then, all the regions with an area larger than 5 percent of the original digit region's area are selected and counted as the number of holes in a digit region.



(a) *(b)* *(c)* *(d)* *(e)*

Figure 3.11: To count the number of holes within the digit template (a), the binary values falling within the digit region (b) are extracted (c) and then inverted (d). By selecting and counting the object regions with an area larger than 5 percent of the original digit region (e), the number of holes within the digit is determined.

After determining the features of the digit template, they are gathered and labeled with the corresponding digit class, forming the training data used for the training of a k-NN classifier. To determine the influence of the number of Fourier Descriptors on the classification, training data sets containing 10, 20, 30, 40 and 50 Fourier Descriptors were generated.

The python package Sci-kit learn, designed especially for machine learning purposes (Müller et al. 2016, pp.5-6), was used for training the k-NN classifier. Different parameters can be set, most importantly the number of neighbors, the distance norm and the weighting type (distance weighted classification or uniformly weighted classification). To determine the best performing parameters, various classifiers were trained with an odd number of neighbors ranging from 5 to 11 using both uniform and distance weighted classification. The Euclidean distance was used as the distance norm.

Reliability Score and Average Euclidean Distance (AED)

The k-NN classifiers are then used to classify the digit regions segmented from the six inputs taken from the tally counter. For each segmented digit region, a k-NN classification is performed based on the Fourier Descriptors of the region's boundary and the number of holes found inside the digit region. The result of the classification is the predicted digit class for the digit region based on the nearest neighbors in a feature space, where the features consist of the chosen number of Fourier descriptors and the number of holes. To assess the quality of

the prediction and signal any misclassifications, two evaluation metrics are derived based on the information the nearest neighbors to the classified data point provide. Firstly, a reliability score is calculated. If the neighbors are uniformly weighted, this is done by simply taking the fraction of the largest neighbor group to the total number of neighbors. For a distance weighted classification, each neighbor is provided with a weight given by the inverse distance. The combined weights of the largest neighbor class divided by the total weights for all the neighbors is then taken as the reliability score. Secondly, the average Euclidean distance (AED) in the feature space between the data point and the neighbors is calculated. If the features of the data point form a good match with the features of the digit templates used to train the k-NN classifier, the AED should be low (Wang et al. 2007). On the other hand, a high AED indicates that the features used to train the classifier don't match well with the data point that was classified.

Handling Multiple Digit Regions

In case the segmentation of an input image from the tally counter returns two digit regions, both are classified separately. The results are then compared to determine the best classification result. In case the result for the lower digit region is the expected result based on the result for the upper digit region, the result of the upper digit is returned as the final value. For example, if the upper digit region is classified as a seven and the lower digit region is classified as an eight, this suggests that the regions have been classified correctly and seven is returned as the final value. If this is not the case, it means one of the digit regions was not correctly classified. To determine which digit region is most likely correctly classified, their reliability scores and AED are compared, and the best classification is chosen as the final result. When the reliability score and AED contradict each other, for example, the upper digit has a high reliability score and a high AED and the lower digit has a lower reliability score and a lower AED, the result with the highest reliability score is picked as the final result. The reason for this is that a higher reliability score indicates that even though the features of the nearest neighbors and the unknown digit's region are considerably different, they are still the most similar in shape. A low AED with a low reliability score indicates that even though the shape of the unclassified digit region is similar to the features used to train the classifier, there is still uncertainty to which class it belongs exactly.

4. Results

4.1 Application Development

A prototype application was developed to serve as a tool for SUA members to store water consumption data of the users within their district into a central database. The application is intended to run on devices with Android 9.0, but should also work on devices running Android 8.0. The diagram shown in figure 4.1 depicts the functionality of the application from the perspective of the user.

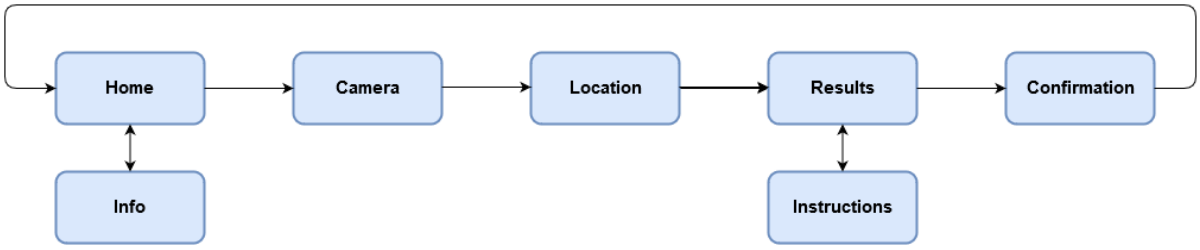
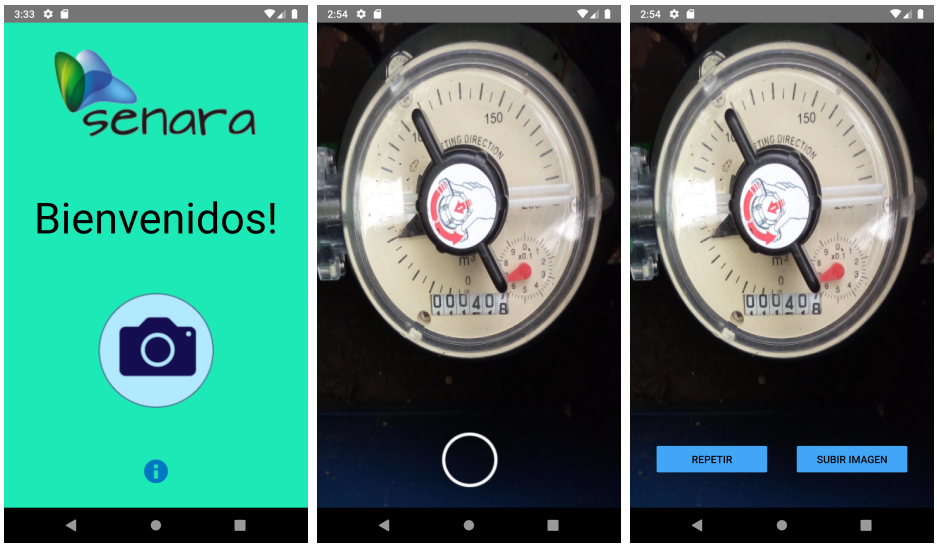


Figure 4.1: Functional diagram of the application

4.1.1 Application

The application opens with the home screen (figure 4.2a) containing a large camera button and an information button. By clicking on the camera button the image analysis process is started by opening the mobile device's camera preview and allowing the user to take a photo by pressing the capture button, depicted by a white circle. When a photo is taken it is saved to the local storage of the mobile device as a JPEG file under a unique name, after which the user is presented with the result. At this point two buttons pop up (figure 4.2c) allowing the user to decide to discard the result and take another photo, by clicking the button on the lower left side of the screen, or to upload the photo to the server for analysis, by clicking the button on the lower right side of the screen.

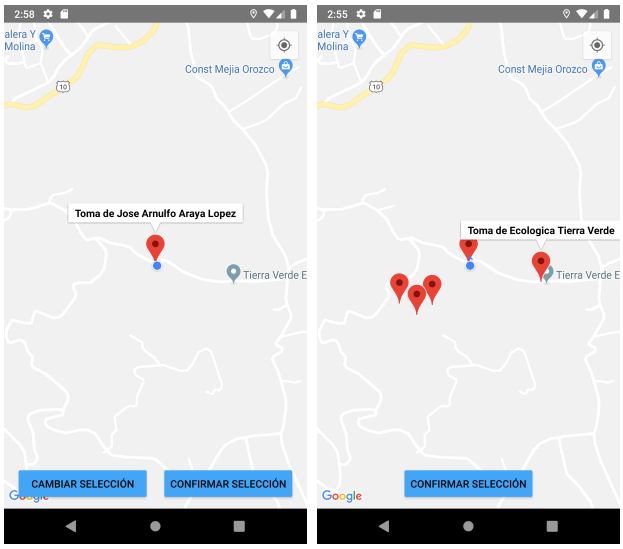
When the user decides to upload the photo, the application sends the JPEG file of the image to the server. When the upload is completed successfully the user is notified via a small pop up message. In case the uploading of the photo fails the user is asked to retry uploading it. In case of a successful upload, the application requests the location of the device, and sends out another request to the server containing the latitude and longitude of the device location as well as the name of the image that is to be analyzed. When the request is received at the server-side, the image processing is started. When completed successfully the server



(a) (b) (c)

Figure 4.2: Collection of screenshots of the application showing: the home screen of the application (a), the camera preview (b) and the options for the user after taking a photo (c).

returns the results of the analysis, in a new screen, containing a map displaying the location of the nearest water meter to the user's location, as shown in figure 4.3a. In case the image processing returns an error the user is asked to take a new photo and upload it again. In case no water meter could be found in the vicinity of the user, the user is notified there are no nearby water meters for which the analysis could be performed.



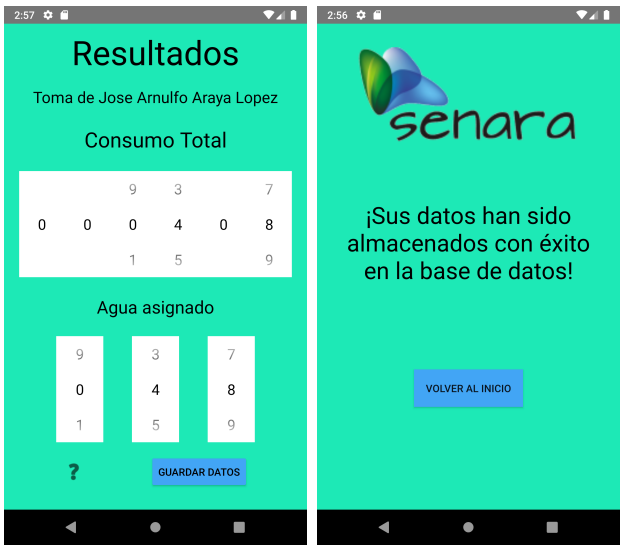
(a) (b)

Figure 4.3: Collection of screenshots of the application showing the map view for the location of the nearest water meters (a) and the location of the five nearest water meters within a 1000 m radius (b).

After receiving the location of the nearest water meter, the user can confirm if the right water meter was selected, by pressing the blue button at the lower right side of the screen. In case

the water meter selected is not correct, the user can indicate he wants to change the selected water meter by clicking on the blue button on the lower left part of the screen. Doing so, the five nearest water meter's within a 1000 m radius to the user's location are retrieved from the server and displayed, as shown in figure 4.3b. The user is then required to select the correct water meter.

After confirming the location, a screen is opened showing the results of the analysis of the photo of the water meter (figure 4.4a). In the upper part of the screen, the results of the reading of the tally counter are shown while the results of the pointer reading are shown at the bottom of the screen. At this point, the user can review the results of the analysis and correct the results via a number picker widget, if needed. For the values of the tally counter, the resulting digit values can only be changed in case the prediction for a digit has been labeled unreliable, based on it's AED and reliability score. An instruction screen can be accessed in case any confusion arises regarding the review of the results. After the results have been reviewed, the user confirms them by pressing the save button after which they are sent back to the backend and stored in the database. On completion of this step, the user is presented with a confirmation screen, allowing the user to return to the home screen (figure 4.4b).



(a) *(b)*

Figure 4.4: Collection of screenshots of the application showing the screen displaying the results of the analysis (a) and the screen confirming the data have been successfully stored in the database (b).

4.1.2 Backend

A backend architecture was developed to handle the communication between the database, application and image processing algorithm. In total there are four situations in which the user interacts with the backend of the application. Firstly, the image is uploaded to the server, as shown in figure 4.5. The JPEG file of the image is sent to the server-side via an HTTP request, where it is stored in a designated folder. The location of the image file is saved in the database. This way the image can be accessed quicker than if stored entirely into the database. After completion, a response is sent out indicating that the image was uploaded successfully.

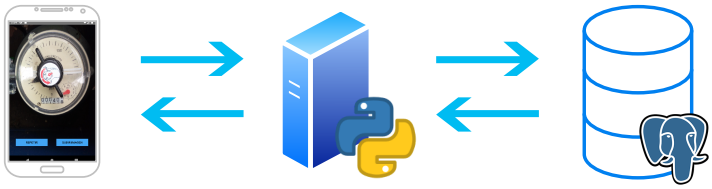


Figure 4.5: Diagram of image upload process.

The next step is image processing and determining the nearest water meter to the user's location. This process is illustrated in figure 4.6. Another HTTP request is sent out from the application to the server, containing the coordinates of the device location and the name of the image file that needs to be analyzed. The backend infrastructure queries the database for the location of the nearest water meter, loads the image specified from the image folder and starts the image processing algorithm. When the image processing is completed successfully the backend returns the nearest water meter information and the results of the image processing to the application. In case image processing is not successful or no water meter could be determined, an error response is sent to the application.

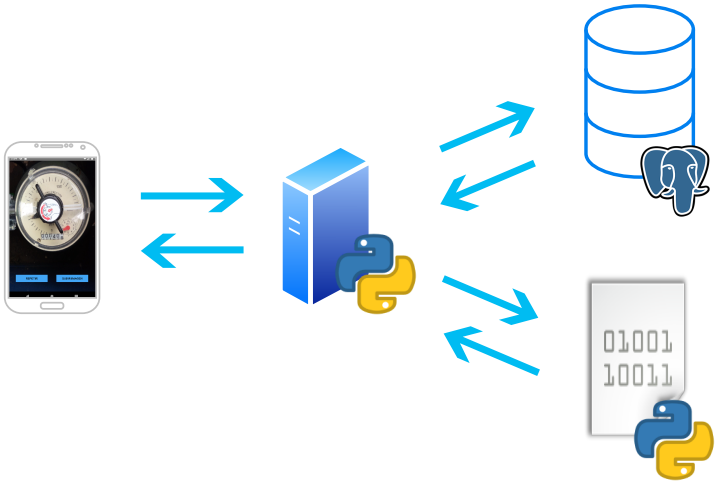


Figure 4.6: Diagram of image processing process.

In case the user wants to change the water meter selected, the location of the five nearest water meters in a 1000m radius is determined. After receiving an HTTP request with the user's location, the backend queries the database for the information of the nearest water

meters and returns these to the application. This process is illustrated in figure 4.7.

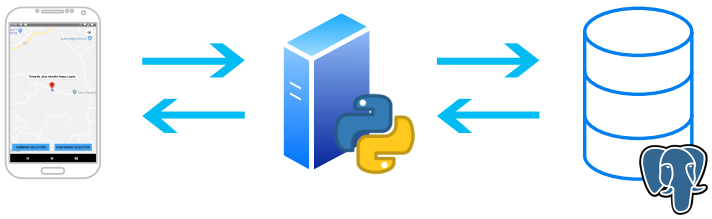


Figure 4.7: Diagram of the process of determining nearest water meter locations.

After the user reviews and adjusts the results of the image analysis, the data is sent back to the backend to be stored into the database, as illustrated in figure 4.8. Before storing the results, each result is checked to see if the user made changes to the initial results. If so, the value gets flagged as a corrected value in the database, making it possible to keep track of the changes made to the predictions and how well the image processing algorithm works. After completing storage of the data into the database, the image used for processing is removed from the image folder and a notification is sent out to the application.

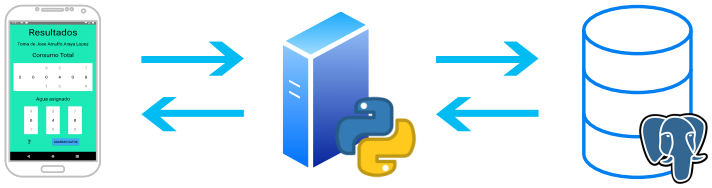


Figure 4.8: Diagram of data storing process.

4.2 Image Processing

4.2.1 Dial Extraction

The results for the segmentation of the dial, red pointer and tally counter can be found in table 4.1. For all results, the total number of inputs, the number of times the module completed without raising an error (Success), and the number of times the module raised an error, indicating that the dial could not be extracted from the input image (Failure) are given.

The segmentation of the dial failed for one out of the 32 input images, giving it a success rate of 97 percent. The segmentation of the red pointer proved to be more challenging than the initial dial segmentation, as the algorithm was unable to detect the position of the red pointer five out of the 31 times. Failure to detect the red pointers is often caused by overexposure of the part of the dial where the red pointer is found. An example of an overexposed red pointer is given in figure 4.9a. On one occasion, the detection of the red pointer fails because of shade being cast on the pointer in combination with overexposure of other parts of the dial, as is shown in figure 4.9b. Even though the pointers are still distinguishable for the human eye, it proved to be problematic for the algorithm to handle the cases where lighting conditions were sub-optimal.

Table 4.1: Results for the image segmentation steps used to extract the dial. Per step the number of inputs, the number of times segmentation was successfully completed (Success), the number of times the input caused an error (Failure) and the Success rate are given.

Step in Algorithm	Input	Success	Failure	Success Rate
Dial	32	31	1	97 %
Rotation (Red Pointer)	31	26	5	84 %
Tally Counter	26	25	1	96 %

(a) *(b)*

Figure 4.9: Examples of how overexposure (a) and overshadowing (b) caused a failure to detect the red pointer.

The tally counter segmentation receives 26 images to process, for which the tally counter was detected in 25 cases, resulting in a success rate of 96 percent. The one failure to detect the tally counter is caused by the white pointer overlapping with the tally counter. Fig 4.10 shows details of incorrectly identified tally counters. Overall, the application was able to interpret 25 of a total of 32 inputs, equaling a success rate of 78 percent.

(a) *(b)*

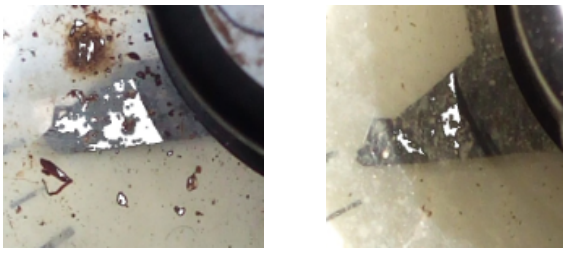
Figure 4.10: Image of the tally counter that was not detected (a) and the corresponding binary image containing the object regions (b).

4.2.2 Pointer Reading

Pointer Segmentation

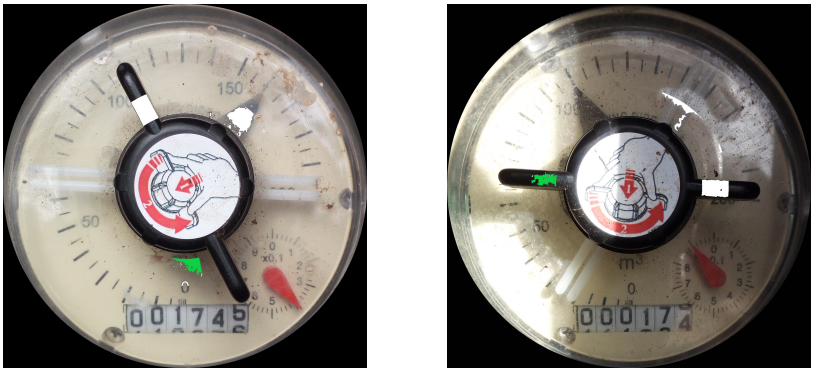
The segmentation of the pointer areas appeared to be handled well by the algorithm, only failing to detect a pointer region twice for the 25 inputs. Figure 4.11 shows details of the two misclassified pointers, which were affected by dirt on the dial's surface and unfavorable light conditions. The algorithm also produced two false positives, where regions were incorrectly classified as pointers, shown in figure 4.12. When taking into account the misclassified pointers, the algorithm correctly detected the pointer for 21 out of the 25 inputs, giving the pointer

detection a success rate 84 percent.



(a) (b)

Figure 4.11: Examples of missed pointers because of the presence of dirt on the dial (a) and unfavourable lighting conditions (b). The white pixels represent pixels belonging to the object regions obtained via thresholding.



(a) (b)

Figure 4.12: False positives caused by the presence of dirt on the dial (a) and lighting conditions (b). To illustrate why the misclassifications happened the object regions are shown in the images, the green region represents the expected pointer region as determined by the algorithm, while the white regions represent the regions that were deemed unfit by the algorithm.

Pointer Value Calculation

The results for the reading of the segmented pointer regions are presented in table 4.2. The reference value represents the value that was visually read from the image while the predicted value represents the pointer value that was predicted by the image processing algorithm and the error is the difference in between the true value and the predicted value. After removing the values for the wrongly segmented pointers from the results the mean absolute error for reading was $2m^3$, which is 1 percent relative to the scale of the dial (200 m^3). Additionally, the average error for the measurements was -1.5m^3 and the standard deviation was 2.8m^3.

Table 4.2: Results for the pointer value reading.

Predicted Value (in m^3)	Reference Value (in m^3)	Error (in m^3)
46	50	-4
46	49	-3
50	51	-1
100	109	-9
48	50	-2
54	54	0
0	0	0
0	0	0
48	45	3
49	51	-2
49	52	-3
0	0	0
141	141	0
80	85	-5
50	50	0
40	40	0
51	57	-6
50	50	0
104	105	-1
50	47	3
50	49	-1

4.2.3 Tally Counter Reading

Digit Segmentation

The digit classification was tested using 192 input images extracted from 32 tally counters. Each input image represented one of the six digits of a tally counter. For all images, a digit region was segmented via segmentation, after which a k-NN classification was performed.

Digit Classification

Accuracy

The accuracy of the uniform and distance weighted digit classification are shown in figures 4.13 and 4.14. The best classification accuracy was 76,6 percent, which was achieved various times for combinations of uniformly weighted and distance weighted k-NN classification and both a small and a large number of neighbors. The variation in classification results is small, the difference between the best and worst classification accuracy only being 3.2 percent.

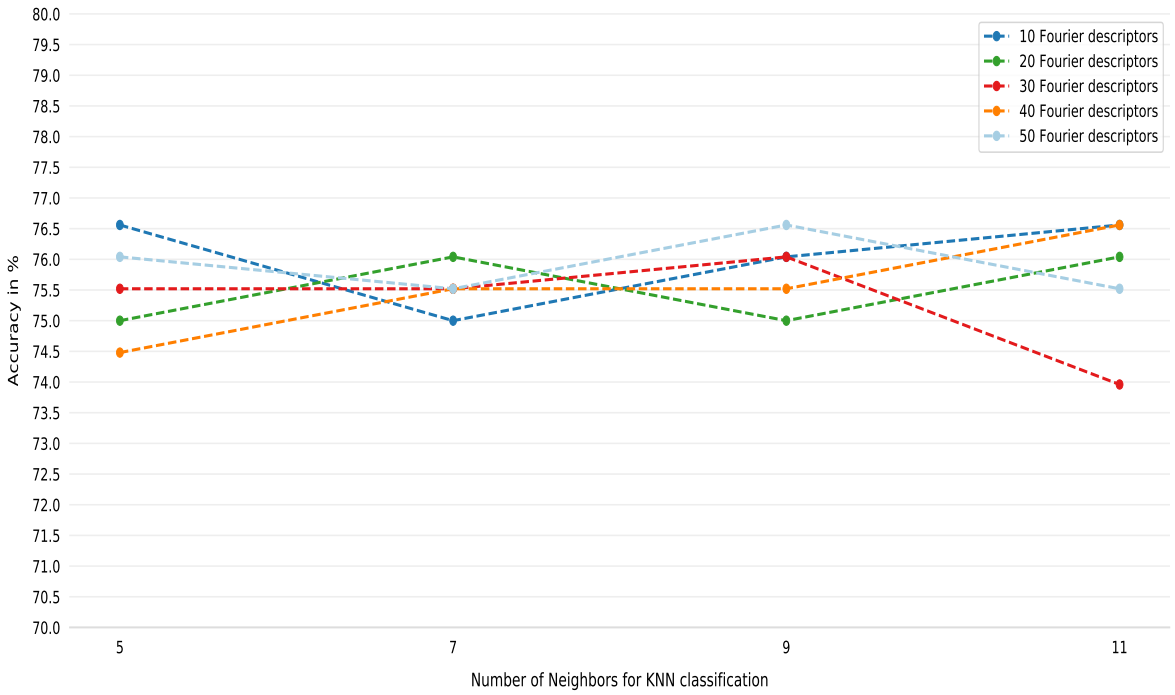


Figure 4.13: Classification Accuracy per number of Neighbors for each set of Fourier descriptors using a uniformly weighted k-NN classification.

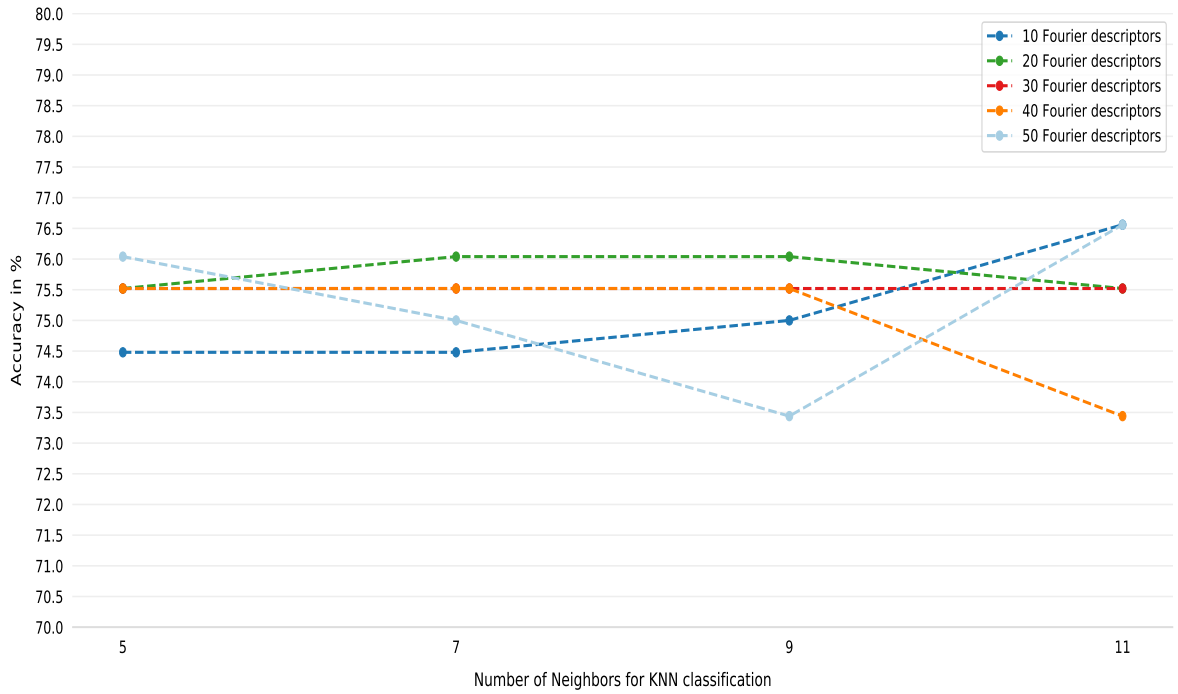


Figure 4.14: Classification Accuracy per number of Neighbors for each set of Fourier descriptors using a distance weighted k-NN classification.

Reliability Score

To evaluate its effectiveness for signaling misclassifications the reliability score was compared for the correct classifications and the incorrect classifications. The average reliability scores for correctly classified digit regions were above .95 for all classification runs. Furthermore, the median, first quartile and the third quartile were situated at the highest possible value for the reliability score (1) for all classification runs. This indicates that at least 75 percent of the correctly classified digit regions received a reliability score of 1. The reliability scores for the incorrectly classified digit regions showed more variation, as shown in figures figs. 4.15 and 4.16. For every classification run the median, first and third quartile are plotted. The position of the third quartile lies at the maximum reliability score for all incorrect classifications, indicating that at least 25 percent of the incorrect classified digit regions received the highest reliability score. The position of the first quartile varies between scores of 0.55 and 0.8, while the position of the median varies between scores of 0.81 and 1.

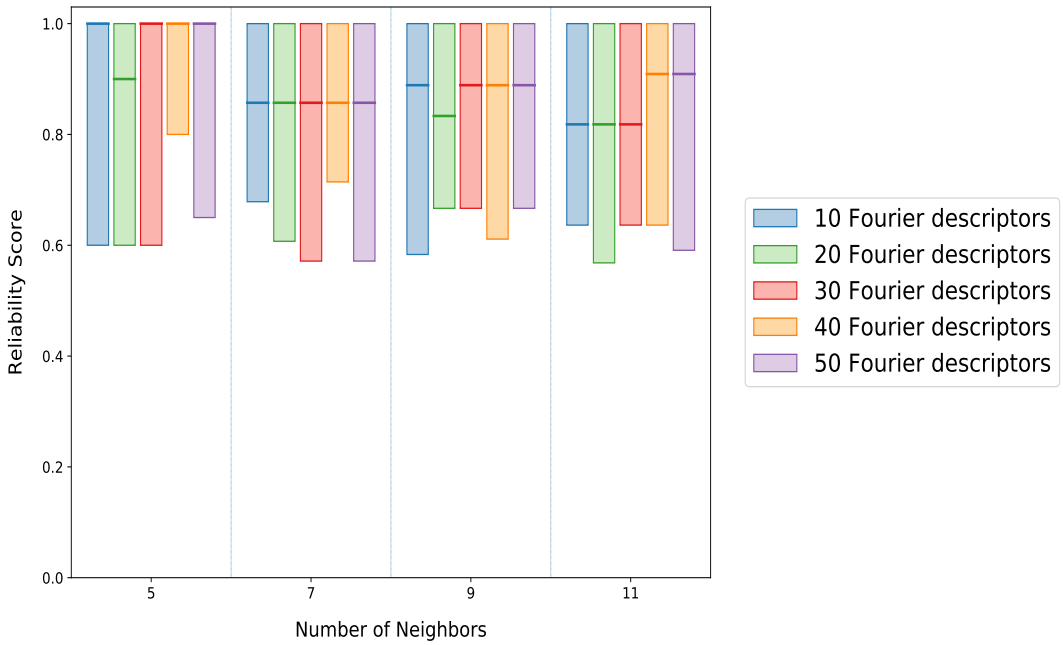


Figure 4.15: The median, lower quartile and upper quartile for the reliability score of the correct and incorrect classifications per set of Fourier descriptors using a uniformly weighted k-NN classification.

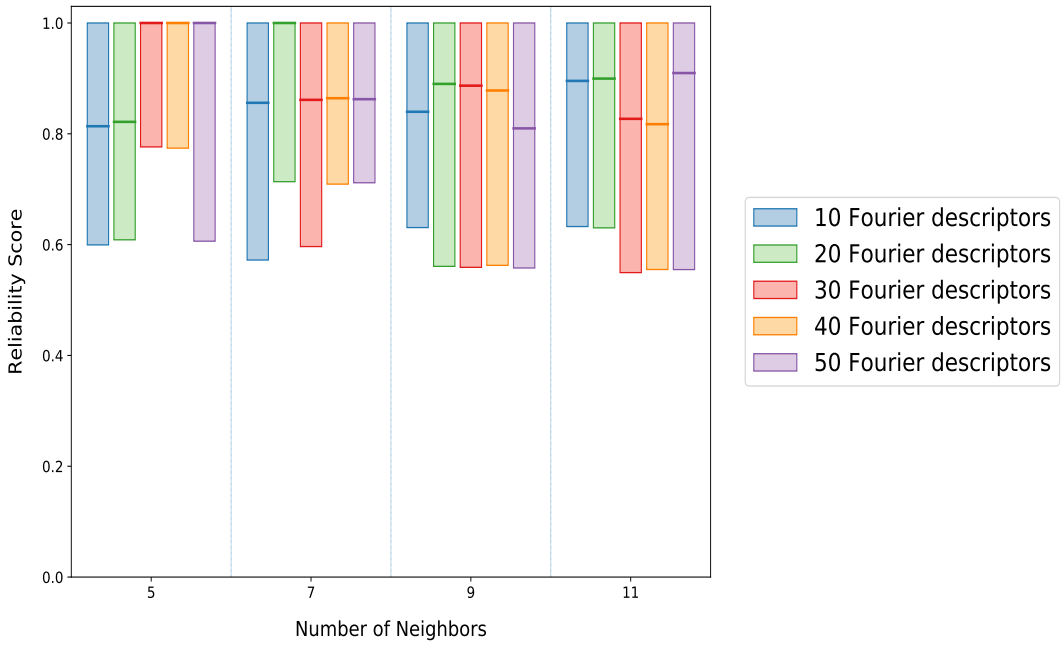


Figure 4.16: The median, lower quartile and upper quartile for the reliability score of the correct and incorrect classifications per set of Fourier descriptors using a distance weighted k-NN classification.

Average Euclidean Distance

To evaluate its effectiveness for signaling misclassification the AED was compared for correct classifications and incorrect classifications. The median, lower quartile and upper quartile for each classification run are shown in figs. 4.17 and 4.18. As expected the AED for correctly classified digits is considerably lower than for the incorrectly classified digits. While the median for the correctly classified digits varies between an AED of 0.25 and 0.28, the median for incorrectly classified digits varies between 0.5 and 0.58. Furthermore, the spread between the first and third quartiles of the AED is narrower for the correct classification, with lower and upper quartiles ranging between 0.18 and 0.21 and 0.4 and 0.44. For the incorrect classifications, on the other hand, the values for the lower and upper quartiles varied between 0.33 and 0.45 and 0.77 and 0.93.

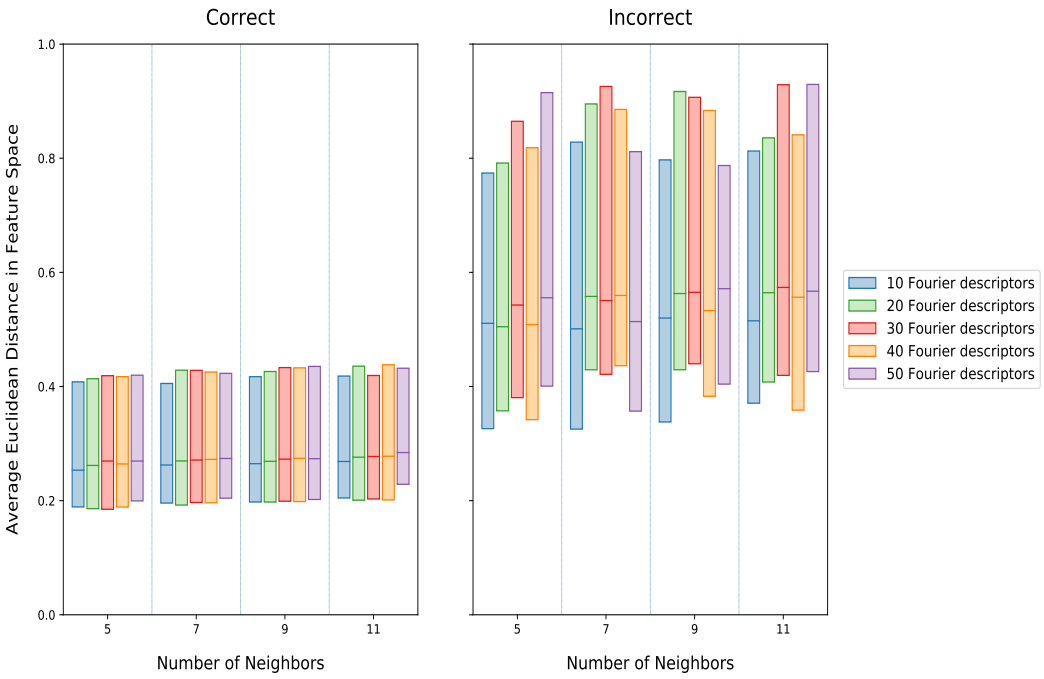


Figure 4.17: The median, lower quartile and upper quartile for the AED for the correct and incorrect classifications per set of Fourier descriptors using a uniformly weighted k-NN classification.

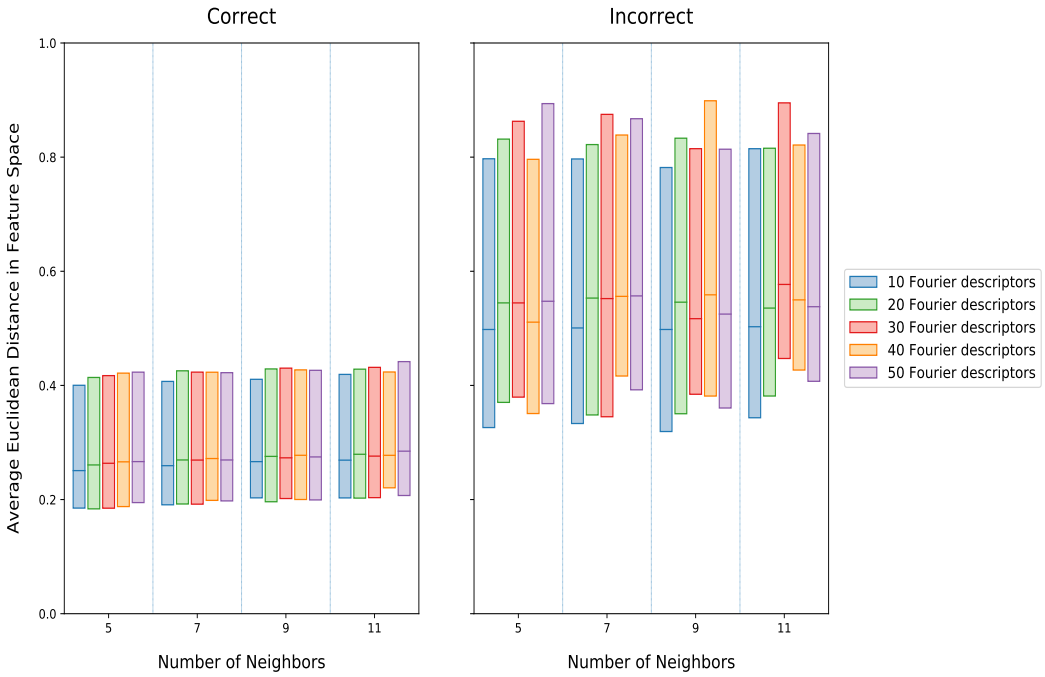


Figure 4.18: The median, lower quartile and upper quartile for the AED for the correct and incorrect classifications per set of Fourier descriptors using a distance weighted k-NN classification.

Correct Classifications

For the following sections, the digit classifications are observed in detail. The examples provided consist of the binary images of segmented digit regions, whose features were classified using a distance weighted k-NN classifier based on 50 Fourier descriptors and the eleven nearest neighbors in feature space.

Individual classification results show the classification method works well when the digit segmentation algorithm produced a well-segmented digit region, meaning that the region's boundary is smooth and unaffected by noise. Figures 4.19a to 4.19c show some results of nicely segmented digit regions, which were correctly classified with a high-reliability score and a low AED. The classification of partially covered digits also performed well, as long as the digit region was properly identified, as figs. 4.19d to 4.19f show. Because of dirt on the dial, poor camera focus or bad lighting, the digit segmentation produced digit regions that strongly differed from the digit shapes used for the training of the k-NN-classifier. However, in certain cases, the classifier was still able to correctly classify the digit region, as is shown in the examples in figs. 4.19g to 4.19i.

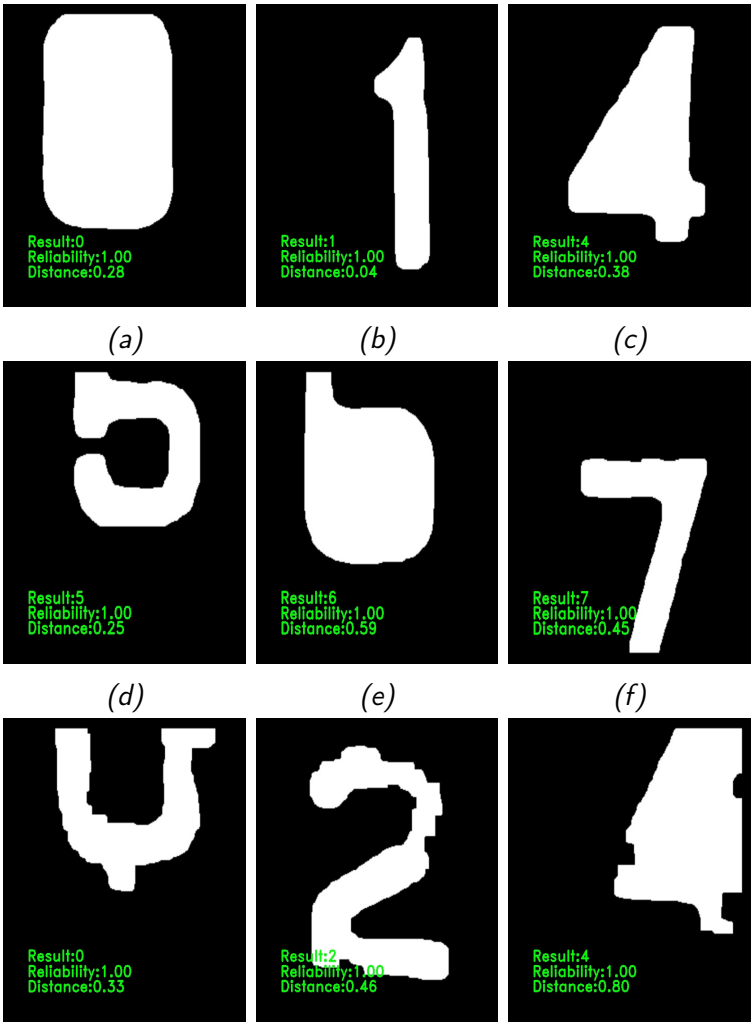


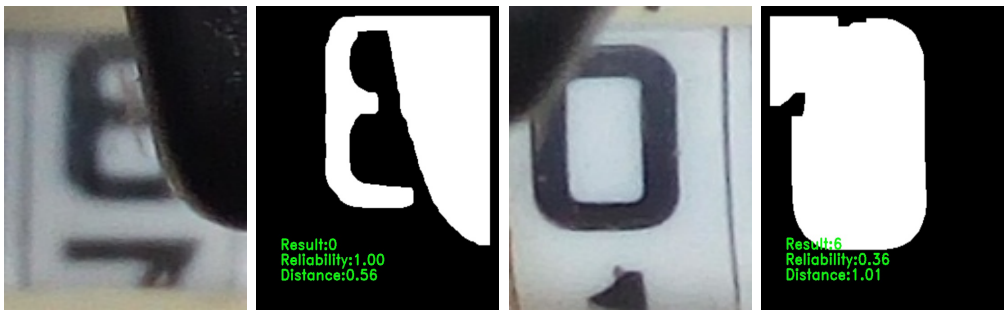
Figure 4.19: Examples of correctly classifications of well defined digit regions (a-c), well defined digit regions of partial digits (d-f) and of noisy digit regions (g-i). Each binary image shows the segmented digit region with the result of the classification, the reliability score and the AED written in green at the lower left part of the image.

Incorrect Classifications

The most obvious reason for misclassifications was a badly segmented digit region caused by dirt on the dial's surface, bad camera focus or overexposure and underexposure. figs. 4.20b and 4.20d present two examples of digit regions that were misclassified because their segmentation was affected by unfavorable conditions. Another cause for misclassifications were the handles of the cap, as they partly cover the tally counter sporadically. Figures 4.21b and 4.21d show two examples of the results of overlapping handles. Furthermore, there were cases of digit regions that seem to be segmented correctly but still misclassified, two examples of which are given in figs. 4.22a and 4.22b.

(a) *(b)* *(c)* *(d)*

Figure 4.20: Example how certain conditions like bad camera focus (a) and dirt on the dial's surface (c) cause thresholding to produce digit regions with unexpected shapes as shown in (b & d).



(a) *(b)* *(c)* *(d)*

Figure 4.21: Examples of how the overlapping handles, as shown in (a & c), cause thresholding to produce unexpected region shapes (b & d).

(a) *(b)*

Figure 4.22: Example of misclassifications of clear digit regions (a & b).

5. Discussion

5.1 Application Development

5.1.1 Application

The android application presented in this thesis performs well for the tasks it is intended to do: communicating with the central database and displaying the location of the nearest water meter(s) and algorithm results for reviewing, altering (depending on the reliability of the prediction) and confirmation. However, the application has currently only been tested using the virtual device emulator provided by Android Studio. Consequently, it is not known how the application will react if installed on a broader range of devices. Furthermore, the application has not been tested in the field, therefore it is uncertain whether the processes that function well using the emulator (i.e., taking a picture of a water meter, uploading it to the server, collecting the location from the mobile device and reviewing and saving the results of image processing) will function well in a real-world environment. Finally, it is important to note that the application has not been tested by the intended users: the members of the SUAs and SENARA. Their feedback and opinions are critical for a well functioning application and should be central to the further development of the application.

5.1.2 Backend

In addition to the testing of the application, the testing of the current backend system has been limited. So far the backend was only tested in combination with the simulated device in Android Studio, meaning it only handled requests from a single user at a time. As a result, it is uncertain how performance will be once multiple users contact the backend within a short time frame.

Using the information of the central spatial database, SENARA could perform analyses for the entire network of water meters, and keep track of water usage over time for both the entire system as well as for singular users. The administrator's functionality offered by the Django Framework (*Writing custom django-admin commands* 2019) makes it possible to write custom commands to query the database, as well as to perform analysis on the extracted data. By using this functionality, applications similar to the ones described by Soto-Garcia et al. (2013) could be realized, e.g., the possibility for farmers to review their water consumption via web and mobile applications, an accounting system for the water consumption per user or the creation of tools for central water monitoring by mapping an area's total water consumption per plot.

5.2 Image Processing

5.2.1 Dial Extraction

As shown in table 4.1, the extraction of the dial's face via image segmentation failed at least once at every step. A high variance of light on the dial appears to be the main cause for these failures, which affects the global thresholding used for image segmentation. Despite making use of different color spaces trying to reduce the influence of lighting conditions in cases where parts of the dial are overexposed segmentation fails to produce the desired results. This was mainly noticeable for the red pointer detection, as shown in the examples in figs. 4.9a and 4.9b. This was also the part of the dial extraction process that failed most often, accounting for five of the seven failures. Of the remaining two failures, the first occurred at the very beginning, when the dial's face is extracted from the background. This failure is caused because the dial is only partly captured on the input image, cutting off part of the black cap. Because of this, the algorithm failed to extract the cap properly and in turn, failed to extract the dial itself. The second failure occurred during the third step of the algorithm: the detection of the tally counter. The reason for this failure is the white pointer overlapping with the tally counter and being classified as a potential tally object because of its resemblance in color (both are white). The result is an exceptionally large and deformed object region, as can be seen in the binary image in figure 4.10b. As a result, the algorithm rejects it as a potential tally counter region because it does not conform with the shape and size expected for a tally counter.

5.2.2 Pointer Reading

The results described in chapter 4.2.2 show that the pointer detection via image segmentation performed reasonably well, with a success rate of 84 percent. For the 25 input images, the algorithm failed to detect two pointers, shown in figs. 4.11a and 4.11b. Additionally, two object regions were incorrectly identified as pointer regions, as shown in figs. 4.12a and 4.12b. As mentioned before, lighting conditions strongly influence segmentation via thresholding and were the reason for two of the four pointer detection failures, shown in figs. 4.11b and 4.12b. Figure 4.12b best shows the influence of the lighting conditions, as it shows the entire dial's face. The left part of the dial is overexposed while the right part is underexposed and shaded and as a result, the pointer is completely missed during segmentation. Instead part of the cap's handles are segmented. The difference in exposure has a major influence on the pixel values of the dial, causing segmentation with a global threshold to fail to segment the desired object regions. A solution to this problem would be to apply a moving window during the pointer detection, going over the area in which the pointer is expected to be, performing image segmentation each time the window moves. This approach has been successfully applied before by Jaffery & Dubey (2012) for the detection of pointer values based on canny edge detection and feature matching.

Noise caused by a dirty dial face was the other reason for failures during the segmentation. Figure 4.11a shows that the presence of dirt on the dial blocks part of the pointer, which is consequently not segmented. Figure 4.12a shows that a large part of the actual pointer region was included as a potential pointer region. However, this region is rejected because the amount of pixels close to the inner edge is too low, as this part of the pointer is largely covered in dirt. Certain operations can be applied to reduce the influence of noise caused by dirt particles, for example smoothing the image by applying a Gaussian filter (Liu & Mason

2016). This is only effective to a certain extent and does not help for cases where the dirt particles become bigger than just a few pixels. So, to make the image processing algorithm as effective as possible, the user should be instructed to clean the dial's surface before taking the picture.

With a mean absolute error of $2m^3$, derived from the readings shown in table 4.2, the current method of calculating the pointer value appears to be precise enough for the application it was designed for. When compared to the precision achieved in other studies, for example Ye et al. (2013) and Jaffery & Dubey (2012), the performance is quite poor. However, these studies were performed with a stationary camera, while the images of the dataset for this study were all taken with a handheld smartphone device. Therefore, there was variation in the angle at which the dial was photographed. As a result, pointer reading was less accurate as the angle of the camera influences the angle between the zero point and the pointer region. For accurate readings, the user should take the photograph perpendicular to the dial's face.

5.2.3 Tally Counter Reading

Overall classification accuracy achieved for the data set was much lower than the classifications results using Fourier descriptors in studies by Shridhar & Badreldin (1984) and Lu et al. (1993), where classification accuracies ranged between 98.6 and 99.8 percent. The main reason for this can be attributed to the highly controlled environmental circumstances, in the research performed by Shridhar & Badreldin (1984) and Lu et al. (1993). The images used in this thesis were obtained in the field and affected by many unfavorable factors. In the case of a practical application, these circumstances are inevitable to a certain degree, making high classification accuracies unfeasible. The influence of the environmental circumstances became clear when examining the digit regions used for classification. As discussed in chapter 4.2.3, the main cause for misclassifications were attributed to the inaccurate segmentation of digit regions.

One of the reasons for the failing segmentation to fail simply lies in the design of the dial, as the handles of the cap occasionally overlap with the tally counter. Because the color of the cap and the digits on the tally counter are similar (both black), the handle could not be separated from the underlying digit properly. As a result, oddly shaped digit regions were segmented. To prevent this type of misclassification, the best solution would be to train the classifier to recognize when the handle is blocking the tally counter. This could be achieved by creating templates of digits blocked by the handle.

Additionally, several environmental factors provided difficulty for proper digit segmentation. For example, bad camera focus resulting in the image to be blurred, caused digit segmentation to fail as the difference between the background and the image could not be determined properly. This is demonstrated in figures 4.20a and 4.20b. Another example comes once again from dirt being present on the dial, which when overlapping with a digit, causes unexpected digit region shapes, as seen in figures 4.20c and 4.20d. To be able to handle this type of deformation the digit template set should be expanded with templates simulating potential noise on the dial's surface.

Overall, the results for the weighted and the uniform classification were very similar; Both methods achieved an accuracy of 76,6 percent as their best result (figs. 4.13 and 4.14). Fur-

thermore, a larger number of Fourier descriptors used for classification does not necessarily lead to better classification accuracies. This corresponds with the notion that most of a region's shape information is captured in only a few descriptors (Gonzalez & Woods 1992, pp.658). This coincides with Lu et al. (1993), who used between 14 and 26 Fourier descriptors to classify handwritten digits. Since the variation of potential digit shapes is rather low for the dataset used for this research (digits on the tally counter are all of the same font), it was assumed that 10 Fourier descriptors would be sufficient for classification.

The AED shows promise as an indicator of misclassifications. Substantial differences between the Euclidean distances of correctly and incorrectly classified digits were observed as depicted in figs. 4.17 and 4.18. As would be expected, the median of the AED scores of the correctly classified digit regions was much lower than the AED of their incorrectly classified counterparts. A high AED indicates that all the nearest neighbors are located far away in feature space implying that the current digit template set does not contain enough information to correctly classify the region. The reliability score did not turn out to be useful as an indicator of misclassifications. Even though the correctly classified digit regions had a high average reliability score, the reliability scores of incorrectly classified digits were not much lower, as demonstrated in figs. 4.15 and 4.16.

6. Conclusion and Recommendations

6.1 Conclusion

The following section provides a concise answer to the research questions stated in section 1.3.

1. Which functionality is promising for a simple Android application that allows the automated reading of water meters?

The design and functionality of the application presented in this thesis are promising. Allowing the user to review the results of the analysis prevents erroneous predictions to be stored in the database. At the same time, the AED indicator shows promise as an indicator of misclassification, enabling a reviewing system in which only unreliable predictions can be changed by the user.

Considering the high variability of the circumstances in which the testing images where taken, the current performance of the image processing algorithm offers perspective for the future. From the results, it became quite clear that the high accuracy obtained with controlled environments, such as reported by Shridhar & Badreldin (1984), Lu et al. (1993) and Jaffery & Dubey (2012), is not attainable for a practical application. However, current results still leave room for improvement by better preparing for the conditions encountered, such as the presence of dirt on the dial and lighting circumstances.

2. What accuracy is achieved to read pointer values indicating water gifts?

The pointer value was read with a mean absolute error of $2m^3$ (1 percent relative to the scale). The average error was -1.5 m^3, indicating a minor bias. To obtain the most accurate readings the water meter should be photographed perpendicularly, furthermore the presence of varying lighting conditions, causing the dial of the water meter to be underexposed or overexposed should be avoided.

3. With what accuracy can the digits of the cumulative tally counters be read?

A classification accuracy of 76.6 percent was achieved for the reading of digit values of the cumulative tally counters using a k-NN classifier. This result is promising considering the main reason for misclassification was the incorrect segmentation of digit regions caused by the presence of dirt on the dial and the handles of the cap overlapping with the tally counter. Therefore, to improve the accuracy of readings it is recommended to clean the dial beforehand and position the handles of the cap so that they don't block the tally counter.

4. How can the application be used within the context of a spatial database infrastructure?

The application presented in this thesis is suitable as a tool to input data into the central spatial database from within the field. When combined with the administrative functionalities offered by the Django framework SENARA could create tools for the central monitoring of water use as well as perform accounting for water consumption of its users.

6.2 Recommendations

The current application is only a prototype. Even though the user experience was kept in mind, it was not one of the main concerns during development. In case the application presented in this thesis is put into use in a real working environment, it is recommendable to rework the application in collaboration with the intended end-users. Furthermore, it is important to note that the current backend architecture was only tested out handling requests from a single simulated device. A recommendation for SENARA would be to implement the system presented in this thesis within a small testing environment to evaluate its workings. Ideally, any potential shortcomings in the design would present itself during testing. Focus should lie on making sure the application is easy to use by the intended users and determining whether the current set up is suitable for the handling of a heavier workload, e.g. handling multiple requests within a short time frame.

The analytical potential offered by the Django framework in combination with the central spatial database has not been fully used during this thesis. Based on the foundations laid there are promising options to incorporate GIS technologies into the backend for the creation of monitoring tools for water consumption, both for farmers and SENARA. The tools and applications described by Soto-Garcia et al. (2013) can serve as a reference for this.

One of the major restrictions to properly assess the workings of the image processing algorithm is the relatively small dataset on which the functionality is based. Currently, it is uncertain how the algorithm would perform in a different setting or environmental conditions. To obtain a more complete evaluation of the image processing algorithm, it is advisable to test it out with a larger data set, containing images of water meters in different weather conditions and lighting circumstances. Furthermore, it is worth investigating different classification methods for digit recognition. The machine learning techniques introduced for digit classification by LeCun et al. (1990) could serve as a starting point to improve the digit classification, or perhaps machine learning could be applied for the entire process of reading the dial's water consumption values.

References

Acharya, S., Pandey, A. & Chaube, U. (2014), 'Use of geographic information systems in irrigation management: A review', *Journal of Indian Water Resources Society* **34**(2), 32–39.

Al-Tairi, Z. H., Rahmat, R. W. O., Saripan, M. I. & Sulaiman, P. S. (2014), 'Skin segmentation using yuv and rgb color spaces.', *JIPS* **10**(2), 283–299.

Alvarado, D. (2003), 'Ley de creación del servicio nacional de aguas subterráneas, riego y avenamiento (senara) n 6877', *Recuperado de: http://www. drh. go. cr/textos/documentos/100% 20ley% 20aguas* **20**.

Android Emulator (2019).
URL: *https://developer.android.com/studio/run/emulator*

Android Studio (2019).
URL: *https://developer.android.com/studio*

Babcock, M., Wong-Parodi, G., Small, M. J. & Grossmann, I. (2016), 'Stakeholder perceptions of water systems and hydro-climate information in guanacaste, costa rica', *Earth Perspectives* **3**(1), 3.

Birkel, C. & Demuth, S. (2006), 'Drought in costa rica: temporal and spatial behaviour, trends and the relationship to atmospheric circulation patterns', *IAHS PUBLICATION* **308**, 338.

Brahmbhatt, S. (2013), *Practical OpenCV*, Apress.

Čadík, M. (2008), Perceptual evaluation of color-to-grayscale image conversions, *in* 'Computer Graphics Forum', Vol. 27, Wiley Online Library, pp. 1745–1754.

Carper, W. J., Lillesand, T. M. & Kiefer, R. W. (1990), 'The use of intensity-hue-saturation transformations for merging spot panchromatic and multispectral image data', *Photogrammetric Engineering and remote sensing* **56**(4), 459–467.

Chirici, G., Mura, M., McInerney, D., Py, N., Tomppo, E. O., Waser, L. T., Travaglini, D. & McRoberts, R. E. (2016), 'A meta-analysis and review of the literature on the k-nearest neighbors technique for forestry applications that use remotely sensed data', *Remote Sensing of Environment* **176**, 282–294.

Córdoba, S. M. S., Pino Gómez, M. & Gaviria Montoya, L. (2016), 'Build up a database to determine the management of drinking water in the province of cartago, costa rica', *Journal of Water, Sanitation and Hygiene for Development* **6**(4), 584–592.

de Bruin, S., Pereira, J. F. A., Biosistemas, S. J., Mesén, R. A., Quirós, E. S. & Blancos, C. (2017), 'Opportunities and requirements for implementing an irrigation monitoring and management platform in costa rica'.

Denoeux, T. (1995), 'A k-nearest neighbor classification rule based on dempster-shafer theory', *IEEE transactions on systems, man, and cybernetics* **25**(5), 804–813.

Elman, J. & Lavin, M. (2014), *Lightweight Django: Using REST, WebSockets, and Backbone*, " O'Reilly Media, Inc.".

FAO (2015).

Fix, E. & Hodges Jr, J. L. (1951), Discriminatory analysis-nonparametric discrimination: consistency properties, Technical report, California Univ Berkeley.

Goldberger, J., Hinton, G. E., Roweis, S. T. & Salakhutdinov, R. R. (2005), 'Neighbourhood components analysis', pp. 513–520.

Gonzalez, R. C. & Woods, R. E. (1992), *Digital image processing 2/E*.

Gonzalez, R. C. & Woods, R. E. (2002), *Digital image processing 3/E*.

Holovaty, A. & Kaplan-Moss, J. (2009), *The definitive guide to Django: Web development done right*, Apress.

IDC (2019), 'Smartphone market share'.
URL: *https://www.idc.com/promo/smartphone-market-share/os*

Ioannou, D., Huda, W. & Laine, A. F. (1999), 'Circle recognition through a 2d hough transform and radius histogramming', *Image and vision computing* **17**(1), 15–26.

Jaffery, Z. A. & Dubey, A. K. (2012), 'Architecture of noninvasive real time visual monitoring system for dial type measuring instrument', *IEEE Sensors Journal* **13**(4), 1236–1244.

Jonoski, A., Almoradie, A., Khan, K., Popescu, I. & Van Andel, S. (2013), 'Google android mobile phone applications for water quality information management', *Journal of Hydroinformatics* **15**(4), 1137–1149.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E. & Jackel, L. D. (1990), Handwritten digit recognition with a back-propagation network, *in* 'Advances in neural information processing systems', pp. 396–404.

Li, C. H. & Lee, C. (1993), 'Minimum cross entropy thresholding', *Pattern recognition* **26**(4), 617–625.

Li, C. & Tam, P. K.-S. (1998), 'An iterative algorithm for minimum cross entropy thresholding', *Pattern recognition letters* **19**(8), 771–776.

Liu, J. G. & Mason, P. J. (2016), *Image processing and GIS for remote sensing: Techniques and applications*, John Wiley & Sons.

Lu, Y., Schlosser, S. & Janeczko, M. (1993), 'Fourier descriptors and handwritten digit recognition', *Machine Vision and Applications* **6**(1), 25–34.

Maps SDK for Android (2019).
URL: *https://developers.google.com/maps/documentation/android-sdk/intro*

Müller, A. C., Guido, S. et al. (2016), *Introduction to machine learning with Python: a guide for data scientists*, " O'Reilly Media, Inc.".

Neelin, J. D., Münnich, M., Su, H., Meyerson, J. E. & Holloway, C. E. (2006), 'Tropical drying trends in global warming models and observations', *Proceedings of the National Academy of Sciences* **103**(16), 6110–6115.

OECD (2012), 'Meeting the water reform challenge'.

OpenCV (2019), 'Miscellaneous image transformations'.
URL: *https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html*

Otsu, N. (1979), 'A threshold selection method from gray-level histograms', *IEEE transactions on systems, man, and cybernetics* **9**(1), 62–66.

PostGIS (2019), 'Postgis:spatial and geographic objects for postgresql'.
URL: *https://postgis.net/*

PostgreSQL (2019), 'Postgresql:the world's most advanced open source relational database'.
URL: *https://www.postgresql.org/*

Rahman, S., Rahman, M. M., Abdullah-Al-Wadud, M., Al-Quaderi, G. D. & Shoyaib, M. (2016), 'An adaptive gamma correction for image enhancement', *EURASIP Journal on Image and Video Processing* **2016**(1), 35.

Rey, D., Holman, I., Daccache, A., Morris, J., Weatherhead, E. & Knox, J. (2016), 'Modelling and mapping the economic value of supplemental irrigation in a humid climate', *Agricultural Water Management* **173**, 13–22.

Rodriguez, A. (2008), 'Restful web services: The basics', *IBM developerWorks* **33**, 18.

Saravanan, S., Jennifer, J. J., Abijith, D. & Singh, L. (2019), A gis-based spatially distributed crop water demand modelling for pullambadi canal command area in lower cauvery basin, tamil nadu, india, *in* 'Advances in Remote Sensing and Geo Informatics Applications', Springer, pp. 33–35.

Shridhar, M. & Badreldin, A. (1984), 'High accuracy character recognition algorithm using fourier and topological descriptors', *Pattern Recognition* **17**(5), 515–524.

Solano, P., Alvarado, L. F., Jimenez, E., Contreras, W. & Alfaro, M. (2012), 'Escenarios de cambio climático regionalizados para costa rica'.

Solomon, C. & Breckon, T. (2011), *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*, John Wiley & Sons.

Soto-Garcia, M., Del-Amor-Saavedra, P., Martin-Gorriz, B. & Martínez-Alvarez, V. (2013), 'The role of information and communication technologies in the modernisation of water user associations' management', *Computers and electronics in agriculture* **98**, 121–130.

Tkalcic, M. & Tasic, J. F. (2003), *Colour spaces: perceptual, historical and applicational background*, Vol. 1, IEEE.

Todorovic, M. & Steduto, P. (2003), 'A gis for irrigation management', *Physics and Chemistry of the Earth, Parts A/B/C* **28**(4-5), 163–174.

Valverde-Arias, O., Garrido, A., Valencia, J. L. & Tarquis, A. M. (2018), 'Using geographical information system to generate a drought risk map for rice cultivation: Case study in babahoyo canton (ecuador)', *Biosystems Engineering* **168**, 26–41.

Wang, J., Neskovic, P. & Cooper, L. N. (2007), 'Improving nearest neighbor rule with a simple adaptive distance measure', *Pattern Recognition Letters* **28**, 207–213.

Willmott, C. J. & Matsuura, K. (2005), 'Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance', *Climate research* **30**(1), 79–82.

Writing custom django-admin commands (2019).
URL: *https://docs.djangoproject.com/en/2.2/howto/custom-management-commands/*

Yao, S., Chunhong, L., Qiao, D. & Yixuan, W. (2014), Research of sf6 pressure gauge automatic reading methods based on machine vision, *in* 'International Conference on Computer and Computing Technologies in Agriculture', Springer, pp. 534–545.

Ye, X., Xie, D. & Tao, S. (2013), 'Automatic value identification of pointer-type pressure gauge based on machine vision.', *JCP* **8**(5), 1309–1314.

Zhang, D., Lu, G. et al. (2001), A comparative study on shape retrieval using fourier descriptors with different shape signatures, *in* 'Proc. of international conference on intelligent multimedia and distance education (ICIMADE01)', pp. 1–9.

Zuiderveld, K. (1994), *Contrast limited adaptive histogram equalization*.

Appendices

A. K-NN Classification Statistics

A.1 Classification Accuracy

Table A.1: Classification accuracy achieved for classifications performed with a uniformly weighted k-NN classification.

Fourier descriptors	Neighbors	Accuracy (%)
10	5	76.6
10	7	75
10	9	76
10	11	76.6
20	5	75
20	7	76
20	9	75
20	11	76
30	5	75.5
30	7	75.5
30	9	76
30	11	74
40	5	74.5
40	7	75.5
40	9	75.5
40	11	76.6
50	5	76
50	7	75.5
50	9	76.6
50	11	75.5

Table A.2: Classification accuracy achieved for classifications performed with a distance weighted k-NN classification.

Fourier descriptors	Neighbors	Accuracy (%)
10	5	74.5
10	7	74.5
10	9	75
10	11	76.6
20	5	75.5
20	7	76
20	9	76
20	11	75.5
30	5	75.5
30	7	75.5
30	9	75.5
30	11	75.5
40	5	75.5
40	7	75.5
40	9	75.5
40	11	73.4
50	5	76
50	7	75
50	9	73.4
50	11	76.6

A.2 Reliability Scores

Reliability Score for Uniformly Weighted Classification

Correct

Table A.3: 5 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	147	144	145	143	146
mean	0.98	0.97	0.98	0.98	0.97
std	0.09	0.12	0.09	0.09	0.1
min	0.6	0.4	0.6	0.4	0.6
25%	1.0	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.4: 7 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	144	146	145	145	145
mean	0.97	0.97	0.97	0.97	0.97
std	0.1	0.1	0.1	0.1	0.11
min	0.43	0.43	0.43	0.43	0.43
25%	1.0	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.5: 9 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	146	144	146	145	147
mean	0.96	0.97	0.96	0.97	0.96
std	0.11	0.09	0.11	0.1	0.11
min	0.44	0.56	0.44	0.44	0.44
25%	1.0	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.6: 11 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	147	146	142	147	145
mean	0.96	0.96	0.97	0.96	0.95
std	0.11	0.12	0.1	0.1	0.13
min	0.45	0.36	0.45	0.45	0.45
25%	1.0	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Incorrect

Table A.7: 5 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	45	48	47	49	46
mean	0.83	0.82	0.83	0.85	0.83
std	0.2	0.2	0.21	0.19	0.22
min	0.4	0.4	0.4	0.4	0.4
25%	0.6	0.6	0.6	0.8	0.65
50%	1.0	0.9	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.8: 7 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	48	46	47	47	47
mean	0.8	0.81	0.81	0.83	0.81
std	0.2	0.21	0.22	0.2	0.21
min	0.43	0.29	0.29	0.29	0.29
25%	0.68	0.61	0.57	0.71	0.57
50%	0.86	0.86	0.86	0.86	0.86
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.9: 9 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	46	48	46	47	45
mean	0.79	0.81	0.81	0.81	0.8
std	0.21	0.21	0.21	0.22	0.21
min	0.33	0.22	0.33	0.33	0.33
25%	0.58	0.67	0.67	0.61	0.67
50%	0.89	0.83	0.89	0.89	0.89
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.10: 11 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	45	46	50	45	47
mean	0.79	0.79	0.78	0.81	0.79
std	0.21	0.22	0.23	0.22	0.24
min	0.36	0.27	0.27	0.27	0.27
25%	0.64	0.57	0.64	0.64	0.59
50%	0.82	0.82	0.82	0.91	0.91
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Reliability Score with a Distance Weighted k-NN Classification

Correct

Table A.11: 5 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	143	145	145	145	146
mean	0.97	0.98	0.98	0.97	0.97
std	0.1	0.09	0.08	0.1	0.1
min	0.39	0.58	0.6	0.41	0.41
25%	1.0	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.12: 7 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	143	146	145	145	144
mean	0.98	0.97	0.97	0.97	0.97
std	0.08	0.09	0.1	0.11	0.09
min	0.55	0.43	0.42	0.42	0.55
25%	1.0	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.13: 9 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	144	146	145	145	141
mean	0.97	0.96	0.97	0.97	0.97
std	0.1	0.1	0.1	0.1	0.09
min	0.43	0.46	0.44	0.43	0.55
25%	1.0	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.14: 11 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	147	145	145	141	147
mean	0.96	0.96	0.96	0.97	0.95
std	0.12	0.12	0.12	0.09	0.12
min	0.36	0.37	0.44	0.48	0.37
25%	1.0	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Incorrect

Table A.15: 5 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	49	47	47	47	46
mean	0.82	0.81	0.85	0.84	0.84
std	0.2	0.22	0.2	0.2	0.2
min	0.38	0.38	0.39	0.37	0.4
25%	0.6	0.61	0.78	0.77	0.61
50%	0.81	0.82	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.16: 7 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	49	46	47	47	48
mean	0.78	0.83	0.8	0.82	0.81
std	0.22	0.22	0.22	0.22	0.21
min	0.28	0.28	0.28	0.28	0.27
25%	0.57	0.71	0.6	0.71	0.71
50%	0.86	1.0	0.86	0.86	0.86
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.17: 9 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	48	46	47	47	51
mean	0.78	0.8	0.8	0.8	0.78
std	0.22	0.23	0.22	0.22	0.23
min	0.33	0.33	0.32	0.34	0.23
25%	0.63	0.56	0.56	0.56	0.56
50%	0.84	0.89	0.89	0.88	0.81
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

Table A.18: 11 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	45	47	47	51	45
mean	0.79	0.81	0.78	0.76	0.8
std	0.22	0.21	0.24	0.24	0.23
min	0.27	0.35	0.29	0.28	0.35
25%	0.63	0.63	0.55	0.56	0.55
50%	0.9	0.9	0.83	0.82	0.91
75%	1.0	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0

A.3 AED Statistics for k-NN Classifications

AED Results with a Uniformly Weighted k-NN Classification

Correct

Table A.19: 5 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	147	144	145	143	146
mean	0.32	0.32	0.32	0.33	0.33
std	0.2	0.19	0.18	0.2	0.18
min	0.03	0.03	0.04	0.04	0.04
25%	0.19	0.19	0.18	0.19	0.2
50%	0.25	0.26	0.27	0.26	0.27
75%	0.41	0.41	0.42	0.42	0.42
max	1.17	0.91	0.92	1.24	0.91

Table A.20: 7 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	144	146	145	145	145
mean	0.32	0.33	0.33	0.34	0.33
std	0.19	0.2	0.19	0.21	0.19
min	0.03	0.03	0.04	0.04	0.04
25%	0.2	0.19	0.2	0.2	0.2
50%	0.26	0.27	0.27	0.27	0.27
75%	0.41	0.43	0.43	0.43	0.42
max	1.17	1.24	0.95	1.25	0.97

Table A.21: 9 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	146	144	146	145	147
mean	0.33	0.33	0.34	0.33	0.34
std	0.2	0.2	0.21	0.19	0.21
min	0.03	0.04	0.04	0.04	0.04
25%	0.2	0.2	0.2	0.2	0.2
50%	0.26	0.27	0.27	0.27	0.27
75%	0.42	0.43	0.43	0.43	0.44
max	1.18	1.24	1.25	0.92	1.26

Table A.22: 11 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	147	146	142	147	145
mean	0.33	0.34	0.33	0.34	0.35
std	0.2	0.21	0.18	0.2	0.2
min	0.03	0.04	0.04	0.04	0.04
25%	0.2	0.2	0.2	0.2	0.23
50%	0.27	0.28	0.28	0.28	0.28
75%	0.42	0.44	0.42	0.44	0.43
max	1.18	1.24	0.83	0.97	1.06

Incorrect

Table A.23: 5 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	45	48	47	49	46
mean	0.57	0.6	0.62	0.6	0.66
std	0.31	0.32	0.33	0.32	0.35
min	0.13	0.14	0.14	0.07	0.14
25%	0.33	0.36	0.38	0.34	0.4
50%	0.51	0.5	0.54	0.51	0.56
75%	0.77	0.79	0.86	0.82	0.91
max	1.27	1.33	1.35	1.36	1.57

Table A.24: 7 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	48	46	47	47	47
mean	0.58	0.63	0.64	0.64	0.62
std	0.32	0.32	0.33	0.32	0.33
min	0.06	0.14	0.14	0.15	0.14
25%	0.33	0.43	0.42	0.44	0.36
50%	0.5	0.56	0.55	0.56	0.51
75%	0.83	0.9	0.93	0.89	0.81
max	1.28	1.35	1.36	1.35	1.38

Table A.25: 9 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	46	48	46	47	45
mean	0.58	0.64	0.64	0.63	0.62
std	0.32	0.32	0.32	0.33	0.31
min	0.07	0.15	0.14	0.15	0.15
25%	0.34	0.43	0.44	0.38	0.4
50%	0.52	0.56	0.57	0.53	0.57
75%	0.8	0.92	0.91	0.88	0.79
max	1.28	1.35	1.36	1.37	1.36

Table A.26: 11 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	45	46	50	45	47
mean	0.6	0.63	0.65	0.63	0.65
std	0.31	0.32	0.33	0.34	0.33
min	0.14	0.15	0.15	0.15	0.15
25%	0.37	0.41	0.42	0.36	0.43
50%	0.52	0.56	0.57	0.56	0.57
75%	0.81	0.84	0.93	0.84	0.93
max	1.29	1.35	1.37	1.37	1.37

AED Results with a Distance Weighted k-NN Classification

Correct

Table A.27: 5 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	143	145	145	145	146
mean	0.31	0.32	0.32	0.33	0.33
std	0.18	0.2	0.2	0.2	0.19
min	0.03	0.03	0.04	0.04	0.04
25%	0.19	0.18	0.18	0.19	0.19
50%	0.25	0.26	0.26	0.27	0.27
75%	0.4	0.41	0.42	0.42	.42
max	0.89	1.23	1.24	1.24	0.92

Table A.28: 7 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	143	146	145	145	144
mean	0.32	0.33	0.33	0.33	0.33
std	0.19	0.2	0.2	0.2	0.19
min	0.03	0.03	0.04	0.04	0.04
25%	0.19	0.19	0.19	0.2	0.2
50%	0.26	0.27	0.27	0.27	0.27
75%	0.41	0.43	0.42	0.42	0.42
max	1.16	1.24	1.25	1.25	0.91

Table A.29: 9 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	144	146	145	145	141
mean	0.33	0.34	0.34	0.33	0.33
std	0.19	0.2	0.21	0.19	0.19
min	0.03	0.04	0.04	0.04	0.04
25%	0.2	0.2	0.2	0.2	0.2
50%	0.27	0.28	0.27	0.28	0.27
75%	0.41	0.43	0.43	0.43	0.43
max	1.17	1.24	1.25	0.92	0.94

Table A.30: 11 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	147	145	145	141	147
mean	0.33	0.34	0.34	0.34	0.35
std	0.2	0.19	0.2	0.19	0.22
min	0.03	0.04	0.04	0.04	0.04
25%	0.2	0.2	0.2	0.22	0.21
50%	0.27	0.28	0.28	0.28	0.28
75%	0.42	0.43	0.43	0.42	0.44
max	1.17	0.98	1.26	1.0	1.26

Incorrect

Table A.31: 5 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	49	47	47	47	46
mean	0.58	0.61	0.61	0.6	0.63
std	0.33	0.32	0.32	0.32	0.34
min	0.06	0.14	0.14	0.14	0.14
25%	0.33	0.37	0.38	0.35	0.37
50%	0.5	0.54	0.54	0.51	0.55
75%	0.8	0.83	0.86	0.8	0.89
max	1.28	1.35	1.34	1.35	1.36

Table A.32: 7 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	49	46	47	47	48
mean	0.58	0.61	0.62	0.63	0.64
std	0.31	0.32	0.33	0.32	0.33
min	0.06	0.14	0.15	0.15	0.15
25%	0.33	0.35	0.35	0.42	0.39
50%	0.5	0.55	0.55	0.56	0.56
75%	0.8	0.82	0.87	0.84	0.87
max	1.27	1.34	1.35	1.36	1.36

Table A.33: 9 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	48	46	47	47	51
mean	0.58	0.61	0.62	0.64	0.62
std	0.32	0.33	0.32	0.33	0.33
min	0.07	0.14	0.15	0.15	0.15
25%	0.32	0.35	0.38	0.38	0.36
50%	0.5	0.55	0.52	0.56	0.52
75%	0.78	0.83	0.81	0.9	0.81
max	1.28	1.36	1.36	1.36	1.36

Table A.34: 11 Neighbors

	10 Fourier Descriptors	20 Fourier Descriptors	30 Fourier Descriptors	40 Fourier Descriptors	50 Fourier Descriptors
count	45	47	47	51	45
mean	0.6	0.63	0.65	0.63	0.63
std	0.31	0.33	0.32	0.32	0.32
min	0.14	0.15	0.15	0.15	0.15
25%	0.34	0.38	0.45	0.43	0.41
50%	0.5	0.54	0.58	0.55	0.54
75%	0.81	0.82	0.9	0.82	0.84
max	1.28	1.35	1.36	1.36	1.37