Performance assessment of AI tools for digitizing ECG scans

Ioannis Adamopoulos

Thesis submitted for the degree of
Master in Applied Computer and Information Technology - ACIT
(Specialization: Data Science)

Department of Computer Science
Faculty of Technology, Art and Design

Oslo Metropolitan University — OsloMet

May 2022

Performance assessment of AI tools for digitizing ECG scans

Ioannis Adamopoulos

Performance assessment of AI tools for digitizing ECG scans

http://www.oslomet.no/

Printed: Oslo Metropolitan University — OsloMet

Acknowledgments

I would like to thank my main supervisor Pedro Lind as well as my co-supervisors Anis Yazidi and Hårek Haugerud for the critical guidelines and the valuable suggestions that gave me in order to make this master thesis as good as possible. Individually I would like to thank Professor Pedro Lind for being an inspired teacher with consistency and professionalism and for the important knowledge that he shared with me inside and outside the classroom during this master's program. In addition, I would like to thank Sushil Acharya for developing the tool that I present in this thesis and for giving me the permission to use it and test it. This tool couldn't be properly assessed without the dataset that we were kindly provided by Akershus University Hospital. In particular i would like to thank Doctors Helge Røsjø and Magnus Lyngbakken for their contribution. Finally, this master thesis is dedicated to my beloved mother in Greece.

Abstract

The digitization of electrocardiogram (ECG) signals recorded on paper is a very challenging task usually prone to errors and inaccuracies. Until now there is no available tool which can perform that task both universally and in a fully automated way. ECG is a well established medical modality to record the activity of the human heart which dates to over 100 years ago. Medical experts can diagnose potential heart irregularities by interpreting the recorded signals on ECG papers. There are millions of ECGs worldwide and the digitization of them is of paramount importance for research, analysis and diagnosis in medicine. By recognizing patterns in the digitized ECGs, artificial intelligence algorithms can predict cardiovascular diseases and help clinicians to make better medical diagnoses. Therefore, the development or the improvement of an automated tool that will be able to digitize massively ECG signals at once, is essential. In this thesis we describe and develop a tool for digitizing ECGs and we test its performance using ECG scans provided by Akershus University Hospital. In particular, we introduce some improvements which can make it operate in a more automated way. The original tool has several parameters in its various steps that prevent it from being fully automated. However, with proper further improvements, it has a great potential to become fully automated at least for ECG scans similar to the ones in the database of Akershus University Hospital. The current master thesis was held during the last semester of the ACIT's master program of Oslo Metropolitan University, namely from January 1st until May 15th.

Contents

List of Figures

Chapter 1

Introduction

1.1 Historical background and the problem of ECGs' digitization

In the very beginning of the 20^{th} century, Willem Einthoven discovered the electrocardiogram (ECG) which has been proved to be the most essential tool in cardiology and perhaps in the entire medical field [1]. Its wide use and popularity is not only due to its low cost and its non-invasive nature [2], but also due to its effectiveness in detecting cardiovascular diseases [3]. The function of the human heart is such that in every pulse it can produce electricity which can be detected through the electric potential differences created in the surface of our body [4]. ECG basically measures those potential differences that the heart generates during each of its cycle, by adjusting several electrodes on some special spots of the human skin. The results of the measurements are recorded on a specific paper known as ECG paper [5].

ECG papers were the only available format until the end of the 20th century. After that period, the ECG signals started to be recorded in digital format offering considerable flexibility [6]. In contrast to ECG signals on paper, digital ECG signals can be easily processed and manipulated. They offer all the benefits of a digital file like the simplicity in storing, accessing and transferring. Paper on the other hand remains a physical entity with a high probability of being partly or completely damaged by time. In cases where the patient have to keep the ECG paper and show it for a follow up examination, the reading and interpretation of the ECG signal might be difficult when the paper is torn or badly folded [7].

One can imagine the large number of ECG papers which have been produced in the past years. Even since 1987, there are many researchers who have introduced several methods and techniques for digitizing old ECG signals recorded on papers [8]. The digitization is based on the document's scanning and its transformation into a digital picture and then into a digital ECG signal. The whole process can be quite challenging

because despite the numerous efforts from scientists to digitize the paper-based ECG signals, no one has achieved to present a method that can be applied universally and in a fully automated way. It appears that there is not any technique yet which performs best in every step of digitization [6]. An important role for that, could play the fact that not every ECG paper has the same format or the same topology of its components on the paper. Therefore, it could make a common digitization process even harder. Based on the above facts, a couple of reasonable questions arise: Is it possible to have a fully automated tool which can digitize paper-based ECG signals? Can we overcome the challenges that appear in the various digitization steps and build a reliable digitization tool that can operate automatically without any human intervention? In this thesis we will try to explore and give answers to these questions.

1.2 Motivation: why to digitize ECGs?

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are a variety of dysfunctionalities of the heart and the blood vessels. These abnormalities can be related to the muscles and the valves of the heart for example, or the disability of the blood vessels to deliver the blood to different parts of the body [9]. Some of the main conditions of CVDs are heart attack, stroke, heart failure or arrhythmia [10]. The statistics related to CVDs are interesting and at the same time discouraging. In 2019 approximately 17.9 million people died from CVDs covering the 32% of deaths worldwide. 85% of those deaths were caused by heart attacks and strokes. In addition, CVDs represent almost 40% of all deaths under the age of 70 of people who had chronic diseases for the year 2019 [9]. The WHO points out the significance of the CVDs' detection at the early stages, in order to treat them on time with proper counselling and medication.

Since ECG is a very efficient tool in detecting CVDs and it has been used worldwide for many years, we can take advantage of the amount of information that we can extract from all these ECG signals that have been produced in the past and therefore contribute to the prevention of this important health issue for the society. The morphology of a digitized ECG signal is such that we can observe many different parameters such as typical interspike intervals and use them for making further statistical analysis. The collected data from a pool of ECG signals are perfect to feed machine learning algorithms, which have become very popular nowadays and are implemented in other medical fields as well [11]. Despite that the last years ECG signals are stored in digital format, they are still not enough for a sufficient statistical analysis. In order to draw conclusions about some clinical observations like sudden cardiac death, we need to have a large amount of data over time at our disposal. Therefore, it is necessary to digitize all the thousands of paper-ECGs from the past decades and offer them for AI analysis [12].

It has been reported that through this digitization, the scientific community will gain considerable knowledge on cardiac physiology, how arrhythmia works and how it is related to clinical diseases. A great number of valuable paper-ECGs from many clinical studies which concern people with CVDs can be found and be available in various health institutions. Another important source of non-digitized ECGs are countries with poor technical capacity [8]. We can conclude that an efficient and universal digitization tool is highly needed, in order to give us the opportunity to automatically digitize ECGs from patients for which we already know their full medical history and use the data for future AI tools in CVDs diagnosis. Akershus University Hospital which we collaborate with, is one of those health institutions that holds an enormous quantity of ECGs on paper, connected to patients with different diseases. This dataset contains over six million ECGs that were produced 8 – 10 years ago with known outcome of these patients. The digitization of those labeled ECGs could provide us undoubtedly a lot of valuable information and contribute to medical AI research.

Many machine learning and neural network algorithms have been already available in the literature which classify different types of CVDs. They can detect irregularities in the ECG signals and proceed to a diagnosis, which will help doctors to make the final decisions about the treatment selection [13]. There are times that ECGs need to run for hours in order for the doctors to observe an abnormality in the signal, because some symptoms may not occur during the few seconds that a regular ECG lasts. In those cases it is very difficult and time consuming for a human eye to detect where exactly the abnormality appears and therefore the assistance from the algorithm is essential [14]. Of course there are issues and limitations in the classification process [13] and the most often question which arises is who can see better or who can make better conclusions, the machine or the doctor's eye?

In [15] the authors compared how well a very rare variant of cardiomyopathy disease can be recognised in an ECG by both experts cardiologists, and machine and deep learning algorithms. The doctors scored the best specificity of all the models, but they were very poor in sensitivity and general accuracy, while all the models performed better. The authors argued that due to the sparseness of the disease the training dataset was not very large. We believe that the algorithms could have had even much better performance if there were more available data coming from the digitization of the old paper based ECGs. In any case both algorithmic models and medical specialists should complete each other for making accurate and safe diagnoses, because both have some benefits to offer. The former ones can be executed quickly, tirelessly without distraction or external pressure, while the latter ones have experience, knowledge and access to more information, details and other medical parameters that can be combined and produce better conclusions [16].

We also confirmed the above findings ourselves with the discussion we had

with the doctors from Akershus University Hospital. According to them, they are experienced enough to make proper diagnoses from every ECG. However there are some cases due to lack of time or personnel where they need a reliable automatic diagnosis from a computer. Specifically they mentioned the rare probability of patients having a stroke originating from hip fracture incidents. If the algorithm could predict that probability for every ECG taken from the numerous patients with hip fracture coming to the hospital, then they would treat them with special care trying to prevent the potential stroke. Obviously the investment in time and additional human resources for detecting such a rare incident is valueless and instead the assistance from the algorithmic models can be much cheaper and more effective.

1.3 What do we do?

In this project we first present a tool which digitizes the signals from ECG scans which have been provided to us by Akershus University Hospital of Oslo, for scientific research. The dataset does not contain any personal information of the patients and complies with all ethical requirements from Norwegian entities as well as the hospital.

The tool has been launched by Sushil Acharya, a master student from Oslo Metropolitan University, from July till December 2021 at the OsloMet's Centre of Research Excellence NordSTAR, Nordic Center for Sustainable and Trustworthy AI Research (details can be found in NordSTAR's website https://www.oslomet.no/en/nordstar). The purpose of the tool is to digitize as many ECG scans as possible and digitize them automatically without any human intervention. The goal in this thesis is to evaluate the functionality of each step of the tool by testing it on every ECG scan from our dataset. We then identify and report possible drawbacks that can be obstacles for the automation of the tool's function. We detect those parameters that need to be tuned and adjusted manually for the different scans or those that apply only for a specific group of scans in our dataset. Finally, we try to suggest solutions for some of the performance gaps that we identify and contribute to the improvement of the tool.

The current paper is organized as follows. In Chapter 2 we set the necessary theoretical background, so that the reader can be familiar with the main concepts, ideas and tools related to our project. In addition, we present the related work from other researches including the similar or different approaches compared to ours, as well as their results. In Chapter 3 we present our dataset as well as other datasets that are publicly available. Moreover, we explain in detail how our tool works and we describe every step of it. In Chapter 4 we present the dysfunctionalities of the tool for every of its steps after the tests we did, providing some examples. In Chapter 5 we introduce approaches for solving some of the challenges that we identified in the previous chapter and we show the results. Finally in Chapter 6 we discuss the results trying to explain the reasons for potential failures based on the challenges we had to

face and give ideas for future work.

Chapter 2

Background and related work

2.1 ECG signals and ECG data

Human heart is a muscle that periodically performs contractions (systoles) for pumping blood and deliver it to the body through a circulation system. Contractions are activated by electrical stimulations coming from the sinus node of the heart and travelling through neurons. Before a systole happens, heart is in an electrically polarized state (resting state) and during the systole it depolarizes. Those two events occur sequentially and have the duration of a heartbeat. The above electrical activity can be measured in the form of potential difference in the human body with the help of electrons attached on different spots. Electrocardiogram signal is a time series graph which shows how these potential differences change over time [17].

ECG signals are widely used by clinicians for ECG analysis because they contain an important volume of information. Their different waveforms describe the concatenation of all the electrophysiological events that take place during a cardiac cycle. By measuring some of the waves' properties like amplitude and duration or examining their morphology, doctors can recognise critical heart diseases and abnormalities like cardiac arrhythmia [18]. The early and precise detection of arrhythmia by studying ECG signals can prevent deadly incidents and reduce a considerable amount of medical costs globally [19].

ECG signals are recorded on graph paper which consists of squares of 1 mm^2 size. During printing, the graph paper passes under the printing pen usually with a speed of 25 mm/s which means that every 1mm in the x axis corresponds to 0.04 s or 40 ms time duration. In the y axis we measure the amplitude of the signal which represents the voltage. Commonly every 10 mm (10 little boxes of 1 mm^2 size) in the y axis of the graph paper corresponds to 1 mV. Different values of paper speed and number of boxes are used occasionally, and these parameters are written on the bottom of the ECG paper [5].

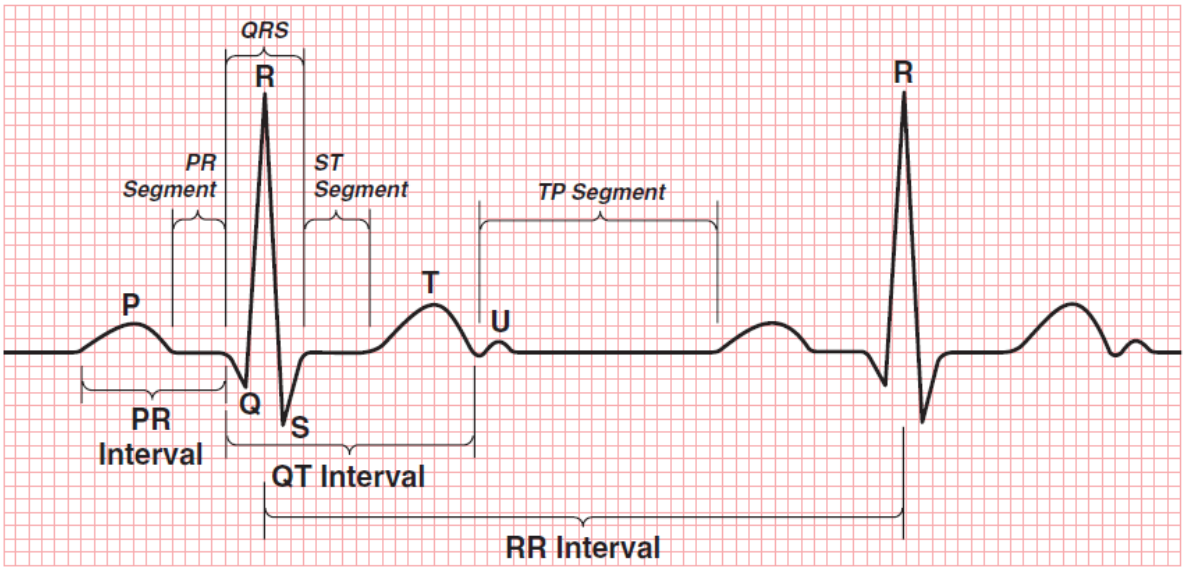


Figure 2.1: Anatomy of an ECG signal [20].

Figure 2.1 illustrates the basic components of an ECG signal during two cardiac cycles. We can observe five waveforms (P, QRS, ST, T and U), four intervals (RR, PR, QRS and QT) and three segments (PR, ST and TP). All these features are essential for monitoring the heart's functionality or detecting any anomalies. The five waveforms' sequence represent a full cardiac cycle of the heart's electrical activity. The P wave shows the beginning of the heart's contraction where the atria is stimulated (atrial depolarization). The QRS wave (or QRS complex) shows the further stimulation of the ventricles (ventricular depolarization) while the rest three waves (ST, T and U) describe the return of the ventricles to the resting state (ventricular repolarization) [20].

Different shapes and values of the above parameters in an ECG signal comparing to normal, can be interpreted as dysfunctionalities. For instance, a very tall QRS complex is usually the result of the hypertrophy of at least one ventricle and it can indicate premature ventricular contraction. On the other hand, a symmetrical form of T wave means that the patient might have a pathological state like ischemia [21]. An example of an abnormal ECG signal is shown in figure 2.2 and concerns the atrial fibrillation (AF) episode. AF is a type of arrhythmia which refers to the occurrence of irregular heartbeats. It happens because of the rapid, continuous, and sometimes simultaneous electrical stimulations of the atria without any constant tempo. The atria is unable to perform proper contractions and send the blood to the ventricle so that it can in turn contract properly as well [22]. The result is to observe random pulses without steady rhythm on the ECG signal. In addition, there is not any P wave present like it appears in a normal signal and instead it is replaced by many other random waves

[23]. Moreover, we can see that the QRS complex lasts shorter than the normal one and that's why it looks narrower.

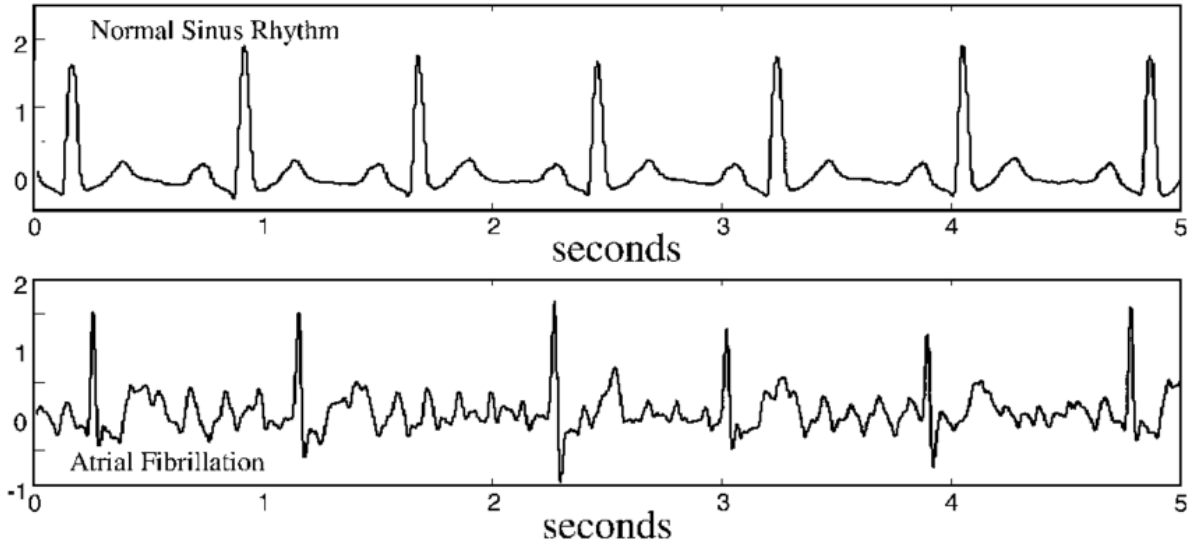


Figure 2.2: Top: Normal ECG signal. Bottom: Abnormal ECG signal caused by atrial fibrillation episode [23].

The effective diagnosis of any heart's disease requires the monitoring of its electrical activity from different angles and perspectives. After all, a proper ECG, also known as a 12 lead ECG, represents a three-dimensional supervision and recording of heart's electrical signals. That is the reason why multiple electrodes are placed in various parts of the body (leads) for measuring electric potential differences. There are 12 different voltage measurements made by 10 electrodes. 4 electrodes are attached to arms and legs (right leg is used as a ground spot) and they carry out 6 measurements which represent the 6 limb leads (I, II, III, aVR, aVL and aVF) and cover the frontal plane of the human body. The other 6 electrodes are placed around the chest area and they represent the 6 precordial leads (V1, V2, V3, V4, V5 and V6) which cover the horizontal plane of the body [20]. Figure 2.3 shows a typical 12 lead ECG which includes four strips. In the first three strips there are the 12 signals from the different leads and we obviously expect them to have different morphology and polarity from each other. In the fourth strip there is the signal of one selected lead with longer duration for better observation. For calibration reasons each strip usually starts with a pulse which has 10mm height.

ECG is produced by special devices which are called ECG machines and consist of four main components. First, we have the electrodes which were mentioned previously and are responsible for detecting the electrical signal. Second, there are the wires which transfer the signal from the electrodes to the amplifier. The amplifier (third component)

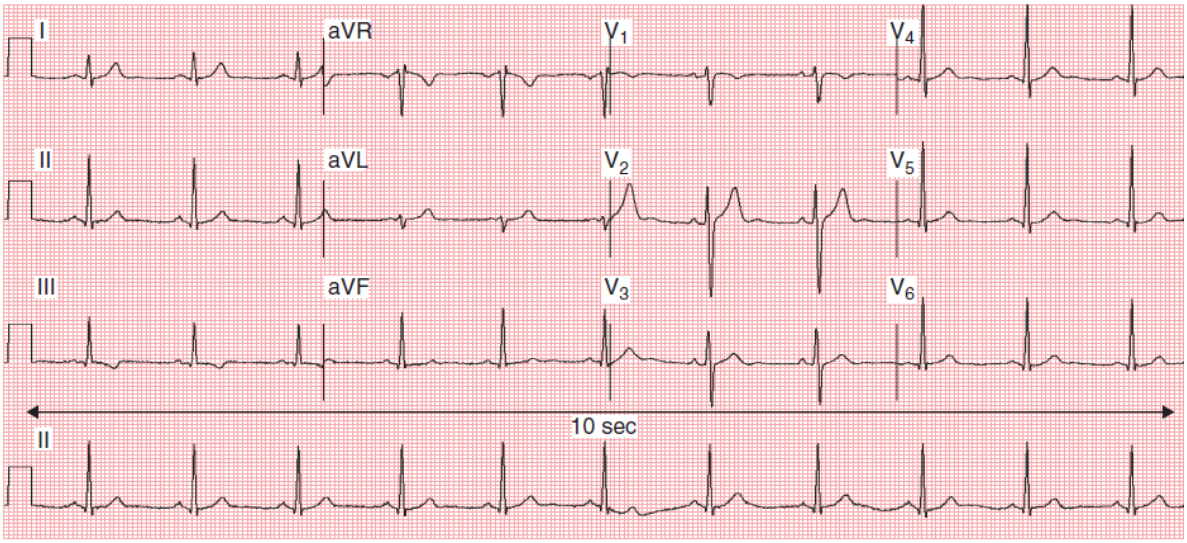


Figure 2.3: A typical 12 lead ECG [20].

receives the signal and make it stronger. Finally, the last component is the output where the signal is converted from analog to digital and is then recorded or displayed in different ways including graph paper, computer and oscilloscopes [24].

Nowadays, there is a big variety of ECG machines in size, price, or type. One can find expensive brands used in hospitals and clinics which offer high quality and reliability. On the other hand, a big number of portable or wearable ECG devices have flooded the market and anyone can record an ECG even though the process is performed with limited leads (1-6). There are many vendors who supply the market with ECG machines, but the leading ones are GE Healthcare, Philips Healthcare, Nihon Kohden Corp., Medtronic, Spacelabs Healthcare and Schiller AG. In 2018 all of them together owned the 55% of the total market share. The plethora of vendors has arisen interoperability problems for many years because each one of them uses its own ECG management system. Today it is very essential that all kinds of ECG data from any device, are integrated under one common open platform like DICOM waveform standard for storing, sharing and processing. Some vendors are already trying to switch their technology for serving that purpose [25].

The ECGs which have been produced by various machines over the years can appear different format and topology. A basic categorization of the ECG format is based on the alternative ways that the leads can be illustrated on the screen or recorded on the paper. Single channel ECG machines (lower in price) can record only one lead waveform at a time producing a long and narrow ECG paper with all the leads next to each other. This is called 1x12 ECG format (left part of figure 2.4). Three channel ECG machines can print three waveforms at the same time so a 12 lead ECG can be recorded

in four sections as we can see in the right part of figure 2.4. Following the same logic, six channel and twelve channel ECG machines which are more expensive, produce 6x2 and 12x1 ECG formats respectively (figure 2.5) [26]. Optionally in the above formats, there is a long record of a specific lead at the bottom of the ECG.

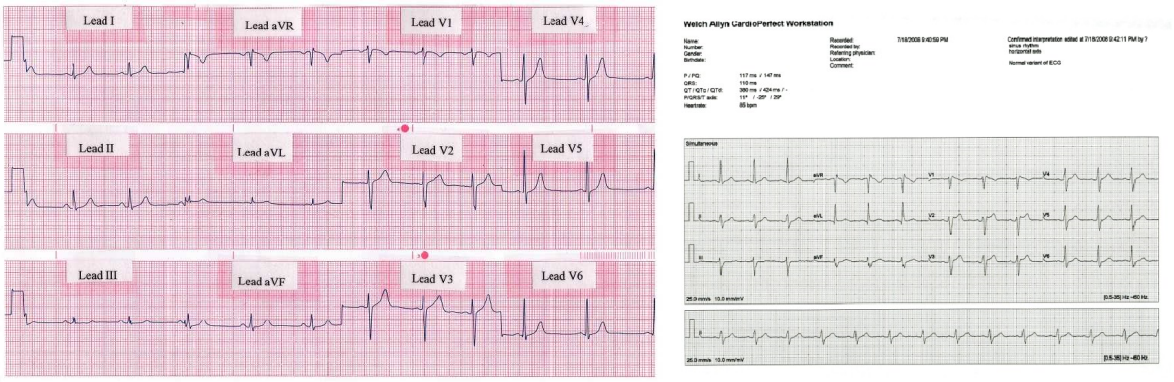


Figure 2.4: Left: 1x12 ECG format on a long and continuous paper, cut and illustrated in three parts [26]. Right: 3x4 ECG format (Welch Allyn supplier) [27].

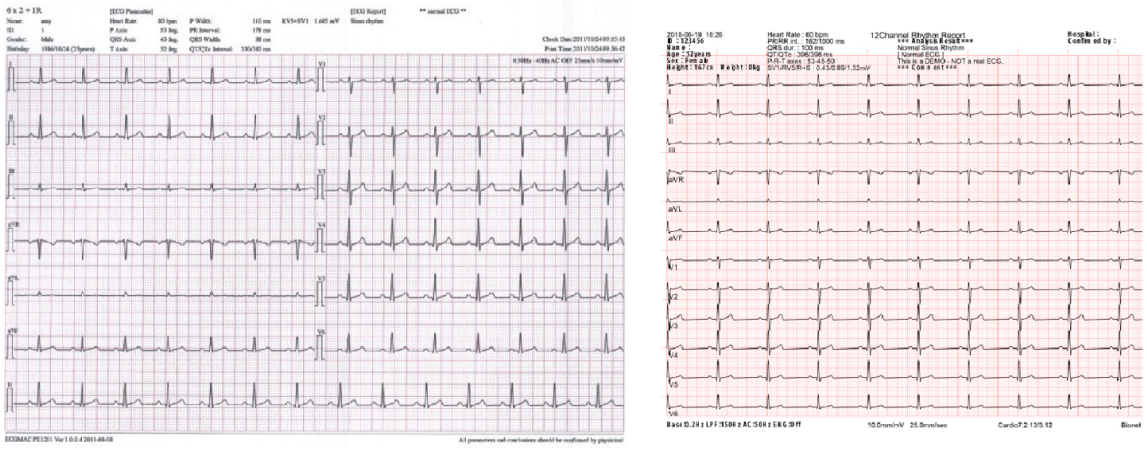


Figure 2.5: Left: 6x2 ECG format (ECGMAC supplier) [26]. Right: 12x1 ECG format (Walen Tec supplier) [28].

The identification of the various ECG formats is important for the purpose of this project since we are dealing with the digitization of ECG scans. However, this is not the only challenge when it comes to digitization process, since there are other types of variations or anomalies between the ECG scans. One of them is whether the background is colored or not because as far as we know there are many scans

in black and white. Others have overlapping signals in neighboring strips or noise from ink spill. In addition, some ECGs have different recording speed on the paper and the selected long lead at the bottom is not always the same. We can also observe variations of how the different lead signals in a single strip, are separated from each other (vertical lines, dot lines, spaces, etc.). Finally the calibration pulse can be found either in the beginning of the strip, or in the middle before a new lead signal starts, or at the end of the strip. An ECG scan having some of the above variations is shown in figure 2.6. Some of the variations of this particular scan will be also addressed in this project as part of our dataset.

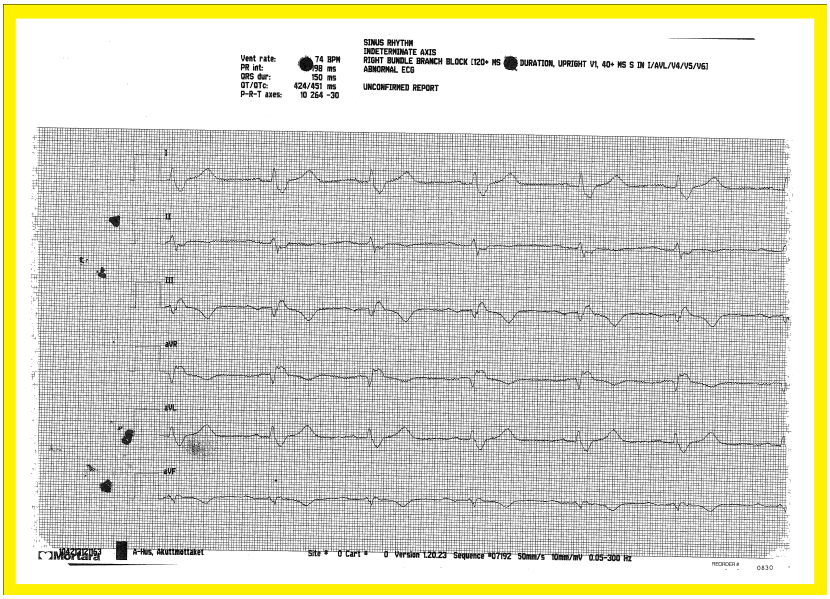


Figure 2.6: A noisy black and white ECG scan with 50mm/s recording speed (Mortara supplier). Source: Akershus University Hospital.

2.2 Concepts on computer vision

The human eye has the ability to capture images from physical environment with a great perception of their three-dimensional properties. We humans can very easily distinguish the various objects around us and recognize their different shapes and sizes. We can identify an object as a whole even if some parts of it have different color, brightness, have a very similar background or are covered by other objects. A person is capable of recognizing human faces, guessing their ages or even talking about their feelings that they have that moment, only by looking a picture for example [29]. These

human skills may seem natural and taken for granted but scientists still try to find out and understand how the human vision functions, involving the eyes and the brain [30].

For many decades people have been trying to develop tools and algorithms, so that a machine can perform like the human eye and therefore the concept of computer vision (CV) was introduced. The goal of CV is to use techniques for processing images or videos at pixel level in order to perceive them as we do. This means that an algorithm or a CV system can extract information, make conclusions and interpret what it "sees" by analyzing an image [31]. The above procedure is highly challenging even until now despite the multi-year research that has been done. An image usually contains a very large amount of complex visual data that makes it difficult for CV to clearly identify objects and their boundaries [32]. It will take many years so that CV can detect everything in a picture as a two year old kid would [29].

However, the last years CV has been developed dramatically mainly because of the significant increase of the computational power. Computers today have very powerful memory and storage capacity while in the early years a special device was required just to store a single image with good quality. In addition, the machine learning field has known a huge progress until today, it has become more understandable by humans and therefore has been used largely in CV. In the coming years, we surely expect that CV will grow more, especially with the support of artificial intelligence [32].

Let's describe how CV was evolved over the years and what were the main ideas about it in each decade. When the concept of CV was born in the beginning of the 70's, the first steps to detect the 3D form of an object in a 2D image, was the edges extraction method. Later, more quantitative approaches followed like feature-based stereo correspondence algorithms and intensity-based optical flow algorithms. During the 80's the focus was on quantitative image analysis using more advanced mathematical methods. The tools that have been used that period was versions of image pyramids, shape from shading, shape from texture and shape from focus techniques, as well as 3D physically based models and regularization-based surface reconstruction. In the 90's, the techniques from the previous decade were improved and the statistical learning algorithms were introduced such as the principal component analysis for face recognition. The most important thing though, was the appearance of computer graphics mainly in the field of image-based modeling which came from the idea of creating animations from real world pictures. During 00's computational photography methods appeared and feature-based techniques for object or scene recognition. However, the biggest achievement which set new rules in CV until today was the development of machine learning algorithms and an important reason for that was the huge number of digital labeled data on the web. The following decade, CV made a tremendous progress based on massive data sets and implementation of sophisticated algorithms like deep neural networks [29].

At this point it is worth to mention image processing which is a related area to

CV and is essential for its implementation. Image processing was used in the late 60's from NASA and basically applies mathematical transformations on digital images resulting specific effects like smoothing, sharpening or contrasting [33]. We could say that CV uses the image processing tools to prepare an image before it applies intelligent algorithms to it for pattern recognition [31]. In reality what an algorithm "sees" is not a picture as humans perceives it, but instead it is a one-dimensional array of numbers. The example in figure 2.7 will make things clearer. Here we can see a digital grayscale picture of Abraham Lincoln which consists of several pixels and for each pixel a 8 bit number is given. The value of each number depends on the pixel's brightness and varies from 0 (black) to 255 (white). In the third part of the figure, we can see the image represented only by numbers which are stored in the computer memory as an array. Thus, whenever we apply image processing or CV techniques and algorithms to an image, we retrieve and manipulate those numbers [34].

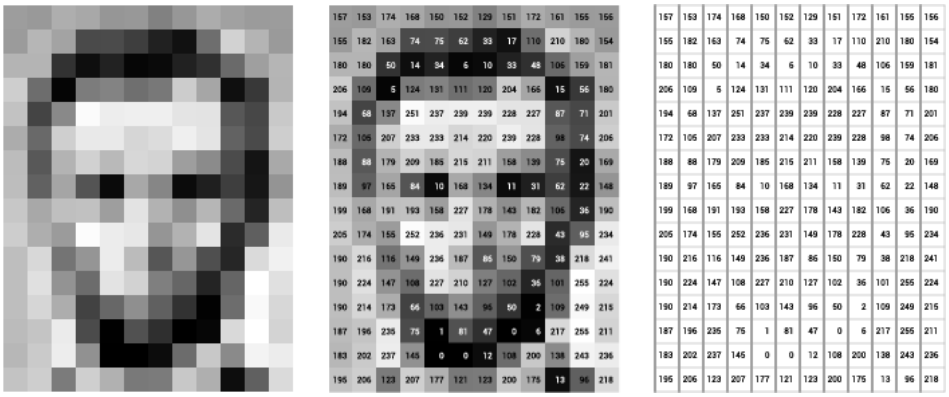


Figure 2.7: Pixel-wise presentation of a digital image [34].

Pattern recognition as was referred above can be applied in CV area and both of them can be considered related fields. While the first one represents a broader field of applications, it can be useful to CV for achieving main tasks like image recognition. Pattern is an entity that have certain repeated characteristics which can make them unique, interesting, or worth detecting. We can observe patterns in many ways with our own eyes everywhere in nature around us. They can be in images, in videos, in faces, in handwritings or in speech signals for example. But we can also extract patterns from data that we are not able to see clearly, using statistical algorithms [35].

It is believed that humans make decisions by recognizing patterns and therefore we want machines to imitate that behavior. Utilizing machine and deep learning algorithms, computers can identify patterns from the environment and categorize them through logical steps. This classification task is achieved either by comparing the observed pattern to known pattern categories (supervised learning) or by discovering new trends and patterns which do not belong to any predefined category

(unsupervised learning). Nowadays most pattern recognition techniques are based on artificial intelligence and can be implemented to numerous applications like the recognition of speech, image, face, emotion, movement, signals and characters or handwriting in a text. Pattern recognition is an integral part of CV which supports its main ideas and concept as described in the beginning. The difference between them is that pattern recognition can take all kinds of data as input like text, images or audio files while CV concentrates on images [35].

Modern applications around us are based on CV and many users do not even realize it. Augmented reality (AR) is one of them and people use it in many apps on their mobile phones such as Snapchat where they can apply different virtual filters on real images and videos. AR combines the real and the virtual world by placing virtual objects inside a real-world image, after the detection of the area of interest from machine learning algorithms. CV can also read real data from the physical environment with the use of the lens from the phone's camera. Apps like Google Lens can read texts, barcodes or translate signs on the street. The camera can be used for face recognition as well where the owner's face can be identified from the mobile phone and give him/her access to it [36].

CV algorithms can be applied for private or public security during the video recording from security cameras. For home cameras, the object recognition or movement detection techniques can be responsible for storing or sending to the IoT server, only the part of the video where the "suspicious" activity took place, saving a lot of energy and storage resources. In public, CV can be used for surveillance purposes for the face recognition of criminals or other suspects. Another great but challenging and still in progress application of CV is autonomous vehicles which gives them the capability of driving on their own without a human driver. A deep learning algorithm reads the environment using all the available cameras of the vehicle and therefore it can self-navigate avoiding obstacles like people or other cars [36].

Health industry is another field where CV is implemented and has undoubtedly made a big impact. Hundreds of scientific papers has been published the last years about CV tools in medical imaging. There are massive, labeled datasets available which can be used for training algorithms in order to perform medical diagnoses. Various patterns can be recognized in X-rays, in computerized tomography (CT) scans or magnetic resonance imaging (MRI) scans for example, where they can show tumor development or other undesirable diseases (figure 2.8). Moreover, in cardiology which is one of the most popular medical field, many deep learning algorithms has been applied for the detection of cardiac anomalies that are shown in the ECGs. Apart from predicting or recognizing potential health abnormalities, CV uses image processing techniques for the enhancement and reconstruction of noisy or bad quality medical images [37]. Such techniques are utilized in this project where the medical images are old ECG scans, and the purpose is to digitize and extract the ECG signals from them.

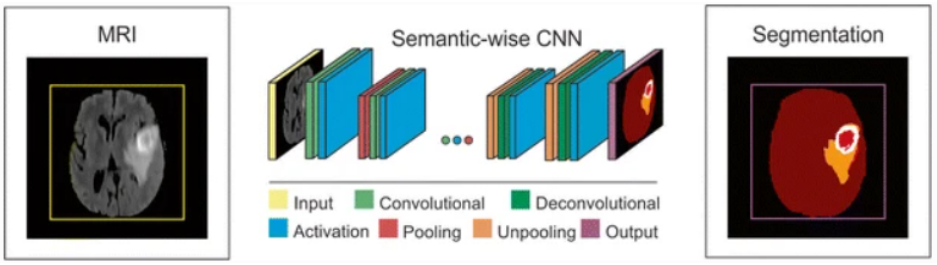


Figure 2.8: Schematic illustration of a CNN architecture for brain tumor segmentation task from an MRI scan [38].

2.3 Main concepts about signal digitization

The digital recovery of the various biomedical signals that have been recorded on paper and especially the ECG signals have attracted the attention of the scientific and medical community. Many ways and techniques have been proposed for the digitization of those signals, stepping on a procedure which appears to be relatively common for everyone [39]. Here we will present and explain the basic steps for signal digitization, and we will mention some related usual techniques that have been implemented in the literature. Briefly the steps are as follows. The first step is the scanning of the medical paper, the second step is the correction of the image orientation, the third step is the grid removal, the fourth step is the noise removal and some additional processing and the last step is the signal extraction.

For the scanning, a flatbed scanner is commonly used and the resolutions are usually 200 or 300 dots per inch (dpi) but can also be 600 and 1200 dpi [39]. The scanning resolution together with the recording speed and the amplitude of the calibration pulse which are written on the paper, can give us the value of each pixel in the time and amplitude scale which will be later used for the calibration of the extracted digital signal. The paper can be scanned in various digital formats like Jpeg, Tiff, Bmp or Png and the resulting image can be either colored, or gray scaled or black and white [40].

Due to human factor or defective scanner, the scanned images might be skewed and this creates a problem that needs to be solved [40]. A skewed image results from the wrong orientation of the paper inside the scanner. A 0.5 degree deviation angle from the right orientation is enough to make the whole process more complex and produce false extracted signals [41]. Thus, the implementation of the second step of the digitization procedure is necessary. The deviation angle is basically equal to the slope angle of a specific gridline of the image's background. The angle can be found by initially selecting any two points on that gridline. The correction of the deviation angle afterwards, happens by rotating the image until the angle is zero [8].

A very basic and important step for the signal digitization is the grid removal.

All biosignals in general are printed on a grid paper so that they can be easily read and interpreted, but of course this kind of background in our case is unwanted and therefore it has to be separated from the actual signal in the image [39]. The separation methods are usually based on the different characteristics between the grid and the signal that can distinguish them. It could be for example the thickness between the signal and the gridlines or their different color [8].

A very popular image processing method for the separation of the signal from its background, is the binarization of the image, which means that all the colored pixels of the image are converted to black or white pixels [41]. For achieving this, the finding of a threshold value which decides which pixels will become white and which pixels will become black, is an essential task. A reliable and often used tool for accomplishing that task is Otsu's algorithm [40]. The image binarization make sense when the available images are colored or gray scaled, but it hasn't any value when the scans are already binary (black and white). In that case the separation process becomes much more challenging since the signal and anything on the background have the same pixel values [39].

The previous step might not always lead to a full separation of the signal from its background. It can leave behind some leftover noise like gridline pixels which are attached to the signal or are randomly distributed in the background as isolated dots. At the same time, it can also add some additional noise after the process is completed. Therefore, at the next step, noise removal methods are applied. One of the most popular noises called "salt and pepper noise" can be removed by the implementation of a 3x3 median filter or the eight-neighbor tracing filter. Other noises include unwanted written characters or annotations on the image and can be removed by tools like optical character recognition [8].

In the current step, additional image processing techniques are applied in order to make the signal tracings as smooth and clear as possible and to improve the accuracy of the ECG parameters (e.g., high and low pass filter or alignment technique). In addition, continuity algorithms need to be utilized for filling the gaps that the signal might have, because it is important that the signal appears as a one piece during its extraction [8]. Those gaps can occur from ink failure during printing or even from other techniques that have been applied earlier, like the image binarization. Other signal improvement techniques can include thinning algorithms for the homogeneity of the signal's thickness [8] or morphological operations like erosion and dilation [40]. The selection of the applied processing method depends each time on several parameters like the quality or morphology of the signal in the picture.

The last step of the digitization process is the extraction of the digitized signal and its storage in a digital format. Here the signal is encoded in a 1d vector which contains the x and y coordinate values so that it can be plotted afterwards as a time series signal for evaluation. The coordinate values can be calculated if we know the x and y values

that each pixel represents in the image as we mentioned in the first step. According to [8] the evaluation of the accuracy of the digitization process can be quantitatively or qualitatively where we compare visually the digitized signal to the corresponding signal in the image. Quantitative evaluation usually concerns the comparison between the digitized signal extracted from the image and the same signal which was recorded simultaneously and stored straight in a digital format. In that case we can compare individual characteristics of the signals like their intervals (PR, QRS etc.), amplitudes, properties between waves or harmonics using Fourier analysis. The utilized statistical evaluation tools include root mean squares, Pearson correlation coefficients, median absolute deviation, kappa statistics and others.

The above sequence of steps is a general procedure for obtaining an ECG signal from a paper format and transforming it into a digital format. Variations of this procedure can occur depending on the different datasets, needs, challenges or the different approaches and point of views of those who work on the ECG digitization. For instance, skew correction might be skipped in some cases, or some of the processing techniques that were described above are not necessary or are implemented in different steps. Moreover, the whole process concerns the digitization of a single waveform in a specific region of interest. Additional techniques need to be applied for the extraction of multiple waveforms or for cropping specific parts of the image depending on the various ECG formats as they were presented in section 2.1 Thus, in the next section we will discuss the different approaches and methods of several proposed digitization tools that one can find in the literature.

2.4 Main works on ECG digitization

Going through the existing literature, we did not find any fully automated solution of digitizing paper-based ECGs. It seems that the problem remains timeless until now, despite all the scientific research that have been carried out. The various projects and tools do not deal with all the challenges of the digitization process at the same time. Instead, they focus on specific ECG datasets, generated by systems with particular standards and they suggest solutions only for some parts of the problem.

In [42] the authors try to solve and focuses on the issue of the overlapping lead characters with the ECG waveforms using Optical Character Recognition (OCR) function in Matlab. During preprocessing steps, they first corrected the skew angle with Hough Transform, they applied a thresholding technique to binarize the image and then they removed some salt and paper noise with median filter. Later they applied Connected Components Analysis for separating the ECG signal strips and OCR to remove the overlapping characters. Finally, they converted the raw signals into time series signals and evaluated their accuracy with the help of experienced cardiologists. Results showed that correlation coefficient between digitized and paper

ECG signals were almost in every clinical feature (like QRS complex) of the signal above 0.9. The whole process is executed in a Matlab-based GUI application where we observed the following: The tool works with specific ECG scans format with colored grid and from limited ECG vendors. In addition, the user needs to interact often manually by detecting the area of interest and indicating the lead characters for the OCR. The different lead signals in the same row must also be separated by the user which makes the process time consuming.

A similar approach with the previous one is presented in paper [43] where the authors used another Matlab tool. The highlights of this project are described below. For the removal of the background grid by binarizing the image, the threshold value were based on the histogram of the grayscale image (pixel intensity distribution) and the fact that the signal was darker than the grid. This was an expected finding since the scans were colored before they were transformed into grayscale images. Then, in order to find the outline of the ECG waveform, a column-wise pixel scan was applied. Moreover, they managed to store each digitized ECG with the corresponding patient's record in the database by reading patient's info in the scanned paper with OCR. The user though, is asked to mark the info area manually for the different ECG formats. Overall, they found a correlation value from 0.75 to 0.8 between the digitized signals and the ones in the paper with the help of clinicians. The clinical features of the signal were very well preserved after the digitization as well. One thing that is not clearly specified is whether and how the individual lead signals were extracted from a single row. We suspect that this is something that the user does manually.

Older tools like the one suggested in [44] are also based on the user's manual detection of the area of interest. The same thing happens with one of the most recent public available tools which is presented in project [12]. According to the writers, the specific tool managed to digitize only the 50% of their validation dataset. The tool is friendly and straightforward for the user, even though it requires several manual actions. However, when we tried to apply it ourselves on some of the ECG scans that we had at our disposal, the resulting digitized signals was far away from the ones in the scan, in terms of matching.

A solution for the automated distinguishment of the ECG leads in a single strip is partly presented in [11]. The Matlab tool that they propose is able to separate two ECG leads in a row allowing only the digitization of 6x2 (or 6x2 plus the long lead at the bottom) ECG scans. In order to achieve the automated (and not manually by the user) region detection of all the signals both between the different rows and inside a single row, the authors were based on the values of the standard deviation of the black and white pixels in each row and column of the image. Once they defined those areas, they proceeded with the extraction of the pure signals by initially binarizing the image with the Otsu's thresholding technique. Then they found the centerlines of the ECG signals with Hough Transform method and for the signal scaling, the calibration pulse

was used as a reference considering that it is at the same position in all the ECG scans, otherwise the user has to indicate that pulse manually. For the selected dataset and the specific ECG format, the tool performed quite well. The Pearson's linear vector correlation coefficient value which was used as an accuracy measurement was a bit more than 0.95 on average between the original and the digitized ECG records. There is a tutorial provided in YouTube but the tool can not be found available in public.

Another paper in the literature ([45]) claims in its description that it can deal with the issue of the simultaneous segmentation of the 12 areas of the different lead ECG signals without manually detecting them. However, it only explains and illustrates the procedure of separating the whole strips and not the signals inside the strips, something that is also achieved by others. Despite that, the way that they have approached the strip separation is interesting because all the horizontal strips from the binary image are projected on a vertical axis where the boundaries of the individual areas that need to be cropped, are identified from the gaps in that axis. This method focuses on the ECG images taken by a cell phone camera and has also some limitations like the requirement of colored background grind and the non-overlapping signals.

A variety of image processing techniques for some individual steps of the ECG digitization can be found in several papers. In [46] for instance an entropy-based bit plane slicing algorithm is introduced for extracting ECG signals from images with various types of degradation. This included shaded, blurry, folded, noisy, multicolored, or other similar images. The results approved to be almost perfect, but the process concerned images with only one ECG signal. The same thing applies also in project [47] where it focuses on extracting the ECG signals with morphological operations. An additional limitation here is that the ECG images must be scanned with resolution of 600 dpi (dots per inch). Moreover, the authors of [48] proposed Wiener filter for noise removal, Sauvola's thresholding algorithm for grid removal and Pan-Tompkins algorithm for ECGs' parameter extraction.

The ECG digitization problem can be also approached without the use of the ordinary steps for grid removal or signal refinement for instance. The authors of [49] have implemented the U-NET architecture which is a deep learning neural network, and the problem has been handled as an image segmentation problem. This means that after the training procedure of the algorithm, the separation between the signal and anything else in the image, occurs by labelling every single pixel (whether it belongs to the signal or not). The dataset contained only noisy and binary (black and white) ECG scans which make the digitization task harder and even that, the tool managed to achieve high performance. The digitized and the original ECG signals had 0.916 correlation coefficient overall. However, the provided solution could not deal with the overlapping waveforms of the signals.

In [50] we can also observe the participation of neural network in a specific part of the digitization. Despite the limitations of the suggested tool, it is interesting to

see how the authors here used a 3-layer deep learning neural network for finding the proper threshold value for each image, in order to binarize it. First of all, the used dataset was based on different colored photos of ECGs which were taken by a smartphone. The idea was to create a graph for each image with all the possible threshold values in the x axis (0 to 255) and a quantity in the y axis which was called normalized sum with possible values from 0 to 1. In addition, they tried to find manually the proper threshold value of 66 images which were going to be used as training dataset for the neural network together with the 66 corresponding graphs. The dataset was enough to train the algorithm and find the threshold value for every new image with accuracy 97%, taking the above graphs as inputs.

The goal of the previous tool though was the use of another deep learning model for the classification of the digitized signals into five potential diseases (94.4% accuracy). Many classification methods have been applied to ECG signals for predicting cardiac irregularities. The signals sometimes come from scanned ECGs where they need to be digitized first, like in the tool we just described, and some other times they come straight from a digital dataset like the one used in [2]. Here the writers have used a deep neural network with Tensor Flow framework and a deep learning library from Google, in order to classify heartbeats for arrythmia detection.

Finally, in the literature, one can also find a completely different point of view for the purpose of ECGs' digitization. In fact, the authors of [51] put on the table the argument that the whole procedure of digitizing old ECG scans for obtaining the raw ECG signals for the potential further research is not necessary. Instead, the utilization of deep learning algorithms for predicting cardiovascular diseases only by training them just with the ECG scans and without the signal digitization, can also be possible. In order to prove that, they tried to apply a convolutional neural network (CNN) on a labeled dataset of ECG images with cardiac anomalies. However, their dataset was generated from already digitized raw signals which were transformed afterwards into low resolution ECG images, and it did not contain original ECG scans.

Chapter 3

Data and tool description

3.1 Available ECG data and dataset description

To our knowledge there is a significant lack of available data which contains ECG images coming from paper-based ECG scans. This can also be confirmed by the different projects in the literature where the authors have been provided their datasets by authorized institutions like hospitals, cliniques, or other associations. Some papers do not mention the source of their dataset and some others state the difficulty of finding an open-source dataset. The fact that a full ECG scan contains personal information and sensitive medical data of the patient, does not allow it to be publicly available because that would violate the personal data protection. That is why we typically see collaborations between researchers and medical institutes.

However, we can find some ECG scans with various resolutions on the web where their info section is cropped. In the webpage [52] for example, which has educational purpose, one can find 100 3x4 ECG scans with a good resolution, but the images can only be downloaded manually one by one. In addition, we can find many ECG images inside medical handbooks or educational books which are digitally available also on the web. In [53] there are many noisy binary ECG scans, in [54] one can find various gray scale scans and in [55] there are less noisy gray scale scans produced by the same ECG machine. Nevertheless, the extraction of the images from those pdf files are an extremely time-consuming procedure because it requires manually identification, cropping and processing of each image. On the other hand, finding digital ECG signals on the web is quite easy. There are many datasets containing thousands of raw and labeled signals which can be offered for training machine learning algorithms for the classification of cardiovascular diseases. One of the popular databases is Physionet [56] which contains among other medical data, a variety of well described ECG datasets.

The dataset we have at our disposal comes from Akershus University Hospital of Oslo and contains 9 pair of ECG scans corresponding to 9 patients. Each pair of

scans concerns one single ECG examination performed by a 6 channel ECG machine (Mortara supplier). The paper speed during the recording is 50 mm/s and for printing 12 lead ECG signals in a 6x2 ECG format, it requires a printing area which has the double size of a A4 paper. Since a regular scanning machine can maximally scan a A4 paper, the ECG printout had to be cut in two pieces and scanned separately. So, we ended up having two different images for the same ECG (figure 3.1). If the paper speed was 25 mm/s as usual, then the printout would fit in a A4 paper and it could be scanned in one piece. The scan in that case would look like the one we saw previously in the left part of figure 2.5. Each scan has a tif image format and the image is black and white (binary) with dimensions 7016 x 4968 pixels and resolution of 600 dpi.

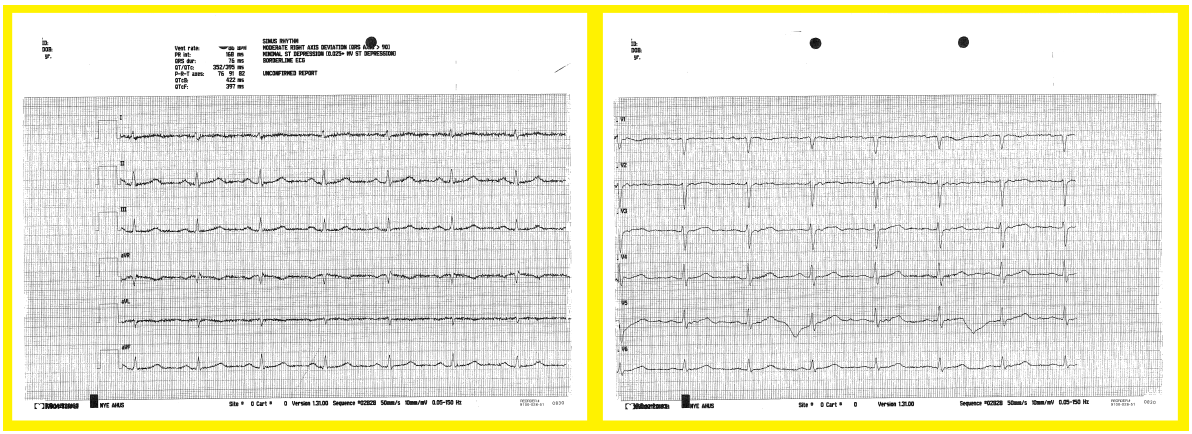


Figure 3.1: Two A4 paper size scans in a 6x2 ECG format. Data provided by Akershus University Hospital, with permission.

3.2 Presentation of the existing tool

In this section we will present our available tool which digitizes ECG signals from ECG scans and we will describe the steps that occur in the whole process. We have selected the scan which is illustrated in figure 2.6 as an example and we will demonstrate how it changes or how it is transformed throughout the digitization process. The tool has been designed with Python, using open cv2 as the main library. The rest of the used libraries include numpy, pandas, matplotlib and scipy. During the description of the process, we will also show the most important parts of the Python commands for better understanding.

3.2.1 Step 1: the area of interest

The first step of the process is to detect the area of interest, where the ECG signals are located. On the top part of every image there is an area with various information like the diagnosis and the heart rate of the patient which has to be removed. Therefore, the goal is to crop the image so that the remaining part is the grid with the ECG signals. Before the crop, first we need to process the image and apply some filters. After the image is loaded, we resize it by pixel for both x and y axis, so that its final dimensions in height and width are 40% of the original ones. This way, the resolution of the image becomes lower. Subsequently it is converted into a grayscale 8 bit image (cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)) and then a Gaussian blurring filter of 3x3 pixels is applied (cv2.GaussianBlur(image, (3, 3), 0)). The filter averages with Gaussian weights the value of every pixel with its neighbors (8 for our case) and the purpose is to smoothen the image and remove some outlier pixels which might cause some noise. It also prepares the image for the next step. The outcome of the Gaussian filter is a slightly blurry image.

Afterwards, we apply an operator which is called Canny edge detector (cv2.Canny(image, 120, 255, 1)). The operator finds discontinuities between the pixels like unexpected changes of the brightness and can detect borders, shapes or edges of the various characteristics of the image [57]. Inside the related python function, we insert the minimum and the maximum value of thresholds as arguments. The pixels whose values are bigger than the maximum threshold value are considered strong contributors to the edge detection while the ones with values between minimum and maximum threshold values are considered weaker. The rest of the pixels are ignored.

Canny edge detector makes it easier for finding the contours of different areas on the image, so that we can select the one we are interested in. As we mentioned before, our goal is to detect the contour which contains the grid area and therefore we use the Python function "findContours()" which can find all the possible contours of the image. One of the arguments in the function concerns the retrieval mode of the algorithm and in our case, we insert the “cv2.RETR_EXTERNAL” which means that it will retrieve only the extreme outer contours of the image since we do not want to detect the boundaries of every object in it. Another argument concerns the way of storing the contour points. Here we choose the “cv2.CHAIN_APPROX_SIMPLE” method which means that only the end points of a contour are stored. For example if part of a contour is a straight line then two points will be stored (or four points for a rectangle contour). The outcome of the function is an array which contains all the end points (x and y pixel coordinates) for each detected contour. Then we obtain the area for each contour by using the "cv2.contourArea()" function and we select the one which is the largest, suspecting that this should be our desired area of interest according to our ECG scans.

Thereafter, we create a minimum rectangle that contains that previous area by using the "cv2.boundingRect(largest area)" function which finds and produces four

numerical values. The first two (x, y) are the coordinates of the pixel which is located at the top left corner of the rectangle. The third one (w) is the width and the fourth one (h) is the height of the rectangle. With those four values we know basically how to slice the array of our initial image and create a new image as we see in fig 3.2 which is already cropped and contains the desired region of interest (ROI = original image[y:y+h, x:x+w]).

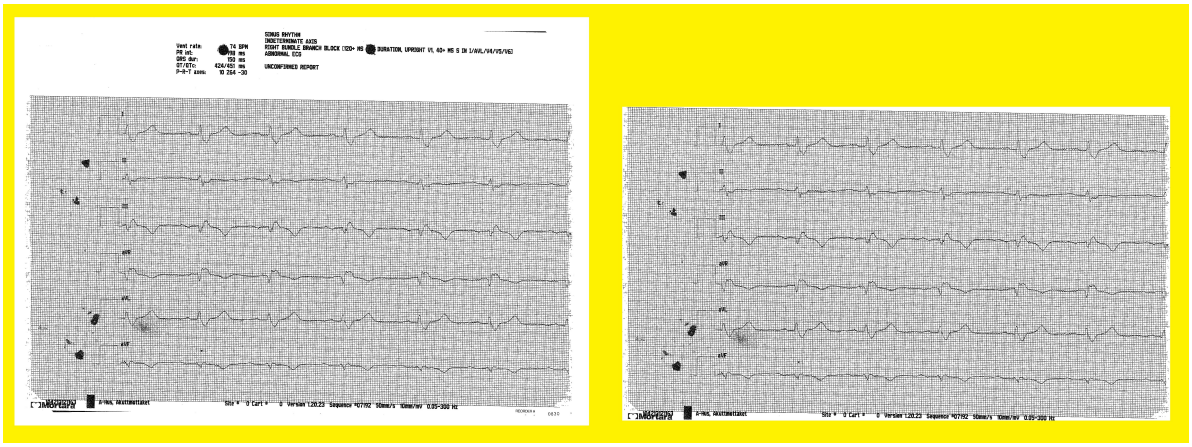


Figure 3.2: Step 1: extracting the area of interest. Left: original image. Right: cropped image.

3.2.2 Step 2: the skewness correction

The second step of the digitization process is to correct a possible skewness of the image which was caused by an improper scanning. As we have mentioned before, even a small skewness can affect the outcome of the digitization. The idea here is to detect straight lines in the picture and calculate their angles with the horizontal line which is used as a reference. In our case many straight lines from the grid can be detected and we can easily find the skew angle of the picture as it will be described below.

The image that is about to be read by the algorithm, is the original cropped image from the previous step. Thus, we need to apply again the proper filters to prepare it for the second step. First, we convert it into a grayscale image and then we apply the Canny edge detector like we did previously. The Probabilistic Hough Line Transform technique is the one that will help us to detect the straight lines under the limitations that we set. The related Python function is "cv2.HoughLinesP()". The technique finds for every point of the image with specific coordinates, all the possible lines that are satisfied by that point. The lines are not represented by the two typical parameters of slope and interception. Instead, they are represented by the r distance and the θ angle

in the polar system. Now the algorithm knows which points belong to the same line as well as its equation. It is then up to us to decide what is the minimum number of points to have in the detected lines (the fewer points, the more detected random lines). This threshold is passed as an argument (minLineLength) inside the function and for our case that value is 100 which is reliable enough to detect the "horizontal" straight lines of the grid. Other arguments are the "maxLineGap" which defines the maximum gap between two points to be considered for belonging to the same line (5 in our case), the r distance resolution in pixels (5) and the angle θ resolution in radians (math.pi / 180.0). The outcome of the function produces the pairs of the endpoints $((x1, y1), (x2, y2))$ of every straight line detected.

Using the above endpoints we calculate the arc tangent in radians for every line (math.atan2$(y2 - y1, x2 - x1)$) and then we convert them in angle degrees (math.degrees()) which are stored in a list where we select their median value. Finally that median value is considered the skew angle and the next step is to rotate the image by that angle. For that, we apply the function "ndimage.rotate()" where we insert as arguments the original cropped image and the skew angle. The results are illustrated in figure 3.3.

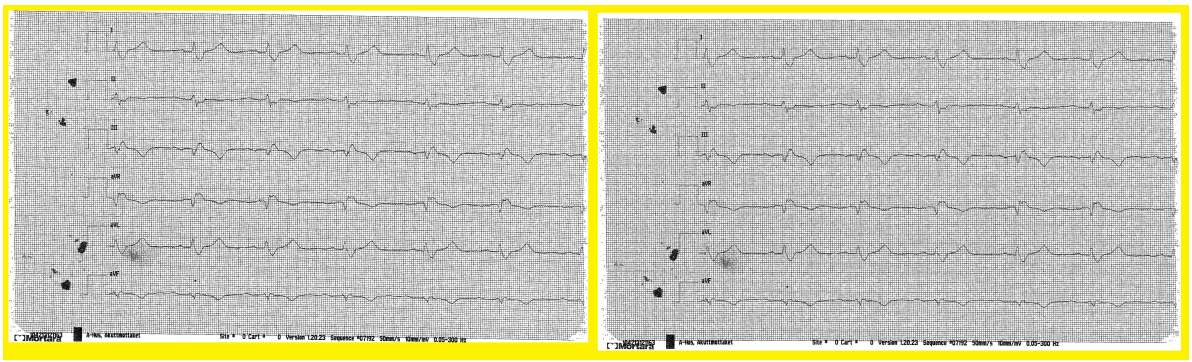


Figure 3.3: Step 2: correcting the skew angle. Left: cropped image before the rotation. Right: cropped image after the rotation.

3.2.3 Step 3: erosion

After we have successfully rotated the image which is still part of the unprocessed original image, we now start to apply a number of new filters and techniques for the separation between the ECG signals and their background. The first one is called erosion which is basically one of the two main morphological operations (dilation is the other one) and usually is applied in binary images. Erosion emphasizes the black foreground objects by transforming the white pixels that are next to their boundaries into black (white pixels are reduced). The operation is carried out by a kernel which

can be custom made and runs through every pixel of the image. In our case the kernel is a 2x2 matrix filled with 1s and checks for every white pixel in the image, if all their neighbors based on the kernel's size are also white. If not, then the targeted pixel becomes black. The utilized Python function is the "cv2.erode()" and as basic arguments we pass the kernel and the number of iterations that the kernel will be operated. The outcome of the erosion is an image with thicker black elements (like the signal and the grid) than they were before (figure 3.4).

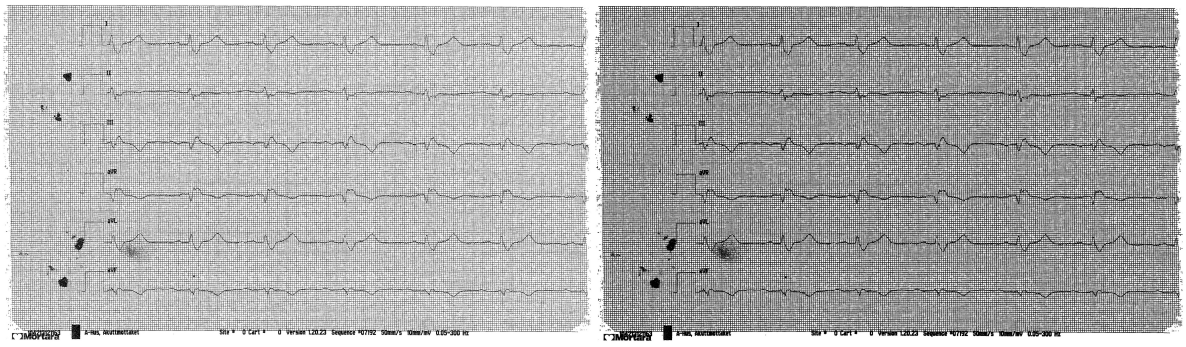


Figure 3.4: Step 3: erosion. Left: image before the erosion. Right: image after the erosion.

3.2.4 Step 4: blurring

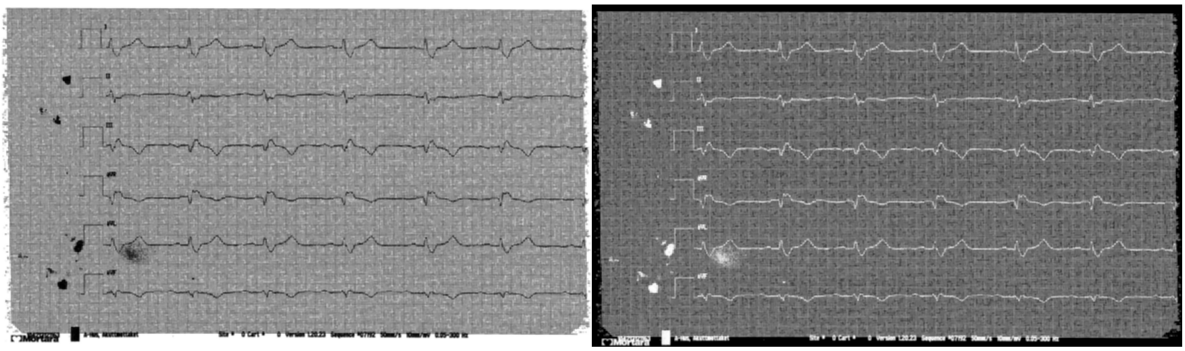


Figure 3.5: Step 4: blurring. Left: blurred image. Right: inverted image.

The filter that we implement next, is a median filter so that we can blur the image and make the background removal easier. Here we select a 9x9 kernel which replaces every pixel value with the average value of itself and all its neighbors that are inside that kernel, calculated with equal weights. Every element of the kernel has equal

value of 1/81. The related function for applying the filter is the "cv2.filter2D()" and the outcome is shown in the left part of figure 3.5. Afterwards we simply invert all the bits for every pixel value in the image by using the "cv2.bitwise_not()" function. Assuming that a 8 bit pixel value can be 10110010, the function will change it to 01001101. Of course that way, the black pixels become white and vice versa. We can see the inverted image in the right part of figure 3.5.

3.2.5 Step 5: masking

With the next step we now start to separate the signals from the grid having the inverted image from the previous step as an input. The utilized function is the "cv2.inRange()" which detects and masks pixels in the image with a specific range of colors given by the user. The idea here is to detect ideally all the background pixels that are not related to the signals and mask them. The masking process basically binarizes the image by turning into white all the detected pixels inside the given color range and by turning into black the rest of the pixels. The color range is defined by a lower and an upper set of pixel values which correspond to two different colors. Our input image is gray scaled, so we talk about dark gray and light gray colors and in our case, we set the lower set of pixel values as (0, 0, 0) and the upper set as (130, 150, 150). The outcome of the operation shows that the signals are clearer than before but still there is some remaining noise in the background (left part of figure 3.6). Next, we invert the outcome image with "cv2.bitwise_not()" like we did previously and the resulting image is illustrated in the right part of figure 3.6.

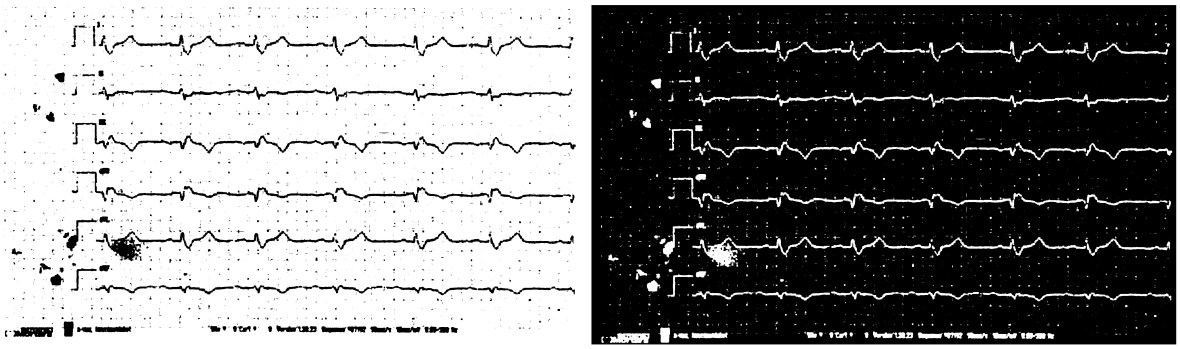


Figure 3.6: Step 5: masking. Left: masked image. Right: inverted image.

3.2.6 Step 6: the CCL technique

The remaining noise needs to be addressed, so our next move is to apply some techniques to remove it as much as possible. First, we crop the image perimetrically

by removing 50 pixels from all its four sides. This results to take out the characters which were located at the bottom of the image (left part of figure 3.7). Second, we implement a technique which is called Connected Components Labeling (CCL) and aims to categorize regions of pixels with the same kind of connectivity. This means that the algorithm identifies and groups the areas in the image where the pixels are neighboring, or they have similar intensity values. That way, it is possible to isolate and mask areas which contain some of the random noise in the background of the image. Since the sizes of the obtained areas vary, an important parameter for this technique is the definition of a threshold value which corresponds to a maximum size of the masked areas which are going to be removed from the background. This value should be big enough to keep the signals untouched and small enough to remove as much background noise as possible. In our case the threshold value (area size) is set to 90000. The outcome of the above process is illustrated in the right part of figure 3.7 where we can see that most part of the spread noise is removed except from some annoying spots in the beginning of the signals. In addition the process has also reversed the black and white pixels resulting a white background.

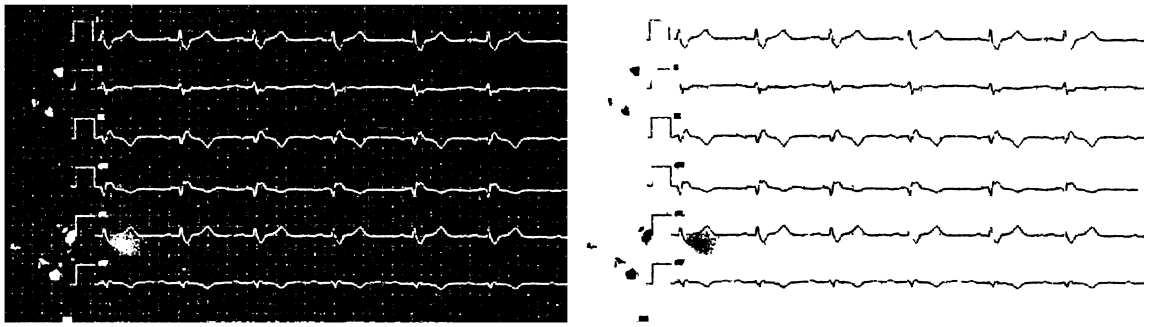


Figure 3.7: Step 6: the CCL technique. Left: perimetrically cropped image. Right: noise removal after CCL technique.

3.2.7 Step 7: extracting the individual coordinates

The previous step is the last one in a sequence of steps where image processing techniques were applied for separating the signals from their background in an ECG scan. Now the only thing we need to do is to complete the digitization of the signals by finding their coordinates and then store them for each signal separately, so that they can be plotted any time and individually in another platform for every possible use like evaluation of their accuracy or training a machine learning model.

Based on the pixel values of the image (0 or 1), we can find the coordinates of the pixels that only belong to the signals (and unfortunately to some of the left over noise

that comes with them). For the moment our processed binary image has black color signals and white background (right part of figure 3.7). We need to invert it one more time so that the signals become white, and the background becomes black because we ask from the related function (coordinates = np.where(img != [0])) to extract the coordinates of those pixels that are not black. The result is to obtain all the x and y pixel coordinates for the six signals. Here we should mention that the values of the y coordinates start from the left top corner of the image (value 0) and increase while we move downwards, and this happens because our image is basically a 2D array where the row indexing starts from above. What we really need instead, is that the y axis should be reversed in order to have the correct y values for plotting. Therefore, before we applied the last function above, we had already flipped our image (cv2.flip()).

The extracted pixel coordinates are now scatter plotted for all signals and the plot is illustrated in the left part of figure 3.8 where we can also see the x and y axis. The coordinates in the x axis reach until around 2500 and the ones in the y axis until around 1400 (all integers). This is something that we expected since the size of the image is 1459x2594 pixels. From the plot we check the y axis in order to detect which are the 5 threshold values that we can consider, for splitting safely the six signals. We then create six individual lists of coordinates based on those y threshold values, and we plot them separately. We can see the results in the right part of figure 3.8.

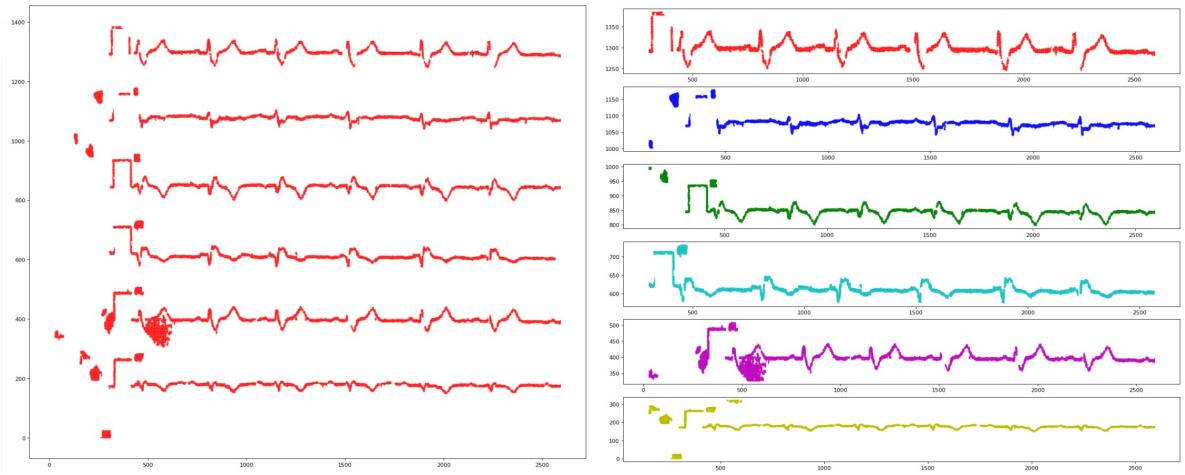


Figure 3.8: Step 7: extracting the coordinates. Left: one scatter plot for all signals from the extracted coordinates. Right: individual scatter plots of the six signals with new sets of coordinates.

The size of each list of coordinates for each one of the six signals varies from 28000 until 40000 which means that we have thousands pairs of x and y values, and that is because there are many pixels with the same x value (pixels of the signal on top of each other for instance). Our new digitized signals need to have coordinates with

unique x values in order to be presented as nice thin graphs and not as scatter plots. Therefore, for the coordinates with the same x value we calculate the mean value of their corresponding y values and we replace them with only one pair of coordinates (common x value and mean y value). The result is the creation of a new list of coordinates for each signal with obviously a much smaller size (around 2100). We then make the graphs of the six signals based on the new coordinates and we can compare them with the original signals in the scanned image (figure 3.9).

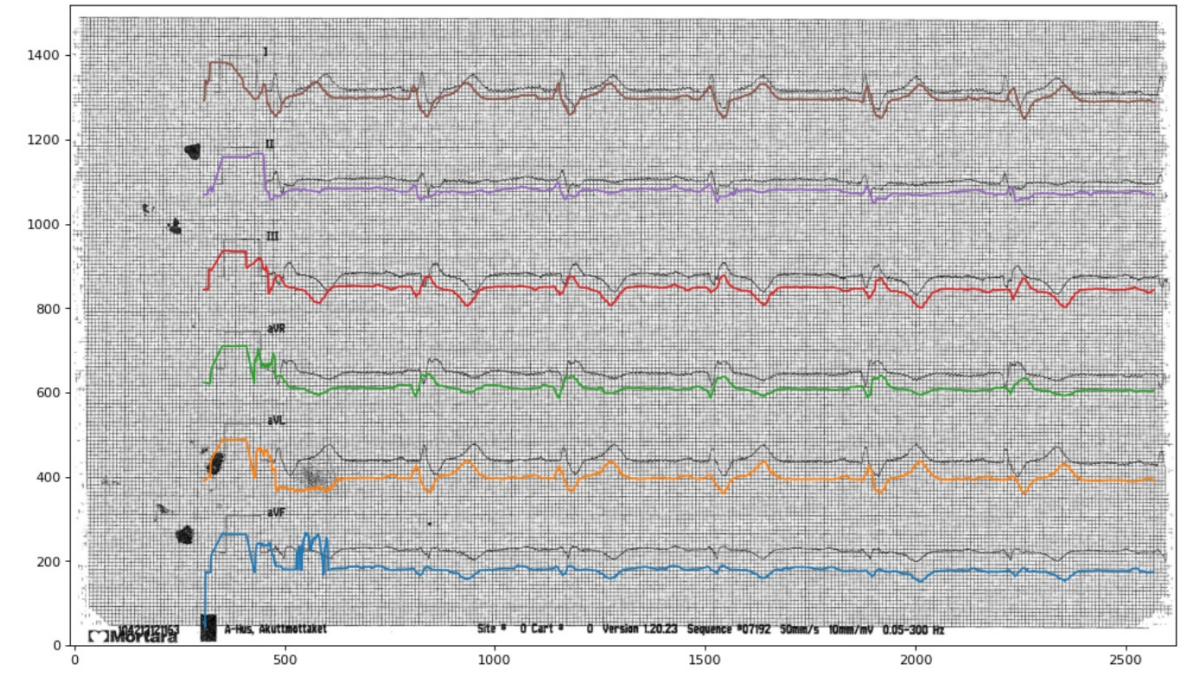


Figure 3.9: Comparison between the original signals in the scan and the new digitized colored signals.

Chapter 4

Critical insights towards a more automated tool

The presented tool has obviously many parameters involved, and this is something that we can expect for digitizing an ECG signal from a scan image. One can introduce different scans to the digitization pipeline and by making every time small or big corrections to the various parameters, the desired outcome can be achieved. However, by this fact we can understand that the tool cannot be fully automated which means that it is impossible to efficiently digitize ECG scans without a human intervention.

In this chapter we will try to identify gaps throughout the various steps of the process, that prevent the tool for being fully automated. First of all, let us mention that for almost every step there are some parameters involved inside or outside the different Python functions, that can be adjustable. Some of them are valid for every scan of our dataset after the testing we have made, and some others need to be tuned manually depending on the scan. Even for those parameters that are the same for our dataset, does not mean that they remain the same for another dataset with different characteristics.

4.1 The preparation steps

In the very first step the tool introduces the first parameter which is the percentage of the image's resizing. Its correct value seems to be crucial for the process which concerns the detection of the area of interest and therefore for the proper cropping of the image. A different resizing value gives very unreliable results. In our case the current value that the tool recommends (40%), works almost for our whole dataset but there are three scans which produce different results if we don't change that value. For instance the scan number 4 is cropped in a way that we lose the first signal (left image of figure 4.1) and its correct resizing value is 30%. In addition, the scan number 6 is

cropped way above the borders where the grid starts (right image of figure 4.1) and its correct resizing value is 42,5%. A common resizing value for all the scans could not be found since a correct value for one scan could be inaccurate for another. We also tried to tune the other two parameters which are related to the area of interest detection (i.e. blurring and canny edge detector) in order to check if we can have desired results by keeping the resizing value constant as the tool suggests. It succeeded only with one of the three scans.

Figure 4.1: Inaccurate scan cropping with the tool's recommended resizing value. Left: scan number 4. Right: scan number 6.

The next steps which include the finding of the skew angle and finally the rotation of the image can be applied for every scan of our dataset, without changing any values of the involved parameters. Let us remind here that all the steps so far as we have explained, do not have to do directly with the signal extraction rather than the preparation before that.

4.2 The erosion and blurring steps

The forthcoming three out of the four steps which are the most important and concern the separation between the signals and their backgrounds are also not automated. These four steps which have been described in detail previously implement erosion, blurring, masking and CCL techniques respectively. Erosion utilizes a 2x2 kernel which highlights the signal but also the grid in the background (it makes them thicker). The size of the kernel works fine for every scan, but we were wondering whether a different size can have any positive impact on the final results. Afterall, we would like to try a kernel whose dimensions are odd numbers, so that it can be applied in a central pixel. When we applied a 3x3 kernel for the erosion, the outcome image had thicker gridlines and the background overall became darker. It was then very difficult

to separate the signal from the grid with the next filters applied by the tool. Obviously, a bigger kernel would produce an even darker image. In that case we consider the 2x2 kernel a fixed parameter for our tool that do not block its automated function.

The blurring technique which comes after erosion, relies on a 9x9 kernel which operates on every pixel as we have explained previously. Just keeping that kernel size constant for every scan, seems like we get the best possible results at the end of the digitization process. If we try slightly different values for the kernel size and then adjust accordingly the parameters of the next two steps, we get similar image results. However, we cannot tell for sure with our bare eyes if those results are better than the ones with the original 9x9 kernel size. Therefore, we need to investigate further whether this parameter can be adjustable or not for different scans and that's why we will consider it as a parameter that prevents the tool for being automated.

4.3 The masking step

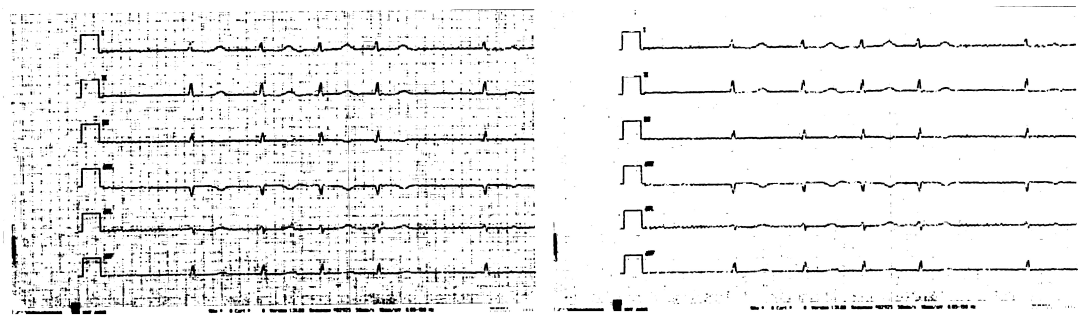


Figure 4.2: Comparison between two different upper threshold values sets from the masking process for the same scan. Left: proposed value set by the tool (130,150,150). Right: alternative value set (145,150,150).

The next step where we apply the masking process is clearly an obstacle for the automated digitization. Specifically, the first value (130) that the tool recommends for the upper threshold set of values in "cv2.inRange()" function does not apply for every scan. This threshold value is very critical for removing the background noise of the signal and the more precise it is, the less noise we get in the outcome image. In figure 4.2 we can see the outcomes of masking when we used different threshold values for the same scan. In the left image which corresponds to the suggested threshold value by the tool, there is plenty of remaining noise. In the right image however, we have much less noise with an alternative threshold value. Needless to say, when we proceeded with the left image throughout the whole digitization process, the final digitized signals were unreliable. We then conclude that in order to execute this step

and have satisfying results, we need to manually change the corresponding parameter depending on the scan.

4.4 The CCL technique step

Manual intervention is also required in the CCL technique which somehow cleans the remaining noise from the previous step of masking. The parameter here is a threshold value which indicates the maximum area size of the connected components (group of connected pixels) that will be removed. The tool recommends the value 90000 as the maximum area size value, meaning that all the areas which belong to noise or signals and are smaller than this size will be removed. We observed that for various scans of our dataset, the recommended threshold value (90000) is not valid even if those scans have the same recommended parameters from the previous steps. As it is illustrated in figure 4.3, the applied recommended value for a specific scan removes part of the signal together with some noise (left image). A smaller value instead looks more appropriate as it keeps most parts of the signal untouched (right image). Here it is important to mention that the threshold value for this step should be selected carefully, because if we set it very high, we remove a lot of noise, but we also lose information from the signals. We can observe for instance some gaps inside the signals or some of their peaks being cropped away. If the threshold value is set low, then we keep much of the noise that will negatively affect the signal digitization. One last thing that we need to report about this step, is that the tool crops 50 pixels around the image before the implementation of the CCL. This action is helpful for the removal of some extra noise and it works for every scan of our dataset, but most likely wouldn't be efficient or necessary for another dataset, due to its different topology.

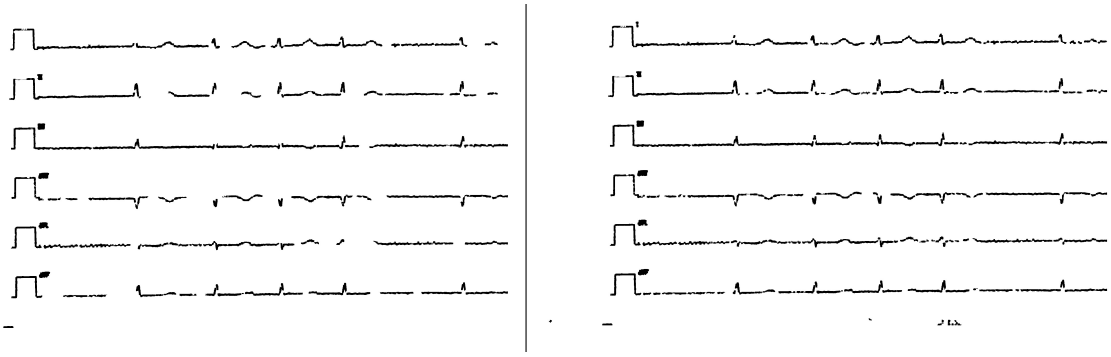


Figure 4.3: Comparison between two different area size values from the CCL technique for the same scan. Left: proposed value by the tool (90000). Right: alternative value (20000).

4.5 The signals' split step

The last challenge that indicates problems in the automated signal digitization procedure is the default threshold values in the y axis that the tool suggests for splitting the six signals for the creation of six different sets of coordinates based on those maximum y values. For the most scans of our dataset the signals are not close to each other and there is enough space between them. Therefore, those standard values in the y axis are quite safe to be selected for separating the six signals. However, there are some scans like the one in the left part of figure 4.4, whose signals are very close to each other or almost touch each other (the original scan can be also seen in the left part of figure 5.4 of the next chapter, where we discuss possible solutions). This can happen when the signals have high negative or positive peaks. In that case the above threshold values are not able to split the signals correctly. In the right part of figure 4.4 we can see how the six signals are split with the suggested y threshold values. The six signals are scatter plotted with their new individual coordinates and the problem that appears is obvious in almost every signal. For instance, the negative peaks of the first signal are cropped and the missing parts have been transferred down in the second plot.

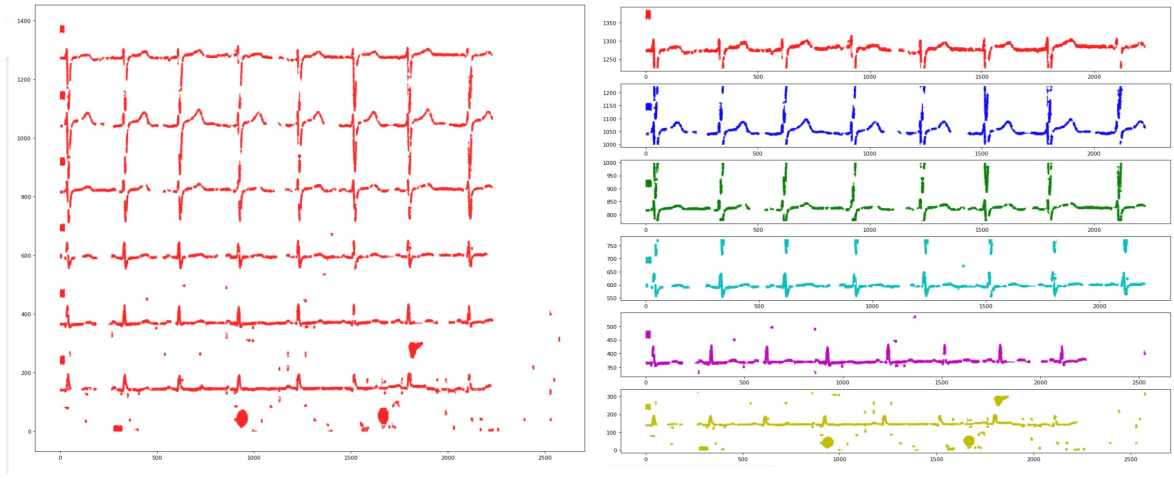


Figure 4.4: Challenge for splitting the signals. Left: common scatter plot before the split. Right: problematic scatter plots for each signal after their split.

4.6 Summarizing the challenges

In this chapter we have detected and described the part of the digitization steps that prevent our tool to be fully automated for the dataset we have at our disposal. We

have also reported other parameters that possibly are not universal for other datasets and therefore need to be tuned manually. Summarizing, there are five main challenges towards automation for our dataset. The first one is related to the non-automated cropping procedure of the scans and we will address it in the next chapter. The next three challenges concern the parameters in the steps related to the separation of the signal from its background. Those are the size of the blurring kernel, the threshold value for the masking technique and the threshold value for the CCL technique. The fifth challenge lastly that we will also try to approach in the next chapter is the invalid set of the fixed y axis values that are required to split the signals from each other.

Chapter 5

Solution approaches towards an improved tool

5.1 The cropping challenge

Cropping the scan as the first step of the signal digitization is essential because we can obtain the part where the grid is located and eventually focus on how to extract the signals from it. It involves three adjustable parameters and as we explained, at least one of them which is the resizing value of the image, causes problems to the automation of the current tool. Here we try to approach that challenge with a completely different perspective that ignores all these three parameters and make this process more independent. Basically, we will introduce an alternative method which is based on the horizontal and vertical projection of the image's pixels for finding the boundaries of the grid. We will take advantage of the fact that inside the grid, the density of the dark pixels is higher than the one outside of the grid. Indeed, if we take a look at a scan (figure 3.1), we will observe that the grid is surrounded by a white area which contains characters and some noise.

All the scans that we have at our disposal are binary which means that they have black and white pixels, and this will make the recommended method easier. The scan is not recognized as binary after we load it for processing, so we need first to transform it into grayscale and then into binary by applying a threshold value. Here any threshold value will do because the image is already binary as we said (a typical value that is often used is 127). Next, we convert all black pixels from value 0 to value 1 and all the white pixels from value 255 to value 0. The result now is a 2x2 array with 0s and 1s, and dimensions the height and the width of the image.

We then add all the pixel values for each row of the array and we put the results in a list whose length is the number of the rows or equally the height of the image. Since every black pixel has value 1 and every white pixel has value 0, the above list actually

contains the number of black pixels per pixel row. That way we managed to project horizontally all the black pixels in a conceivable vertical axis. It is very interesting to plot that list for a couple of scans and make some valuable observations which will help us to follow a proper strategy for correctly crop each scan.

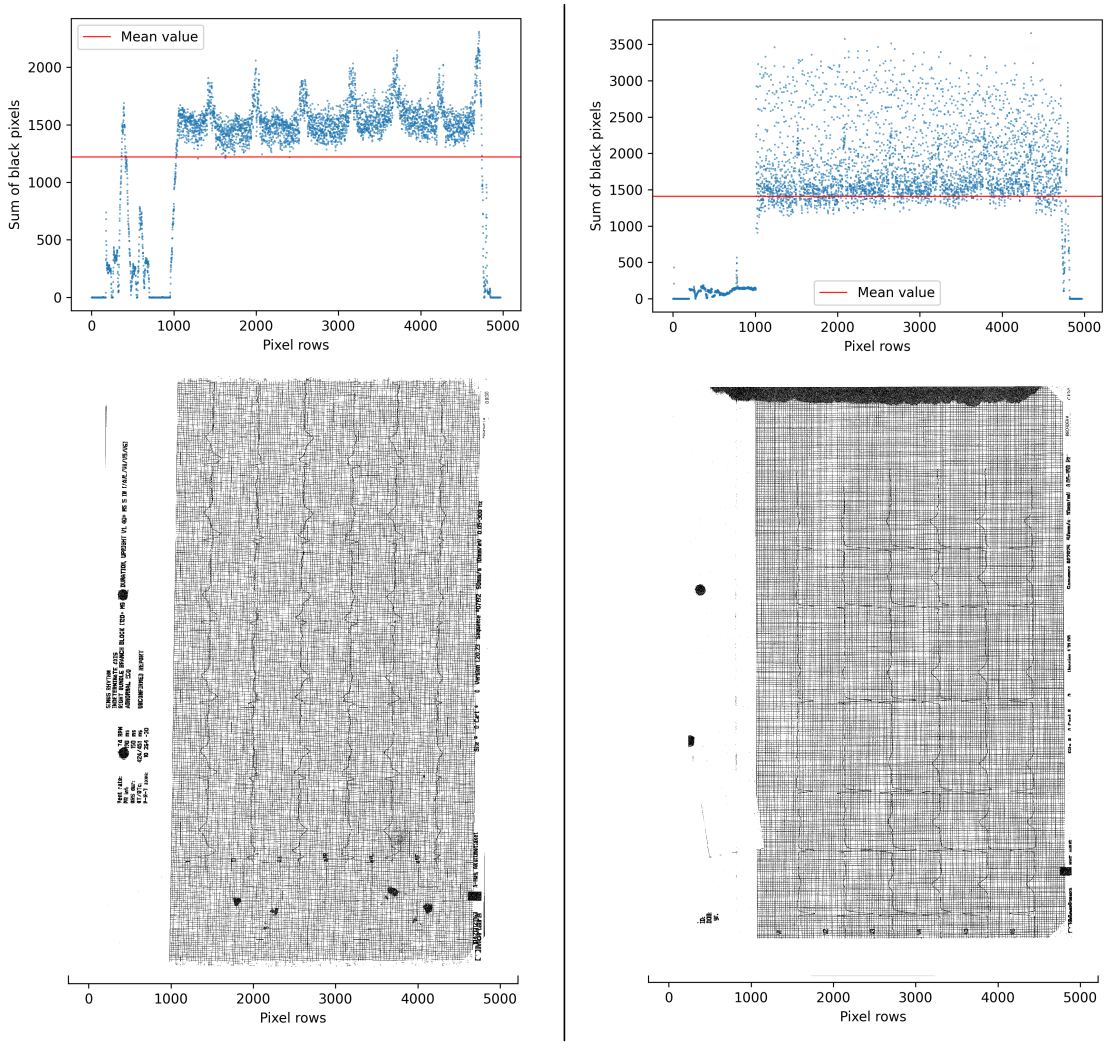


Figure 5.1: Plots of horizontal projections and their corresponding scans rotated by 90 degrees.

In figure 5.1 there are two of these plots and in order to interpret them better, we have attached below them the actual corresponding scan rotated by 90 degrees. In both plots we can obviously see the horizontal boundaries of the grid which are approximately from pixel row value 1000 to pixel row value 4800. We can confirm this by observing the scans below the plots as well. It is also interesting to see how the

different elements of the scans are marked on the plots. In the first plot on the left for example we can see how the words or characters which are located above the grid, create those peaks approximately between pixel row values 250 and 700. The same applies for the six baselines of the signals inside the grid area. Their footprints are well shown in the plot. Another interesting thing is that the plot can show us the areas where there are not any black pixels (white areas in the scan). Especially in the first plot we can see three of the biggest white areas represented by three distinct horizontal lines at the bottom (0 values in the y axis). Moreover, we can notice that between the horizontal boundaries of the grid (1000 – 4800), it is impossible to find any pixel row value that corresponds to zero value in the y axis of the plot and this is something that we need to take advantage.

Therefore, we create a new list (non-zero list) where we place all those pixel row values that correspond to non-zero values in the plot. Then we group in smaller lists all the elements of the non-zero list that are consecutive numbers (e.g., 260,261,262,263...). The group (or the list) which has the biggest length is the one that we are looking for. In addition, the first value and the last value of that group should be the two pixel row values where we have to crop the scan horizontally. The above method works very well for the left scan of figure 5.1, but it is not enough for the right scan because of this black noisy strip which affects the finding of the grid boundaries. In that case the longest group that we discussed above starts approximately from the pixel row value 400 and not from 1000 as we can see both in the plot and the scan, so the cropping would not be accurate. We need then to consider one more limitation to solve that problem.

Thus, we create another list (above-mean list) where we place all those pixel row values that correspond to a value in the y axis of the plot which is bigger than the mean. The mean value is shown with a red horizontal line in the plots. With this way we can reject the unwanted pixel row values that don't belong inside the borders of the grid. The last step is to find all the common row pixel values between the above-mean list and the longest group list and put them in a common list. Like before, the first and the last value of the common list will give us the two pixel row values where we have to crop the scan horizontally. The creation of the common list is for avoiding the scenario where we have row pixel values that don't belong to the grid but they correspond to values in the y axis bigger than the mean as it happens in the left plot of figure 5.1.

The two limitations that we consider and are related to the non-zero values and the bigger than the mean values in the y axis, are enough to find the horizontal boundaries of the grid for every combination of challenges in the scan. The above procedure is applied again for finding the vertical boundaries of the grid. The only thing that changes this time is that instead of having a horizontal projection, we apply a vertical projection. This means that from the initial 2x2 array of 0s and 1s, we now add all the pixel values for each column and then create the corresponding list. At the end

of the process, we get the two pixel column values where we have to crop the scan vertically. In figure 5.2 we can see the plots of the vertical projection together with their corresponding scans. The final cropped scans based on the horizontal and vertical projection method are shown in figure 5.3.

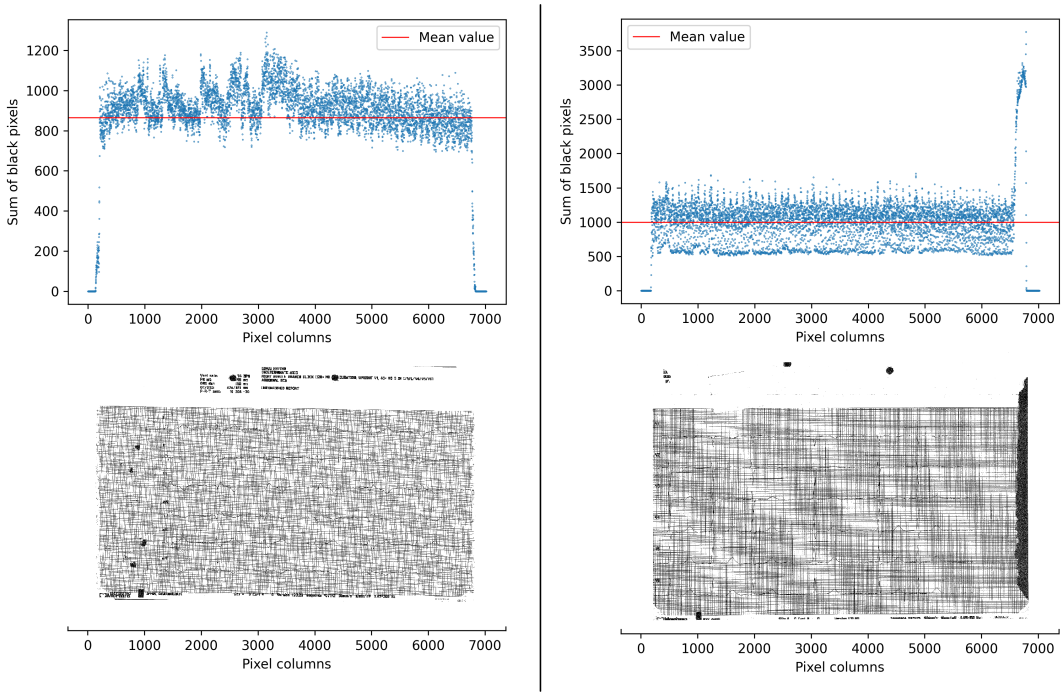


Figure 5.2: Plots of vertical projections and their corresponding scans.

Figure 5.3: Final cropped scans after the projection method.

5.2 The signals' split challenge

As we discussed in the previous chapter, the constant threshold values in the y axis for splitting the signals, are not the same for every scan of our dataset. In addition it is very difficult to predict exactly what is the value in the y axis where the one signal ends and the other starts, in case that the signals are very close to each other (left image of figure 5.4). For approaching this challenge, we need to think differently and find a way to split the signals which does not depend on how close they are.

An idea is to divide the original cropped scan (with the six signals) into six overlapping strips or rectangles which contain the whole corresponding signal. The strips should be wide enough so that a signal with all possible sizes can fit inside. A signal can have high positive or negative peaks so the strips should be big enough both over and under the baseline of the signal. Of course this means that in each strip most likely some parts from the neighboring signals may intrude from the top or from the bottom as it is shown in the right image of figure 5.4. These unwanted parts can be considered as a noise for the main signal and they need to be removed by implementing the noise removal steps of the tool that we have described. If the noise removal is successful, we will ideally have the main signal untouched and then we can extract its coordinates, achieving overall the split of the six signals in an automated way.

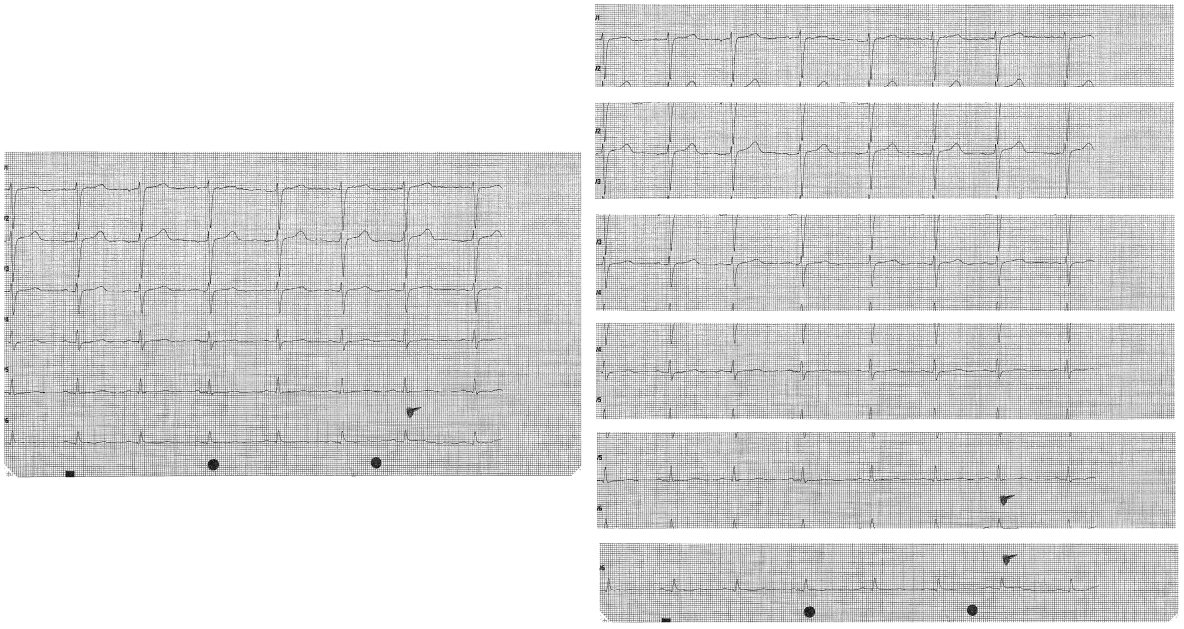


Figure 5.4: Left: Scan with the signals very close to each other, Right: Division of the scan into six overlapping strips.

Figure 5.5: Effect of the different area size values of the CCL technique. For each screenshot these values increase from the top to the bottom. Left: first strip. Right: second strip.

When we tested the above approach things were not as expected. For each strip, we applied the recommended filters by the tool. The CCL technique was the one which we were relied on the most, for removing the undesirable parts from the neighboring signals. This is because the sizes of the individual parts supposed to be smaller than the size of our whole main signal and with the proper value of the threshold parameter (maximum area size) we could have easily removed them from the image. The problem however was that the main signal had some cuts or gaps that occurred from the previous necessary applied techniques and the result was that it was split in smaller pieces. Therefore, it was no longer a big whole unit. As we were increasing the maximum area size value to remove the noisy parts, we were simultaneously losing parts of our main signal. A satisfying threshold value that would remove all the noise could also remove even the half part of the main signal. In figure 5.5 it is demonstrated what we have discussed above. We can see for the first two strips of figure 5.4 what is left of the two main signals when we apply different threshold values of the CCL technique. For each screenshot, these values increase from the top to the bottom. In the last screenshot of the first strip, the noise is totally removed with some loss of the signal while in the second strip the sizes of the noisy parts (neighboring signals) are comparable to the sizes of the pieces of the main signal and they cannot be removed even for a high threshold value.

Chapter 6

Discussion and conclusions

In this master thesis we have presented an available tool which can digitize ECG signals recorded on paper. It takes as an input the scanned paper as a digital image and then extracts the ECG signals from it with their coordinates. We can divide the digitization process into three main stages. The first one is the preparation stage and concerns the detection of the grid, the cropping of the image according to the grid's boundaries and the correction of the skew angle of the image. The second stage concerns the implementation of several filters and techniques which remove the background grid and possible noise. Finally, the third stage separates the signals from each other and finds their coordinates. The tool was tested for every scan of our dataset and for every stage of the process in order to evaluate its ability for being fully automated. In every stage we detected some parameters that did not apply for every scan and instead we had to manually give them the proper values which were depended on the scan.

The first challenge appeared in the very beginning where the first parameter for the image resizing was not common for the whole dataset, resulting failures in the image cropping. We introduced an alternative method for finding the boundaries of the grid and then cropping the image accordingly. The method is based on the horizontal and vertical projection of the black pixels of the scans and it works successfully for the whole dataset. Replacing the cropping method that the tool suggests, with the one that we suggested, we achieved to eliminate three parameters related to the detection of the area of interest, which increase the risk of a non-automated digitization process. Moreover, for multiple scans we can now automatically obtain their grid areas with the ECG signals and then crop them successfully. The improvement that we made in this first step of the ECG digitization can also have another benefit. This digitization step can be independent and operate separately. For instance, it could be used by Akershus University Hospital only for cropping a huge number of scans and removing the additional info from them. It can also be offered as the first digitization step for another more optimized tool that could be developed in the future.

Here we need to make two more comments about our approach in the first challenge. The first comment is that looking at the results in figure 5.3 after our cropping method, we will observe that at the bottom of the images there are some white thin strips which contain some noise and do not belong to the grid. This means that the horizontal projection could not deal with that issue and more limitations should have been applied. However, those strips appear anyway in the cropping method that the tool recommends and for solving that problem it just crops away 50 pixels around the image at a later step. The same can be done in our case as well.

The second comment is that after we are finished with our cropping method, we need to resize all cropped images to 40% again so that the applied filters of the tool that comes after cropping, can properly work. The first thing that the tool does is to resize every scan and it seems that some of the filters that are applied afterwards work for that specific resolution. When we applied our method, we skipped that parameter and we then realized that if we keep the full resolution of the scan and move on with it during the digitization process, we get unreliable results. This fact leads us to an important conclusion. Some of the parameters' values of the applied filters depend on the resolution of the initial scan. Indeed, if we assume that we apply a 5x5 blurring filter for instance on every pixel of a specific image, the results would have been different if that image had a higher resolution (meaning more pixels).

In the second stage, a number of challenges for the automated digitization process appeared as well, where we detected three parameters that require manual tuning (kernel size for blurring, threshold value for masking and threshold value for CCL technique). Those parameters are responsible for separating the background grid from all the signals of the scan. We believe that for every scan, there is a proper combination of values for these three parameters that can give us the optimum result which means the clearest signals as possible with the less noise as possible. Despite that we have not approached this challenge in this thesis, we can provide some ideas for future work. The finding of the above values can be considered as an optimization problem. We can apply algorithms like gradient descent for searching those parameters' values which can satisfy a condition (or more) related to a clear and less noisy signal. Such a condition which can be expressed mathematically could be the following. As we mentioned in section 3.2.7 and in the part where we discussed the pixel coordinates extraction, we calculated the mean y value for all the pixel coordinates that had the same x value in order to create unique (x, y) coordinates for each signal. Here it would be meaningful to calculate the standard deviation for each of these mean values as well. When there are unwanted noisy pixels around the signal, they are unfortunately integrated inside the calculation of the y mean values, making their standard deviations bigger. Thus, a possible condition (between others) for our optimization problem could be the minimum value of those standard deviations or the minimum value of their average. In this way, it is only the algorithm that could

find the correct values of the related parameters as an optimum solution for the current digitization stage and therefore it could make that stage automated without any human factor stepping in.

In the third stage of the digitization pipeline, we observed another automation gap concerning the separation of the six signals in the scan according to some fixed values on the y axis. These values are supposed to act as upper and lower boundaries for the horizontal separation of the signals in case they have a safe distance away from each other. Since this condition does not occur always and the signals can also be quite close to each other, this set of values can be problematic and inaccurate. We proposed to make those boundaries even wider in order to include the main signal regardless of its size, in addition with some unwanted parts from neighboring signals, which were supposed to be removed later. Those noisy parts however were not successfully removed with the additional noise removal techniques that we applied because the size of the main signal was comparable with their sizes.

The above method that we suggested can partly solve the issue in the automated process in this stage, but the current filters of the tool are not enough to complete it. As we have mentioned, the main signal looks small due to its multiple cuts that inherited from before. We need to find a way to fill those gaps so that the signal becomes one piece with a considerable size. This could be probably achieved using signal interpolation techniques which need to be applied to all six signals before we proceed to our proposed method. This way the size of each signal could become bigger than almost all the remaining noise areas around it. The CCL technique would then easily remove the left over small noisy parts. Here it is important to mention that even if we managed to remove the undesirable neighboring signals after the interpolation, we could not have dealt with those signals that actually touch each other. The separation of overlapping signals remains a big challenge and as far as we know, nobody in the literature has managed to present a clear and solid solution.

As we stated above, in this thesis we assessed the capacity of the presented tool for being fully automated. However, we did not evaluate the accuracy of its digitization results. The extracted signals from the digitization process can be compared to the corresponding ECG signals on the scan only visually. If we had for those scans the digitized signals which were recorded straight from the ECG machine in a digital format, then we could have easily compared our results quantitatively and evaluated the accuracy of the digitization process. Nevertheless, such kind of dataset (both ECG scans and their digitized signals) is being prepared by Akershus University Hospital for further collaboration with NordSTAR in the near future.

Another important thing is that the assessment of the ECG digitization tool in this thesis, was held according to a specific dataset that we had at our disposal. After all, the whole tool was designed in a way that can work more efficient for scans like the ones we were provided by Akershus University Hospital with their own particularities and

topology. It cannot deal with a 3x4 ECG scan for example and separate the different lead signals that are in the same row. Our scans have one signal per row. However, it tries to cope with other challenging tasks like the digitization of binary ECG scans. In most of the related work about the digitization of ECG scans, the images that are handled are colored and that makes the separation between the signals and their background grid, an easier case.

Despite the limitations that our tool has in the different ECG formats, it would be a great achievement if we could further develop and improve it in a way that it could automatically digitize at least the enormous number of ECG scans (approx. six million) from Akershus University Hospital. The digitized signals that can be produced by an improved digitization tool contain valuable information that can contribute significantly to further medical research. All these available ECG scans that we have at our disposal are labeled, which means that for every digitized signal, we know in which heart irregularity it corresponds to. Machine or deep learning algorithms can then be trained with these labeled data and find patterns which can help to predict early and accurately a potential dangerous cardiovascular disease by "reading" an ECG from a patient. However, the way towards a fully automated digitization tool is still quite long and the challenges are many. In this thesis we realized that despite the difficulties, we can have access to modern programming tools and techniques that allow us to make small steps to address the various challenges and finally make some progress for improving an ECG digitization tool.

Bibliography

[1] A. B. de Luna. 'Willem Einthoven and the ECG: Antoni Bayés de Luna discusses the discovery of the ECG almost 120 years ago which has remained almost unchanged to the present day'. In: *European Heart Journal* 40.41 (Nov. 2019), pp. 3381–3383. ISSN: 0195-668X. DOI: 10.1093/eurheartj/ehz721. eprint: https://academic.oup.com/eurheartj/article-pdf/40/41/3381/30338394/ehz721.pdf. URL: https://doi.org/10.1093/eurheartj/ehz721.

[2] G. Sannino and G. De Pietro. 'A deep learning approach for ECG-based heartbeat classification for arrhythmia detection'. In: *Future Generation Computer Systems* 86 (Apr. 2018). DOI: 10.1016/j.future.2018.03.057.

[3] F. Sufi and I. Khalil. 'Diagnosis of Cardiovascular Abnormalities From Compressed ECG: A Data Mining-Based Approach'. In: *IEEE Transactions on Information Technology in Biomedicine* 15.1 (2011), pp. 33–39. DOI: 10.1109/TITB.2010.2094197.

[4] L. Tereshchenko, N. Sotoodehnia, C. Sitlani, F. Ashar, M. Kabir, M. Biggs, M. Morley, J. Waks, E. Soliman, A. Buxton, T. Biering-Sørensen, S. Solomon, W. Post, T. Cappola, D. Siscovick and D. Arking. 'Genome-Wide associations of global electrical heterogeneity ECG phenotype: The ARIC (Atherosclerosis Risk in Communities) study and CHS (Cardiovascular Health Study)'. English (US). In: *Journal of the American Heart Association* 7.8 (Apr. 2018). ISSN: 2047-9980. DOI: 10.1161/JAHA.117.008160.

[5] S. Okutucu and A. Oto. 'Fundamentals of ECG'. In: *Interpreting ECGs in Clinical Practice*. Cham: Springer International Publishing, 2018, pp. 1–18. ISBN: 978-3-319-90557-0. DOI: 10.1007/978-3-319-90557-0_1. URL: https://doi.org/10.1007/978-3-319-90557-0_1.

[6] A. Holkeri, A. Eranti, T. V. Kenttä, K. Noponen, M. A. E. Haukilahti, T. Seppänen, M. J. Junttila, T. Kerola, H. Rissanen, M. Heliövaara, P. Knekt, A. L. Aro and H. V. Huikuri. 'Experiences in digitizing and digitally measuring a paper-based ECG archive'. In: *Journal of Electrocardiology* 51.1 (2018), pp. 74–81. ISSN: 0022-0736. DOI: https://doi.org/10.1016/j.jelectrocard.2017.09.007. URL: https://www.sciencedirect.com/science/article/pii/S0022073617303382.

[7] R. Patil and R. Karandikar. 'Robust algorithm for digitization of degraded electrocardiogram paper records'. In: *ICTACT Journal on Communication Technology* 8 (Sept. 2017), pp. 1604–1609. DOI: 10.21917/ijct.2017.0236.

[8] G. S. Waits and E. Z. Soliman. 'Digitizing paper electrocardiograms: Status and challenges'. In: *Journal of Electrocardiology* 50.1 (2017), pp. 123–130. ISSN: 0022-0736. DOI: https://doi.org/10.1016/j.jelectrocard.2016.09.007. URL: https://www.sciencedirect.com/science/article/pii/S0022073616301807.

[9] W. H. Organization. *Cardiovascular diseases (cvds)*. URL: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds) (visited on 20/01/2022).

[10] A. H. Association. *What is cardiovascular disease?* URL: https://www.heart.org/en/health-topics/consumer-healthcare/what-is-cardiovascular-disease (visited on 20/01/2022).

[11] M. Baydoun, L. Safatly, O. K. Abou Hassan, H. Ghaziri, A. El Hajj and H. Isma'eel. 'High Precision Digitization of Paper-Based ECG Records: A Step Toward Machine Learning'. In: *IEEE Journal of Translational Engineering in Health and Medicine* 7 (2019), pp. 1–8. DOI: 10.1109/JTEHM.2019.2949784.

[12] J. D. Fortune, N. E. Coppa, K. T. Haq, H. Patel and L. G. Tereshchenko. 'Digitizing ECG image: new fully automated method and open-source software code'. In: *medRxiv* (2021). DOI: 10.1101/2021.07.13.21260461. eprint: https://www.medrxiv.org/content/early/2021/07/16/2021.07.13.21260461.full.pdf. URL: https://www.medrxiv.org/content/early/2021/07/16/2021.07.13.21260461.

[13] S. H. Jambukia, V. K. Dabhi and H. B. Prajapati. 'Classification of ECG signals using machine learning techniques: A survey'. In: *2015 International Conference on Advances in Computer Engineering and Applications*. 2015, pp. 714–721. DOI: 10.1109/ICACEA.2015.7164783.

[14] U. Rajendra Acharya, P. Subbanna Bhat, S. Iyengar, A. Rao and S. Dua. 'Classification of heart rate data using artificial neural network and fuzzy equivalence relation'. In: *Pattern Recognition* 36.1 (2003), pp. 61–68. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(02)00063-8. URL: https://www.sciencedirect.com/science/article/pii/S0031320302000638.

[15] H. Bleijendaal, L. A. Ramos, R. R. Lopes, T. E. Verstraelen, S. W. Baalman, M. D. Oudkerk Pool, F. V. Tjong, F. M. Melgarejo-Meseguer, F. J. Gimeno-Blanes, J. R. Gimeno-Blanes, A. S. Amin, M. M. Winter, H. A. Marquering, W. E. Kok, A. H. Zwinderman, A. A. Wilde and Y. M. Pinto. 'Computer versus cardiologist: Is a machine learning algorithm able to outperform an expert in diagnosing a phospholamban p.Arg14del mutation on the electrocardiogram?' In: *Heart Rhythm* 18.1 (2021), pp. 79–87. ISSN: 1547-5271. DOI: https://doi.org/10.1016/

j.hrthm.2020.08.021. URL: https://www.sciencedirect.com/science/article/pii/S1547527120308614.

[16] H. Smulyan. 'The Computerized ECG: Friend and Foe.' eng. In: *Am J Med* 132 (Dec. 2019), pp. 153–160. ISSN: 1555-7162. DOI: 10.1016/j.amjmed.2018.08.025.

[17] A. Elen and E. Avuçlu. 'A hybrid machine learning model for classifying time series'. In: *Neural Computing and Applications* 34 (Jan. 2022), pp. 1–19. DOI: 10.1007/s00521-021-06457-x.

[18] R. Rodríguez, A. Mexicano, J. Bila, S. Cervantes and R. Ponce. 'Feature Extraction of Electrocardiogram Signals by Applying Adaptive Threshold and Principal Component Analysis'. In: *Journal of Applied Research and Technology* 13.2 (2015), pp. 261–269. ISSN: 1665-6423. DOI: https://doi.org/10.1016/j.jart.2015.06.008. URL: https://www.sciencedirect.com/science/article/pii/S1665642315000103.

[19] M. Wasimuddin, K. Elleithy, A.-S. Abuzneid, M. Faezipour and O. Abuzaghleh. 'Stages-Based ECG Signal Analysis From Traditional Signal Processing to Machine Learning Approaches: A Survey'. In: *IEEE Access* 8 (2020), pp. 177782–177803. DOI: 10.1109/ACCESS.2020.3026968.

[20] A. L. Goldberger, Z. D. Goldberger and A. Shvilkin. *Goldberger's Clinical Electrocardiography*. Ninth Edition. Elsevier, 2017.

[21] T. B. Garcia. *12-lead ECG: the art of interpretation*. Second Edition. Jones & Bartlett Learning, 2015.

[22] C. Clinic. *Atrial Fibrillation (Afib)*. URL: https://my.clevelandclinic.org/health/diseases/16765-atrial-fibrillation-afib) (visited on 01/02/2022).

[23] F. Castells Ramon, C. Mora, J. Roig, J. Rieta, C. Sánchez Meléndez and J. Sanchis. 'Multidimensional ICA for the Separation of Atrial and Ventricular Activities from Single Lead ECGs in Paroxysmal Atrial Fibrillation Episodes'. In: vol. 3195. Sept. 2004, pp. 1229–1236. DOI: 10.1007/978-3-540-30110-3_155.

[24] B. Turner. *What Are the Four Parts of the EKG Machine?* 27.07.2017. URL: https://healthfully.com/four-parts-ekg-machine-7528590.html (visited on 08/02/2022).

[25] D. Fornell. *6 Trends In ECG Systems*. 20.01.2021. URL: https://www.dicardiology.com/article/6-trends-ecg-systems (visited on 08/02/2022).

[26] H. Stores. *What is difference between Single/Three /Six/ Twelve Channel ECG Machine?* URL: https://www.hospitalsstore.com/what-is-difference-between-three-six-twelve-channel-ecg-machine/ (visited on 08/02/2022).

[27] J. W. Grier. *Comparison and review of portable, handheld, 1-lead/channel ECG / EKG recorders*. URL: https://www.ndsu.edu/pubweb/~grier/Comparison-handheld-ECG-EKG.html (visited on 09/02/2022).

[28] W. Tec. *Korean Bionet 12-Channel ECG Cardio 7*. URL: https://www.walentec.com/products/diagnostic/digitalecg/ecg-machines/korean-bionet-12-channel-ecg-cardio-7/ (visited on 09/02/2022).

[29] R. Szeliski. *Computer Vision: Algorithms and Applications*. Second Edition. Switzerland: Springer, 2022.

[30] M. Nixon and A. Aguado. *Feature Extraction and Image Processing for Computer Vision*. Fourth Edition. Academic Press, 2020.

[31] N. Babich. *What Is Computer Vision & How Does it Work? An Introduction*. 28.07.2020. URL: https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/ (visited on 12/02/2022).

[32] S. J. D. Prince. *Computer Vision: Models, Learning, and Inference*. USA: Cambridge University Press, 2012. ISBN: 1107011795.

[33] R. Sagar. *What Is The Difference Between Computer Vision And Image Processing?* 26.12.2018. URL: https://analyticsindiamag.com/what-is-the-difference-between-computer-vision-and-image-processing/ (visited on 12/02/2022).

[34] G. Levin. *Image Processing and Computer Vision*. URL: https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html (visited on 12/02/2022).

[35] G. Boesch. *What is Pattern Recognition? A Gentle Introduction (2021)*. 2021. URL: https://viso.ai/deep-learning/pattern-recognition/ (visited on 14/02/2022).

[36] B. Dickson. *Computer vision applications: The power and limits of deep learning*. 2019. URL: https://bdtechtalks.com/2019/12/30/computer-vision-applications-deep-learning/ (visited on 18/02/2022).

[37] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. J. Topol, J. Dean and R. Socher. 'Deep learning-enabled medical computer vision'. In: *NPJ Digital Medicine* 4 (2021).

[38] Z. Akkus, A. Galimzianova, A. Hoogi, D. Rubin and B. Erickson. 'Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions'. In: *Journal of digital imaging* 30 (June 2017). DOI: 10.1007/s10278-017-9983-4.

[39] M. Sanromán-Junquera, I. Mora-Jiménez, A. Caamaño, J. Almendral, F. Atienza, L. Castilla, A. García-Alberola and J. Rojo-Álvarez. 'Digital recovery of biomedical signals from binary images'. In: *Signal Processing* 92.1 (2012), pp. 43–53. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2011.05.023. URL: https://www.sciencedirect.com/science/article/pii/S0165168411001927.

[40] S. Jayaraman, P. Swamy, V. Damodaran and V. N. Murthy. 'A Novel Technique for ECG Morphology Interpretation and Arrhythmia Detection Based on Time Series Signal Extracted from Scanned ECG Record'. In: Jan. 2012. ISBN: 978-953-307-923-3. DOI: 10.5772/21785.

[41] A. R. G. e Silva, H. M. de Oliveira and R. D. Lins. 'Converting ECG and other paper legated biomedical maps into digital signals'. In: *CoRR* abs/1502.05906 (2015). arXiv: 1502.05906. URL: http://arxiv.org/abs/1502.05906.

[42] S. Ganesh, P. T. Bhatti, M. Alkhalaf, S. Gupta, A. J. Shah and S. Tridandapani. 'Combining Optical Character Recognition With Paper ECG Digitization'. In: *IEEE Journal of Translational Engineering in Health and Medicine* 9 (2021), pp. 1–9. DOI: 10.1109/JTEHM.2021.3083482.

[43] L. Ravichandran, C. Harless, A. Shah, C. Wick, J. Mcclellan and S. Tridandapani. 'Novel Tool for Complete Digitization of Paper Electrocardiography Data'. In: *Translational Engineering in Health and Medicine, IEEE Journal of* 1 (June 2013), pp. 1800107–1800107. DOI: 10.1109/JTEHM.2013.2262024.

[44] F. Badilini, T. Erdem, W. Zareba and A. Moss. 'ECGScan: A method for conversion of paper electrocardiographic printouts to digital electrocardiographic files'. In: *Journal of electrocardiology* 38 (Nov. 2005), pp. 310–8. DOI: 10.1016/j.jelectrocard.2005.04.003.

[45] F. Lozano-Fernandez, I. Jiménez, M. Sanromán, S. Munoz-Romero, A. Garcia-Alberola and J. L. -Álvarez. 'Auto:Cropping of Phone Camera Color Images to Segment Cardiac Signals in ECG Printouts'. In: Sept. 2016. DOI: 10.22489/CinC.2016.122-281.

[46] R. Patil and R. Karandikar. 'Image digitization of discontinuous and degraded electrocardiogram paper records using an entropy-based bit plane slicing algorithm'. In: *Journal of Electrocardiology* 51.4 (2018), pp. 707–713. ISSN: 0022-0736. DOI: https://doi.org/10.1016/j.jelectrocard.2018.05.003. URL: https://www.sciencedirect.com/science/article/pii/S0022073618303200.

[47] H. Myo Tun. 'Analysis on Conversion Process from Paper Record ECG to Computer based ECG'. In: *MOJ Applied Bionics and Biomechanics* 1 (Sept. 2017), p. 11. DOI: 10.15406/mojabb.2017.01.00011.

[48] R. Patil and R. Karandikar. 'Digitization and Parameter Extraction of Preserved Paper Electrocardiogram Records'. In: *Soft Computing and Signal Processing*. Ed. by J. Wang, G. R. M. Reddy, V. K. Prasad and V. S. Reddy. Singapore: Springer Singapore, 2019, pp. 487–495. ISBN: 978-981-13-3600-3.

[49] Y. Li, Q. Qu, M. Wang, L. Yu, J. Wang, L. Shen and K. He. 'Deep learning for digitizing highly noisy paper-based ECG records'. In: *Computers in Biology and Medicine* 127 (2020), p. 104077. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2020.104077. URL: https://www.sciencedirect.com/science/article/pii/S001048252030408X.

[50] N. Mehendale, S. Mishra, V. Shah, G. Khatwani, R. Patil, D. Sapariya, D. Parmar, S. Dinesh and P. Daphal. 'ECG Paper Record Digitization and Diagnosis Using Deep Learning'. In: *SSRN Electronic Journal* (Jan. 2020). DOI: 10.2139/ssrn.3646902.

[51] R. Brisk, R. Bond, E. Banks, A. Piadlo, D. Finlay, J. McLaughlin and D. McEneaney. 'Deep learning to automatically interpret images of the electrocardiogram: Do we need the raw samples?' In: *Journal of Electrocardiology* 57 (2019), S65–S69. ISSN: 0022-0736. DOI: https://doi.org/10.1016/j.jelectrocard.2019.09.018. URL: https://www.sciencedirect.com/science/article/pii/S0022073619303760.

[52] K. C. D. of Emergency Medicine. *EKG's for EM Physicians*. 2012. URL: https://kchemekg.wordpress.com/ (visited on 05/03/2022).

[53] G. J. Taylor. '150 Practice ECGs'. In: *150 Practice ECGs:Interpretation and Review*. Third Edition. Blackwell Publishing Ltd, 2006, pp. 63–213. ISBN: 9780470693964. DOI: https://doi.org/10.1002/9780470693964.part2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470693964.part2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470693964.part2.

[54] K. Wang. *Atlas of Electrocardiography*. G - Reference,Information and Interdisciplinary Subjects Series. Jaypee Brothers Medical Publishers Pvt. Limited, 2013. ISBN: 9789350902097. URL: https://books.google.no/books?id=bVaopNqIhGwC.

[55] A. Mattu and W. Brady. *ECGs for the Emergency Physician 1*. Wiley, 2013. ISBN: 9781118682159. URL: https://books.google.no/books?id=9I-gCMW0Z8QC.

[56] Physionet. *Databases*. 2022. URL: https://physionet.org/about/database/ (visited on 05/03/2022).

[57] Wikipedia. *Edge detection*. URL: https://en.wikipedia.org/wiki/Edge_detection#cite_note-1 (visited on 04/05/2022).

Appendix

In this appendix we show the pdf output of the code developed in this thesis. The code of the original tool can be found in the Github link https://github.com/Sushil-Acharya/ECG-Digitization/blob/main/disconnected-curves-final.ipynb.

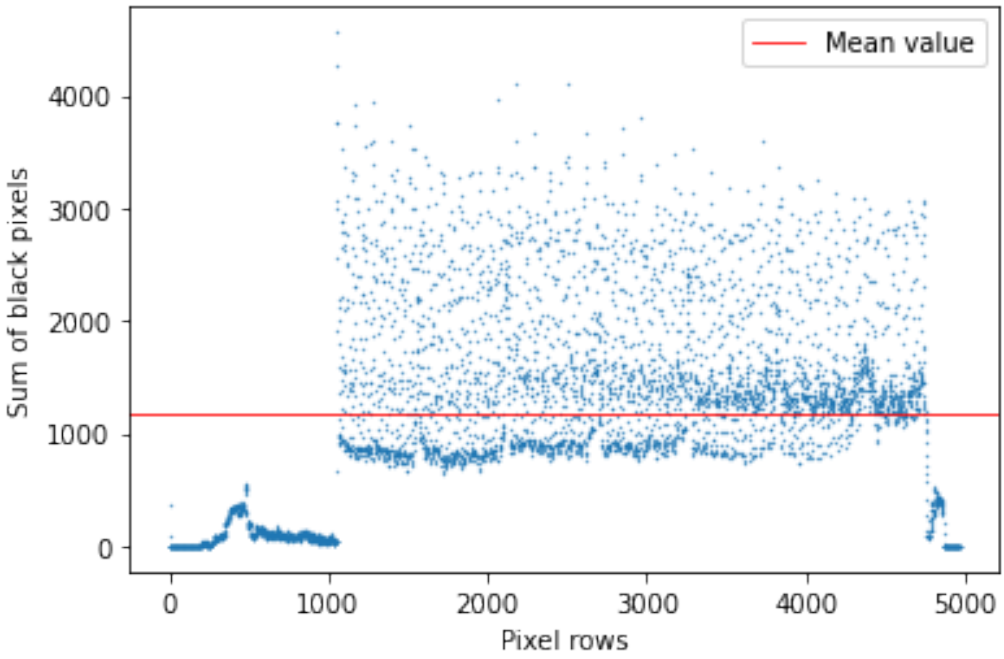
Approaching the challenges

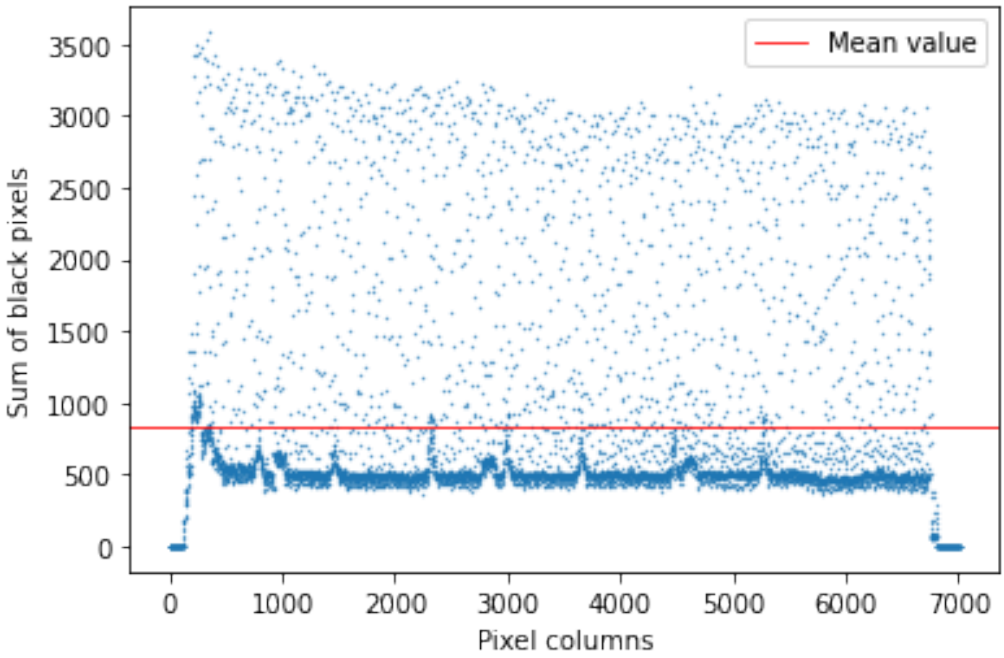
May 15, 2022

[17]:
```
### Automated cropping technique with horizontal and vertical projections

import cv2
import numpy as np
from matplotlib import pyplot as plt
import os
import more_itertools as mit

path = "D:\ACIT - Oslomet\9. Master Thesis\Stored_images"

for i in range(18):
    if i<9:
        var = 'D:\ACIT - Oslomet\9. Master Thesis\Ahus␣
↪Scanns\EKG_ACE2_20210511110737_page_000'+str(i+1)+".tif"

    else:
        var = 'D:\ACIT - Oslomet\9. Master Thesis\Ahus␣
↪Scanns\EKG_ACE2_20210511110737_page_00'+str(i+1)+".tif"

    # read the scan
    original_image = cv2.imread(var)
    # make it grayscale
    gray = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)
    # make it binary
    ret, binary_image = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

    # Convert black pixels to ones
    binary_image[binary_image == 0]   = 1
    # Convert white pixels to zeros
    binary_image[binary_image == 255] = 0
    height, width = binary_image.shape

    ## Find the cropping boundaries in the y axis (height)

    horizontal_projection = np.sum(binary_image, axis = 1)

    # make visual plot of the horizontal projection
```

```
    a=np.arange(height)
    plt.scatter(a,horizontal_projection,s=0.1)
    plt.axhline(y= np.mean(horizontal_projection) , color='r',linewidth=0.
↪8,label="Mean value")
    plt.xlabel("Pixel rows")
    plt.ylabel("Sum of black pixels")
    plt.legend()
    save_plot_hor = path+r"\horizontal projection plots\hor_proj"+str(i+1)+".
↪png"
    plt.savefig(save_plot_hor,dpi=300)
    plt.show()

    # make a list of the pixel rows whose projection values are not zero
    non_zero_position_hor = []
    # make a list of the pixel rows whose projection values are above average
    above_mean_position_hor = []

    for j in range(height):
        if horizontal_projection[j] != 0:
            non_zero_position_hor.append(j)
        if horizontal_projection[j] >np.mean(horizontal_projection) :
            above_mean_position_hor.append(j)

    # group all the elements of the non_zero_position_hor list which have␣
↪consecutive numbers
    grplist_hor = [list(group) for group in mit.
↪consecutive_groups(non_zero_position_hor)]
    # find the longest group in that list
    longest_list_hor = max(grplist_hor, key=len)
    # find commomn elements between the longest_list_hor and the␣
↪above_mean_position_hor list
    common_hor = list(set(above_mean_position_hor) & set(longest_list_hor))

    # the desired boundaries are the first and the last elements of the above␣
↪list
    bottom = common_hor[0]
    top = common_hor[-1]


    ## Find the cropping boundaries in the x axis (width)

    vertical_projection = np.sum(binary_image, axis = 0)

    # make visual plot of the vertical projection
    b=np.arange(width)
    plt.scatter(b,vertical_projection,s=0.1)
```

```
    plt.axhline(y= np.mean(vertical_projection) , color='r',linewidth=0.
↪8,label="Mean value")
    plt.xlabel("Pixel columns")
    plt.ylabel("Sum of black pixels")
    plt.legend()
    save_plot_ver = path+r"\vertical projection plots\ver_proj"+str(i+1)+".png"
    plt.savefig(save_plot_ver,dpi=300)
    plt.show()

    # make a list of the pixel columns whose projection values are not zero
    non_zero_position_ver = []
    # make a list of the pixel columns whose projection values are above average
    above_mean_position_ver = []

    for j in range(width):
        if vertical_projection[j] != 0:
            non_zero_position_ver.append(j)
        if vertical_projection[j] >np.mean(vertical_projection) :
            above_mean_position_ver.append(j)

    # group all the elements of the non_zero_position_ver list which have␣
↪consecutive numbers
    grplist_ver = [list(group) for group in mit.
↪consecutive_groups(non_zero_position_ver)]
    # find the longest group in that list
    longest_list_ver = max(grplist_ver, key=len)
    # find commomn elements between the longest_list_ver and the␣
↪above_mean_position_ver list
    common_ver = list(set(above_mean_position_ver) & set(longest_list_ver))

    # the desired boundaries are the first and the last elements of the above␣
↪list
    left = common_ver[0]
    right = common_ver[-1]

    # crop the image according to the boundaries and save it
    cropped_image = original_image[bottom:top,left:right]
    name = "cropped"+str(i+1)+".png"
    save_crop = path+"\cropped images with projection"
    cv2.imwrite(os.path.join(save_crop,name),cropped_image)
```





[4]:
```
### Spliting a single cropped scan into six overlapping strips and then␣
↪applying and testing different CCL threshold values for each strip
```

```
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
import numpy as np
import cv2
import math
from scipy import ndimage
import os


original_image = cv2.imread(r"D:\ACIT - Oslomet\9. Master Thesis\Ahus␣
 ↪Scanns\EKG_ACE2_20210511110737_page_0010.tif")


### Cropping
imag = cv2.resize(original_image, (0, 0), fx=0.4, fy=0.4)
image = imag.copy()
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (3, 3), 0)
canny = cv2.Canny(blurred, 120, 255, 1)

# Find contours in the image
cnts = cv2.findContours(canny.copy(), cv2.RETR_EXTERNAL, cv2.
 ↪CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]

# Obtain area for each contour
contour_sizes = [(cv2.contourArea(contour), contour) for contour in cnts]

# Find maximum contour and crop for ROI section
if len(contour_sizes) > 0:
    largest_contour = max(contour_sizes, key=lambda x: x[0])[1]

    x,y,w,h = cv2.boundingRect(largest_contour)

    ROI = image[y:y+h, x:x+w]

path = "D:\ACIT - Oslomet\9. Master Thesis\Stored_images"
cv2.imwrite(os.path.join(path , 'cropped.png'),ROI)


### Rotating
img_before = cv2.imread(os.path.join(path , 'cropped.png'))
img_before_copy = img_before.copy()
img_gray = cv2.cvtColor(img_before_copy, cv2.COLOR_BGR2GRAY)
img_edges = cv2.Canny(img_gray, 100, 100, apertureSize= 3)
```

```
lines = cv2.HoughLinesP(img_edges, 5, math.pi / 180.0, 100, minLineLength=100,␣
 ↪maxLineGap= 5)

angles = []

for [[x1, y1, x2, y2]] in lines:
    cv2.line(img_before_copy, (x1, y1), (x2, y2), (255, 0, 0), 1)
    angle = math.degrees(math.atan2(y2 - y1, x2 - x1))
    angles.append(angle)

median_angle = np.median(angles)

rotated = ndimage.rotate(img_before, median_angle, cval=255)
cv2.imwrite(os.path.join(path , 'rotated.jpg'),rotated)

### Erosion
img = cv2.imread(os.path.join(path ,'rotated.jpg'), 0)

kernel = np.ones((2,2), np.uint8)
img_erosion = cv2.erode(img, kernel, iterations=1)
cv2.imwrite(os.path.join(path ,'erosion.jpg'), img_erosion)


### Blurring
mask = cv2.imread(os.path.join(path ,'erosion.jpg'), 0)

sliding_window_size_x = 9
sliding_window_size_y = 9

mean_filter_kernel = np.ones((sliding_window_size_x,sliding_window_size_y),np.
 ↪float32)/(sliding_window_size_x*sliding_window_size_y)
filtered_image = cv2.filter2D(mask,-1,mean_filter_kernel)

invert = cv2.bitwise_not(filtered_image)
cv2.imwrite(os.path.join(path ,'filtered_image.png'), invert)


### Spliting the scan into six strips
before_spliting = cv2.imread(os.path.join(path ,'filtered_image.png'))
height = before_spliting.shape[0]
width = before_spliting.shape[1]
cropped_50 = before_spliting[50 : height - 50, 50 : width - 50]
slice1 = cropped_50[:370,:]
slice2 = cropped_50[160:590,:]
slice3 = cropped_50[390:820,:]
slice4 = cropped_50[620:1050,:]
slice5 = cropped_50[850:1280,:]
```

```
slice6 = cropped_50[1080:,:]

slice_list = [slice1,slice2,slice3,slice4,slice5,slice6]

# making a list with different CCL threshold values
cc_area_list = []
for i in range(10):
    y=30000*i
    cc_area_list.append(y)

structure = np.array([[1, 1, 1],
                      [1, 1, 1],
                      [1, 1, 1]], np.uint8)

### Unmark the followings in case you need to apply erosion and blurring in␣
 ↪every iteration of the loop
# kernel_loop = np.ones((2,2), np.uint8)
# sliding_window_x_loop = 9
# sliding_window_y_loop = 9
# mean_filter_kernel_loop = np.
 ↪ones((sliding_window_x_loop,sliding_window_y_loop),np.float32)/
 ↪(sliding_window_x_loop*sliding_window_y_loop)


### Applying different threshold values for every strip and save the outcome
for j in range (6):
    image = "slice"+str(j+1)+".png"
    cv2.imwrite(os.path.join(path,image),slice_list[j])
    strip = cv2.imread(os.path.join(path,image))
    for i in range(10):
        dark = (0, 0, 0)
        light = (120, 150, 150)
        mask = cv2.inRange(strip, dark, light)
        invert = cv2.bitwise_not(mask)
        labeled_image, cc_num = ndimage.label(invert, structure=structure)
        cc_areas = ndimage.sum(invert, labeled_image, range(cc_num + 1))
        area_mask = cc_areas < cc_area_list[i]
        labeled_image[area_mask[labeled_image]] = 0
        labeled_image = np.where(labeled_image == 0, 255, 0)

        cv2.imwrite(os.path.join(path,'temp.png'),labeled_image)

        outcome_name = "slice"+str(j+1)+"_"+ str(cc_area_list[i])+".png"
        cv2.imwrite(os.path.join(path,outcome_name),labeled_image)

# Unmark the followings in case you need to apply erosion and blurring in every␣
 ↪iteration of the loop
```

```
#         outcome = cv2.imread(os.path.join(path,'temp.png'))
#         outcome_erosion = cv2.erode(outcome, kernel_loop, iterations=1)
#         outcome_filtered = cv2.
 ↪filter2D(outcome_erosion,-1,mean_filter_kernel_loop)
#         strip=cv2.bitwise_not(outcome_filtered)
```