Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Cybernetics

Diploma thesis

Universal Optical Data Acquisition from Patient Monitor

Bc. David Tošner

Supervisor: Ing. Filip Ježek

Study programme: Biomedical Engineering and Informatics

Specialisation: Biomedical Informatics

January 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student:	Bc. David T o š n e r
Study programme:	Biomedical Engineering and Informatics
Specialisation:	Biomedical Informatics
Title of Diploma Thesis:	Universal Optical Data Acquisition from Patient Monitor

Guidelines:

Currently, the possibilities for data segmentation from patient monitor are limited to proprietary solutions, provided from the machine vendors. The goal of this thesis is to prepare low-cost and robust solution to gather the data through optical camera and image segmentation methods. Possibly, the whole systém could run in realtime.

- Sum availiable methods for data acquisition from patient monitor.
- Design a process for data acquisition, discuss most suitable algorithms for each individual step.
- Implement basic standalone application, which could segment numeric values and save them to database with sample of no more than 1s.
- Show at least basic method for segmenting the curves as well.
- Seek to enhance the robustness, e.g. by empowering two or more cameras at the same time.
- Assess the application performance on sample video and discuss future enhancements.

Bibliography/Sources:

[1] Šonka M., Hlaváč V., Boyle R.: Image Processing, Analysis and Machine vision, 3rd edition, Thomson Learning, Toronto, Canada, 2007.
[2] Svoboda T., Kybic J., Hlaváč V.: Image Processing, Analysis and Machine Vision - A MATLAB Companion. Thomson, Toronto, Canada, 1 edition, 2007.
[3] Žára J., Beneš B., Sochor J., Felkel P.: Moderní počítačová grafika. 2. vyd. Brno : Computer Press, 2004. ISBN 80-251-0454-0.

Diploma Thesis Supervisor: Ing. Filip Ježek

Valid until: the end of the summer semester of academic year 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic	prof. Ing. Pavel Ripka, CSc.
Head of Department	**Dean**

Prague, December 12, 2015

Acknowledgements

I would like to thank everyone who helped me while I was working on my diploma thesis. First of all, I would like to mention my supervisor Ing. Filip Ježek for a great support. Also, I would like to thank CVUT University, which enabled me to look into the field of image processing, which allowed me to create this work. At the end, I would like to thank my family who gave me the appropriate background support.

Declaration

Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date…………………… ……………………………………….

signature

Abstrakt

Hlavním cílem této práce je navrhnout levné způsoby jak získat data, hlavně číslice, z pacientského monitoru pomocí kamery v reálném čase. Navržené přístupy by měli být nezávislé na typu a značce monitoru. Každý z těchto přístupů bude dekomponován na jednotlivé kroky, jako jsou: předzpracování, segmentace, výběr kandidátů, klasifikace atd. Pro tyto kroky budou zmíněné různé metody a bude ukázáno jejich použití přímo na snímcích pacientského monitoru. Další částí je implementace jednoho z těchto přístupů, s vhodně vybranými metodami, jako samostatnou aplikaci v programovacím jazyku C#. Aplikace vyextrahuje numerická data ze zachyceného pacientského monitoru a klasifikuje je, pokud to je možné. Také se bude umět vypořádat s běžnými situacemi které mohou nastat v pokoji kde je pacientský monitor umístěn jako zakrývání, změna světelných podmínek nebo jemné pohyby s monitorem. Po samotné implementaci bude aplikace podrobena testům za různých podmínek a budou diskutovány možná rozšíření, jako extrakce křivek nebo více kamerový přístup. Na začátku zmíním všechny aktuální možnosti pro získávání dat z pacientského monitoru.

Klíčová slova: pacientský monitor, reálný čas, sběr dat, kamera, OCR

Abstract

The main goal of this thesis is to design low-cost approaches for data acquisition, especially digits, from patient monitor via a camera in real-time. Designed approaches should be independent of type and brand of the monitor. Both of these approaches will be decomposed in separated steps like preprocessing, segmentation, selection of candidates and classification. For these steps will be mentioned different methods and their use will be demonstrated directly on actual figures of the patient monitor. Next part is implementation one of these approaches, with suitably selected methods, as a standalone application in programming language C#. This application will extract digits from the captured patient monitor and classify them if it is possible. Also, this application will be able to deal with common situations in a room where the patient monitor is located like covering, change of light conditions or gently moving with the patient monitor. After the implementation application is tested under different conditions and we discuss possible extensions like curve extraction or multi-camera approach. In the beginning, all current solution for data acquisition from the patient monitor at the moment will be mentioned.

Keywords: patient monitor, real-time, data acquisition, camera, OCR

Contents

1 INTRODUCTION

The patient monitor is widespread equipment. It is one of the most essential machines for the medical sector, and we can find it at various places from an operating room to the room in a retirement home i.e. in the whole sector of medical and social support. The patient monitor provides a staff of these sectors data about vital functions of the patient like blood pressure, heart rate, pulse and respiratory rate. Staff in the medical sector and the others use these data to make diagnoses, change dosages of drugs and other things which in the end help the patient, but they usually use only real-time data on the patient monitor, for that.

Only a few patient monitors which are in use at the moment have possibly to save acquired data about the patient and export them for next further processing. That is a shame because nowadays we have sophisticated statistical methods and machine learning processes which can extract interesting trends in acquired data. Also, application for data processing and computers with appropriate performance are available at a reasonable price now.

Some patient monitors have the possibility to save and extract data via cable, but their usage in the medical sector is low at the moment. Another way how to acquire data from a patient monitor is via their interface, these solutions work quite well, but they are specified to concrete patient monitor model or at least to the brand. The interesting way how to extract data from the patient monitor is optical recognition of the data on the patient monitor screen. For this purpose, we can use two different ways. One of them extracts data on the patient monitor screen via connected video grabber. This is a good solution because extracted frames are not affected by any distortion, and we can extract data from them nicely, but these grabbers are relatively expensive. The second optical solution is via a camera set in the room, where a patient monitor is located, and directed to the patient monitor screen. Frames captured by this solution are not as good as via grabbers because they are affected by many kinds of distortion. The big advantage of this solution is its price because computers and webcams can be bought for an affordable price at the moment. Because of that, I choose that solution for future implementation.

In this thesis, at first, I mention available solutions for data acquisition from patient monitor. Design two different approaches how to acquire data from them and analyze these approaches step by step with suitable possible methods for these steps. These approaches will be independent of type or brand of the patient monitor After that I select one of these approaches and implement it as a standalone application which acquires digits from the patient monitor via the camera in real-time. In the end, the application will be tested under different conditions and possible extensions will be mentioned. It is important to say that output from my application is inappropriate to make a patient decision, but only for academicals studies.

2 PATIENT MONITOR

In the hospital, we can find many different devices which display data about the patient for example ultrasound, microscope, neuro-navigation and a patient monitor. The patient monitor is one of the most essential devices in hospitals. Sometimes we can find in literature patient monitor called bedside monitor because the patient monitor is ordinary located beside the patient bed.

The patient monitor is usually used to provide data about patients with life-threating illnesses or injuries at departments like intensive care unit or critical care unit. These data are obtained continually along the time, and it is crucial to get them at any moment. Because of that, the reliability of patient monitors must be almost total. Reliability of patient monitor is ensured by a various number of certification which every patient monitor in use must have [1].

The patient monitor checks and show us data about vital functions of the patient like blood pressure, heart rate, saturation and respiratory rate. All these data are observed via sensors on a human body to check the condition of the patient over time. The amount of shown data correlates with the price of the monitor. The common type of patient monitor is shown in the Figure 1.

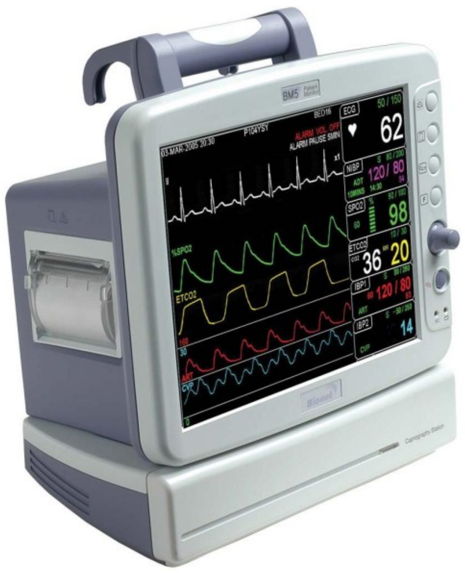


Figure 1 The standard patient monitor. Taken from [2]

2.1 SPECIFICATION OF A PATIENT MONITOR

A patient monitor has a specific structure. This fact you can see in the Figure 1. Some of these characteristics should be useful during the data acquisition process.

First of these facts is a black background of the patient monitor screen. This trend has almost all monitors on the market. I went through various manufacturers, which made them, and I could say about ninety-nine percent of actual patient monitors on the market has a black background or at least this possibility.

All numerical values and curves are highly contrasted against the background of the monitor. This fact is comfortable for personal in hospitals or other sectors where we can found patient monitors and also for a segmentation process. When we have a big difference between the contrast of the background and data on it, it is easier to distinguish what is background and what are data.

Third Interesting fact about the patient monitor is color consistency. As you can see in the Figure 1 almost all data on the patient monitor has its own color and that color is consistent. That means we can use this information lately during the candidates selection, before the classification. The consistency is also advantaging during the segmentation process because it is easier to found the borders of the foreground.

The last specification is a type of the font used by the patient monitor. Usually, the font shown by the patient monitor is similar to synthetic fonts. The reason for this is easy readability for the staff of the medical sector.

2.2 Types of patient monitors

Nowadays there are plenty types of patient monitors on the market. For example, when you look at Figure 2 you can see that position of heart rate is on the first monitor in top left corner and on the second one in right top corner. Some types of patient monitors have a possibility to select the appropriate area for a particular parameter.

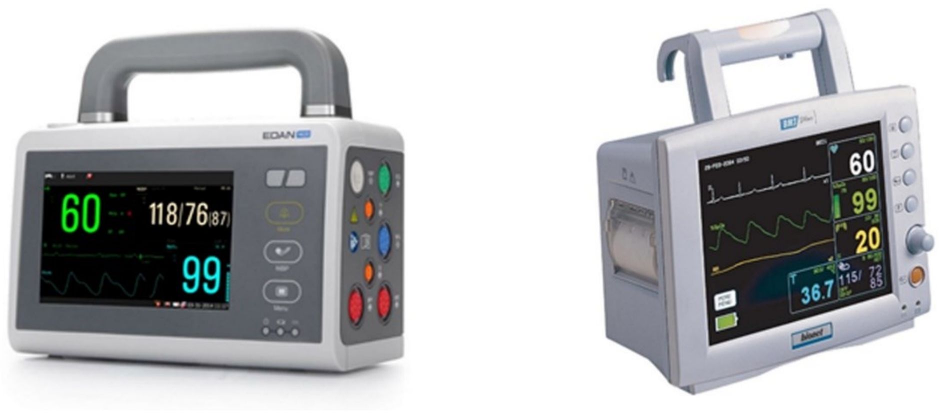


Figure 2 Types of patient monitors. Taken from [3], [4]

The fact that there are many various types of monitors escalate in the almost impossibility to cover all these types and use the information about the position of specified data on them for data acquisition. A solution to this problem is a universal approach which will be calibrated to particular patient monitor by the user before the data acquisition.

2.3 Environment where we can find patient monitors

A patient monitor has various places where can be used from operating room to room in a retirement home i.e. in the whole sector of medical and social support. All these places contain different objects and gizmos. The overall diversity of the entire scene of the room is complex. The example of the patient room is shown in the Figure 3.

Figure 3 A room with a patient monitor. Taken from [5]

It is certain that during the data acquisition process someone, like personal or patient, covers the screen by his body. That means it is impossible to classify data from the patient monitor because they are hidden.

Another problem is gently moving with a patient monitor. For example, when a doctor comes to a patient room and wants to show someone else data about the patient on it, he rotates the monitor by few degrees around the vertical axis.

3 MAIN GOALS

This diploma thesis has a few topics which should be covered. These topics are a sum of available methods for data acquisition of patient monitor to implement a basic standalone application. In upcoming subchapters, every goal of this thesis is described.

3.1 SUM OF ACTUAL SOLUTIONS OF DATA ACQUISITION PROCESS FROM PATIENT MONITOR

Task of this part is to make a research trough articles and already implemented solutions, which deals with this problem and what kind of methods and hardware are in use now for data acquisition from a patient monitor.

3.2 DESIGN A APPROACHES FOR DATA ACQUISITION VIA CAMERA

Another part of diploma thesis is a design of approaches which acquire and classify data, especially numeric ones, from a patient monitor captured via camera. Discuss individual steps of these approaches and propose the most suitable algorithms for these individual steps.

3.3 ANALYSIS OF SELECTED APPROACH

One of the mentioned approaches will be analyzed step by step. These steps correspond to actual steps in future application. For each of these steps will be presented more possible methods. These methods will be analyzed, and I discuss their usability for optical data acquisition from a patient monitor.

3.4 IMPLEMENTATION OF SELECTED APPROACH

The selected approach will be implemented as a standalone application in programming language C#. This application should acquire numeric data from patient monitors across different brands and types. Data will be extracted in real-time by the connected camera. Afterward, they will be saved to the external file.

3.5 EXPERIMENTS ON SELECTED APPROACH

The selected approach will undergo experiments under different conditions. Basic conditions are a different light condition in the room where a patient monitor is located, different angle of camera plane against the plane of the patient monitor screen and different distance to the patient monitor. More sophisticated experiments test the application under conditions which can happen during the ordinary day in the medical sector. These conditions are covering of patient monitor and gently movement with the patient monitor. This part of the thesis also contains basic performance tests of the implemented application.

3.6 DISCUS POSSIBLE EXTENSIONS OF APPLICATION

In this part of the thesis I go through possible extensions of the application and generally the whole work. One of the possible future extension is segmentation of curves from the patient monitor. Other mentioned extension is a usage of multiple cameras to secure better result of the whole process. Also, other enchantments will be discussed in this chapter.

4 CURRENT SOLUTIONS

The patient monitor is one of the most accurate solution how to noninvasive obtain data about the patient. These monitors are really useful for personal in the medical sector. All data on them are nicely readable, but there is a problem with their storage. Many monitors do not have a possibility to storage data which they show, but there are some alternatives.

4.1 STORAGE DIRECTLY IN PATIENT MONITOR

Storage inside of the patient monitor is the best solution. The shown data and the storage data are the same. Also, storage inside the patient monitor is most user-friendly for personal in the medical sector. Data are saved automatically into the inner storage as a loop with an interval equal to hundreds of hours. When personal want stored data, he can connect the USB cable and simply download the data.

This solution has some problems. First of them is the usage of these patient monitors in the medical sector at the moment. Simply the older types of patient monitors do not have this possibility. I do not have some statistical data, but from visual checking of medical workplaces and discussion with experts in this field, an amount of patient monitors with the possibility to save data is low. Another problem of these monitors are their price, which is relatively high, but these monitors getting more and more ordinary. The last problem of this approach is the closed format of data because some brands use their specific format of data to force users to use their software. Usual maximal extraction frequency is five minutes, and these monitors ordinary don’t have the possibility to save curves. Data could also be saved by an optional device connected to a patient monitor, but that means another cost.

4.2 REAL-TIME CABLE APPROACH

Some monitors do not have inner storage, but there is a possibility to acquire data from other connected devices. Of course, monitors could have both of these features. Ordinary the companies have its solutions how to acquire real-time data. However, these solutions are not ordinary free, and you need separate device concentrator and data consolidation server. Some companies for example Philips try to make output ports from patient more open with their IntelliBridge device interfacing [6].

There is also some third party software which can acquire data from patient monitor directly via cable to your PC for example Medicollector [7]. However, this software is still relatively pricey.

Some papers already look at this problem and try to solve it [8]. This presented solution shows a nice free alternative how to acquire data from a patient monitor. Because a cable sends data, the error is lower than via optical acquisition. This concrete work is far from being complete, but it is the step forward. This approach is specified to the concrete type of the patient monitor or at least to a group of monitors.

The problem of these solutions is an issue with the law. Because based on law 268/2014 for the Czech Republic, the medical device can be only used for its intended purpose according to the manufacturer's instructions [1]. That means if we want to use real-time cable approach, instructions for that must be included in manual released by the manufacturer. This option is possible, but in almost all cases only for closed solutions created by manufacturers.

4.3 OPTICAL DATA ACQUISITION FROM PATIENT MONITORS

Patient monitors show us all data about the patient via a graphical interface. As you can see in Figure 1, the doctor, or another specialist can easily read all data on the screen. So if the doctor can read graphical interface via the eye, we can design the process which looks at this graphical interface exactly same as a doctor. That means we store the whole screen, or only part of it, as a figure and apply on that figure algorithms which extract data from them. We have two different approaches how to obtain a graphical view of the patient monitor screen i.e. video signal.

4.3.1 Via grabbers

The first method uses VGA or DVI grabbers. These machines acquire video signal via their VGA or DVI connectors, convert them, and send them to the computer. Video grabbing approach is for example tested at intra operative ultrasound, neuro/navigation and microscope. This approach has one big advantage. The video grabbed from the patient monitor is almost the same as video signal show on the patient monitor screen. That means there is almost no distortion, the problem with low resolution and the problem with blurred contours. Because of that, the data extracted from video signal have accuracy bigger than 95%. [9]

There are also some disadvantages. The first of them is the price. This type of solution is still relatively pricey because the grabber cost about $750, its price depend on the type of the grabber [10]. Another problem with this approach is already mentioned in chapter 4.2 and its case of law 268/2014 [1]. However, as you can read in [9] *the video signal interfaces of the medical devices are certified to provide video signals. Therefore, using the video signal does not violate certification.* That statement is defined for Germany.

4.3.2 Via camera direct to patient monitor

The video signal of the patient monitor could also be recorded noninvasively by the camera. This camera is directed to the patient monitor screen to read all, or only part, of the data on the screen. This is an approach which I have not found in any article or journal, but it is still possible that someone is already working or worked on this approach, and that is the reason why I choose that way.

The big advantage is very low cash requirements. All equipment what is necessary for this approach is a computer and a camera. Nowadays notebooks and webcams are a normal part of almost all workplaces and everyday life for all of us, which makes this solution very available. The second advantage of this approach is an issue with the law. As I mentioned above, this approach is completely noninvasive to a patient monitor that means you can extract information from the patient monitor without any restriction by a law 268/2014 [1]. However, it is necessary not to forget on medical confidentiality and civil rights.

The disadvantage of this solution is the accuracy of obtained data. All frames captured by the camera are load with the optical problems. These problems are for example small resolution, reflection, distortion, dependent noise, impurity or quality of CCD sensor and independent noise caused by screeches on the lens.

5 Possible approaches

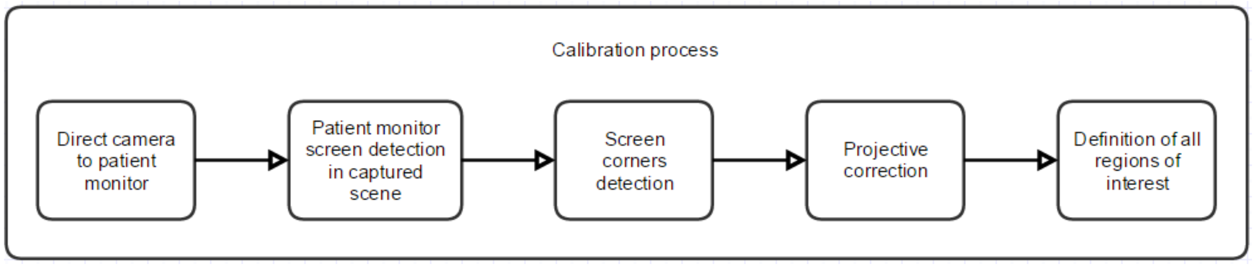
In this chapter, I introduce two designed approaches how to acquire data from a patient monitor. Both of these approaches are designed by me and have advantages and disadvantages. At first, I will introduce the whole pipeline of approaches and afterward describe every single step. On each of these steps could be used different types of algorithms. These algorithms will also be mentioned in this chapter with dependence to data acquisition process from a patient monitor.

The first approach will use a patient monitor screen as the main structure. The screen must be found in every captured frame. Once we have the screen, we can correct projective distortion by founded corners of the patient monitor screen, and extract data from a concrete position on the screen. This position will be the same every time after the correction. The second approach is based on digits. This approach finds digits in a scene during the calibration process and tries to focus on them for the whole process of data acquisition. That means it is not necessary to find them in the whole scene in every frame.

5.1 Monitor based approach

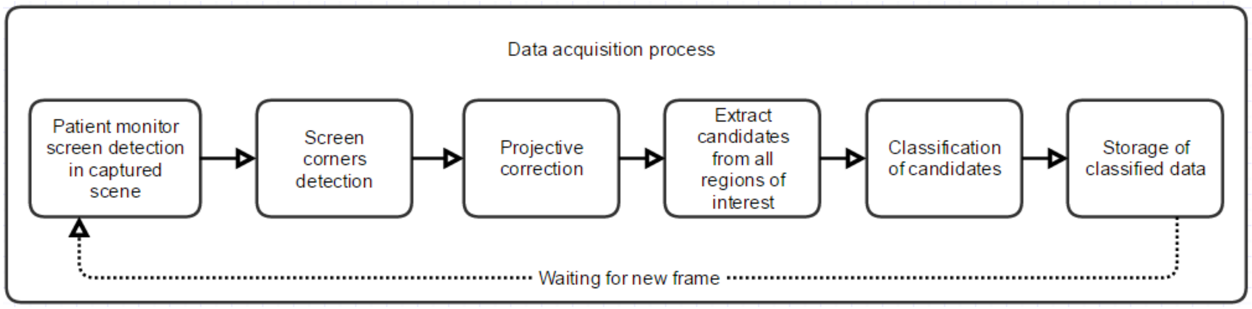
The first approach is strictly dependent on the patient monitor as a whole. That means the whole patient monitor, or at least whole patient monitor screen, have to be in the captured scene by a camera. Without this requirement, it is impossible to work further in the pipeline of this approach.

The whole process starts with camera directing to the patient monitor. Before the acquisition process, it is necessary to do some calibration steps. The first step is recognition of the patient monitor screen in the captured scene. The screen can be found in the whole scene or user can define smaller crop of the scene for better results. The patient monitor screen can be found by different approaches and once when we find that screen, we can find the corners of the screen for upcoming projective correction and do it. When we have a corrected scene by a homogenous matrix, we can define regions of interest on the monitor, for concrete data on the monitor. These regions contain data about vital functions, a list of them is mentioned in chapter 2. The pipeline of the calibration process is shown in the Schema 1.


Calibration process
Direct camera to patient monitor
Patient monitor screen detection in captured scene
Screen corners detection
Projective correction
Definition of all regions of interest

Schema 1 Calibration process for monitor based approach

When we added all regions of interest, we can start data acquiring process from the patient monitor. This process is almost the same as calibration process, only on the end of its pipeline, we add preprocessing and the actual classification. After the classification, the last step is storage of classified data and repeat acquisition process for next frame captured by the camera. The whole pipeline of data acquisition process is shown in the Schema 2.


Data acquisition process
Patient monitor screen detection in captured scene
Screen corners detection
Projective correction
Extract candidates from all regions of interest
Classification of candidates
Storage of classified data
Waiting for new frame

Schema 2 Data acquisition process for monitor based approach

The big advantage of this approach is the fact that once when we found corners of the monitor, it is relatively easy to locate concrete regions of interest. For example when someone steps before the patient monitor and moves with it for a small distance. The acquisition process is trying to find the monitor in a whole scene and once when a person finally is not covering the patient monitor this process find it and continue with acquisition process because the region of interest is in the same position on the patient monitor screen after the projective correction.

The first and biggest disadvantage of this process is an impossibility to do acquiring process when there is no light in the room, where the patient monitor is located. That disadvantage is mentioned in requirements below 5.1.1. Another small disadvantage of this process is bigger CPU load because we need to find the patient monitor in every captured frame before the actual acquisition process. This problem could be fixed by resizing of the captured scene to a smaller scale, find a patient monitor and rescale it back in high resolution. Also, projective correction applied to the whole patient monitor screen is relatively CPU difficult, because it is applied screen pixel by pixel.

5.1.1 Requirements

This approach is strictly dependent on the whole patient monitor as I already mentioned in 5.1. The reason is that we need to find corners of the patient monitor for a projective correction. Once we do not have one, or more corners, in the captured scene, this approach cannot work further. So that means we need to have the whole screen of the patient monitor in the captured scene for the whole time when data acquisition process is in progress.

The next requirement is about the light condition in the room. It is necessary to have some light in the room where the patient monitor is located. It is because we need to find borders of the patient monitor screen. When all lights are off, we cannot distinguish where the screen starts and ends. That means it is impossible to find corners of this screen for projective correction. This requirement is a big problem because the data acquisition from patient monitor it is better to do at night. Because the subject is calm in a bed and the fluctuation of hospital personnel is low.

The captured plane by the camera should be as most as possible parallel to the plane of the patient monitor screen. When this condition is met, there is less distortion and classify, and also the whole process will be easier and give us better results.

The monitor must be turned on during the process of calibration. Without that condition, it is impossible to do this step. It is because the software needs to extract some information from the monitor, for example, the size of fonts and the position of regions of interest before the actual acquisition process.

Symbols on the patient monitor should be highly contrasted against the background of it. As I mentioned in 2.1, this fact is usually secured by the structure of the patient monitor.

Fonts shown on the patient monitor should be very similar to synthetic fonts. This condition is necessary because classifying of candidates for digits is done by library functions of third parties. In this case, it is Tesseract [11]. This library works in basic with synthetic fonts. As I mentioned in 2.1, patient monitors usually use synthetic fonts.

Camera used for capturing has the adequate resolution or quality. That means the camera should capture digits whose height will be minimally 40 pixels in the captured scene and this camera should also be focused on them. This condition must be met because smaller symbols have a tendency to connect together and Tesseract also has this condition in recommendations [12].

5.1.2 Individual steps

In this chapter will be presented and discussed possible methods and algorithms used in a monitor based approach. This approach is separated into four main parts. The first part is focused on finding of the patient monitor in the scene, or at least in the crop of the scene. That part is essential because without founded patient monitor screen i.e. corners of the screen we cannot do the next part. The second part is a projective correction of the patient monitor screen. Next step is a region of interest segmentation, and in the end, we look at the actual classification.

5.1.2.1 Patient monitor screen detection

Detection of the patient monitor screen in the scene is an essential part of this approach. All following steps strictly depend on this step. For detection of the patient monitor screen, we can choose many different approaches. In following subchapters, I describe few of them.

All of these algorithms can be used on the whole captured scene or on the smaller crop where the patient monitor is located. When we use the whole scene, the process takes more time, and there is also the bigger possibility of misinterpretation. Because of that, it is better to use a crop of the captured scene. I work with it in following chapters.

5.1.2.1.1 Corner detectors

The corner can be intuitively described as a junction of contours. Their advantage is that they are localizable in the scene. Corners are stable features over changes of viewpoint and different lighting. Another way how to describe corner is that corner is a point which has large variations in the neighborhood in all directions [13]. Sometimes corners are called interest points.

There are several methods how to detect a corner in the scene. Simply we can split corner detectors into few groups. First of them are edge based methods which operate on the principle of monitoring the changes in the curvature of edges e.g. Wang Brady method. Another group is a group of auto correlate methods. These methods calculate interest points based on moving with the captured scene in different directions. The best-known method of this group is Harris Stephens method, this method is also probably the best-known corner detector at all, and that is why I describe its basics below. There are also some other types of corner detectors, for more information, please refer to [14] [15].

5.1.2.1.1.1 *Harris corner detector*

This detector works with the movement of the captured scene in different directions and calculates gradient in every single point of this scene. The disadvantage of this method are light illusions like crossings between the object of interest and its shadow and noise in the scene. Noise can be reduced by Gaussian filter. This method better detects L-corners than T-corners, that is apparent from the procedure below.

This method starts with the calculation of the auto correlate matrix for all points in the captured scene. This matrix is called Hess matrix of the second partial derivation. This matrix looks like the formula (1) if all second partial derivations exist.

$$A = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x \, I_y \rangle \\ \langle I_x \, I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{1}$$

Now we can calculate eigenvalues by the following formula (2) to distinguish if the actual point is a corner or not. This approach is time-consuming because of the square root operation. So Harris and Stephens invent the new faster formula which can detect if the actual point is a corner or not.

$$\lambda_{1,2} = \frac{\langle I_x^2 \rangle + \langle I_y^2 \rangle \pm \sqrt{\left(\langle I_x^2 \rangle - \langle I_y^2 \rangle\right)^2 + 4 * \langle I_x \, I_y \rangle * \langle I_x \, I_y \rangle}}{2} \tag{2}$$

$$M = Det(A) - K * Trace^2(A) \tag{3}$$

The formula (3) is composed of the determinant of the matrix of second derivations and squared sum of elements on the main diagonal. The formula (3) contains parameter K, which is settable for optimal results. So we can rewrite part of this formula to formulas (4), (5).

$$Det(A) = \lambda_1 \lambda_2 \tag{4}$$

$$Trace(A) = \lambda_1 + \lambda_2 \tag{5}$$

After calculation of M, we can distinguish if the actual point is a corner or not. If calculated value is bigger than defined threshold, point is a corner and vice versa. For more info, please refer to [13]

An example of using Harris detector, and one other corner detector, on the captured scene with the patient monitor is shown in the Figure 4. This operation was done in Matlab [16]. On the captured image was applied Gaussian filter, grayscale transformation and locally adaptive black and white thresholding. I also try a different setup for corner detectors, but this one gives best results. The test was done on the image with resolution 2633px x 2520px, test was also tried on downscaled image, but results were pretty the same.

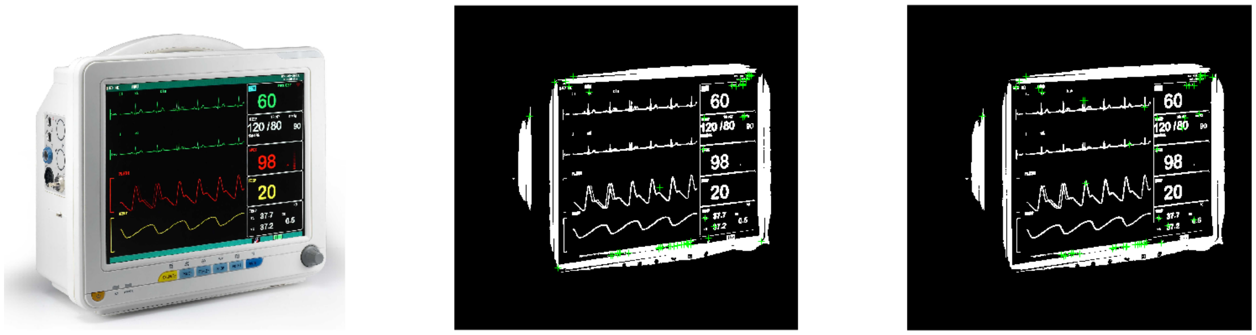


Figure 4 First figure is original image of patient monitor; second is a preprocessed patient monitor with founded corners by Harris-Stephens algorithm; on the last one is a preprocessed patient monitor with foudned corners by MinEigenFeautures [80]

As you can see in the Figure 4 results are for two different corner detectors almost the same, but two bottom corners of the screen are missing in both of them.

5.1.2.1.2 Hough transformation

This method is used to detect lines or objects composed by lines in a captured scene. It also can be used to detect other shapes like circles and ellipse, but in our case, we do not need this possibility. At all Hough transformation is a nice option how to segment objects which have simple borders. The big advantage of this method is the possibility of using it also in cases when borders are not solid and regular. That means this method is relatively robust.

Every single line in two dimension space can be described by the formula (6). Where *r* is the angle of the line and θ is distance of the line to the beginning of the coordinate system. All lines can be represented in this form when ($\theta \in (0, 180)$ AND $r \in R$) OR ($\theta \in (0, 360)$ AND $r \geq 0$ [17]. That means, every single line in the original image can be described as a combination of two values *r* and θ.

$$r = x * \cos(\theta) + y * \sin(\theta) \tag{6}$$

Before using of Hough transformation, it is necessary to apply some edge detector on the image. For more about image edge detection techniques please refer to [18].

Once when we have detected edges, we can use transformation from <x, y> space to <r, θ> (Hough) space for every single point in the image after edge detection. As I mentioned before, every single line can be described as a point in a Hough space. So, we test all the possible lines for all points in the image in the Hough space. For example when we have two points in the image transformed to Hough space with their all possible lines, we can detect the line which connect these two points by intersection of these curves in the Hough space. This is best seen in the Figure 5.

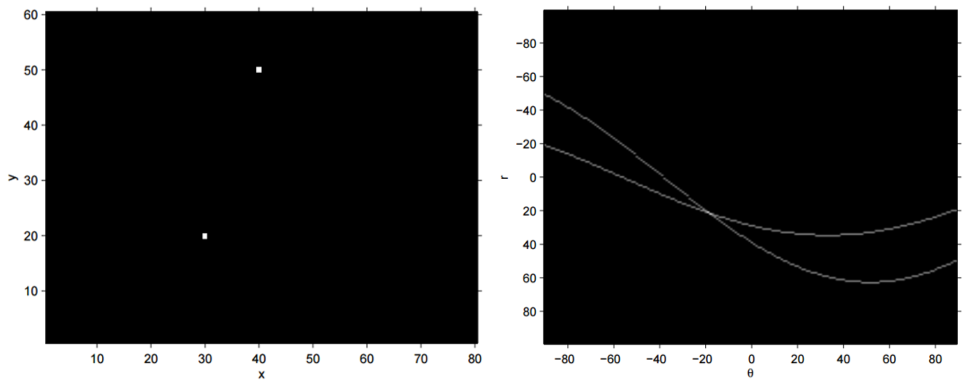


Figure 5 Left: two points in <x, y> space; Right: all possible lines in Hough space <r, θ> through two points on the left. Taken from [17]

In the regular image, after edge detection, we test all possible lines for all detected points. The lines in the original image are then detected if the accumulation of curves in point in the Hough space is bigger than the set threshold. It is important to mention that the Hough space is discrete, so if the resolution is 1 degree for θ and 1 pixel for r the accumulation will be smaller than when the resolution is 10 degrees for θ and 10 pixels for r. On the other hand, with the smaller resolution the precision of a line detection decreases.

When we detect few most important lines in the image, we can simply find their intersections in the original image. These intersections should be the corners of the patient monitor screen. In the Figure 6 is shown the process of the Hough transform on the image with the patient monitor.

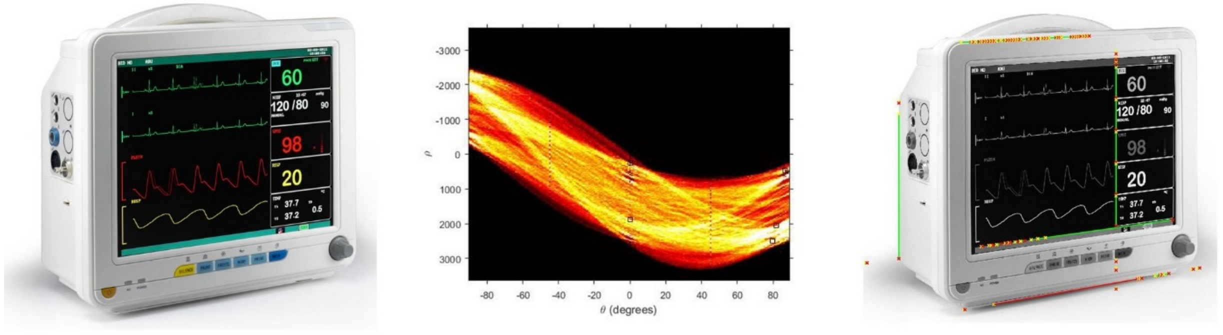


Figure 6 Original image; all possible rotation of lines in points from canny detector in Hough space; in third image you can see only line segments above threshold from Hough space (lines was reduced to line segments by a function in Matlab for a better presentation [80]

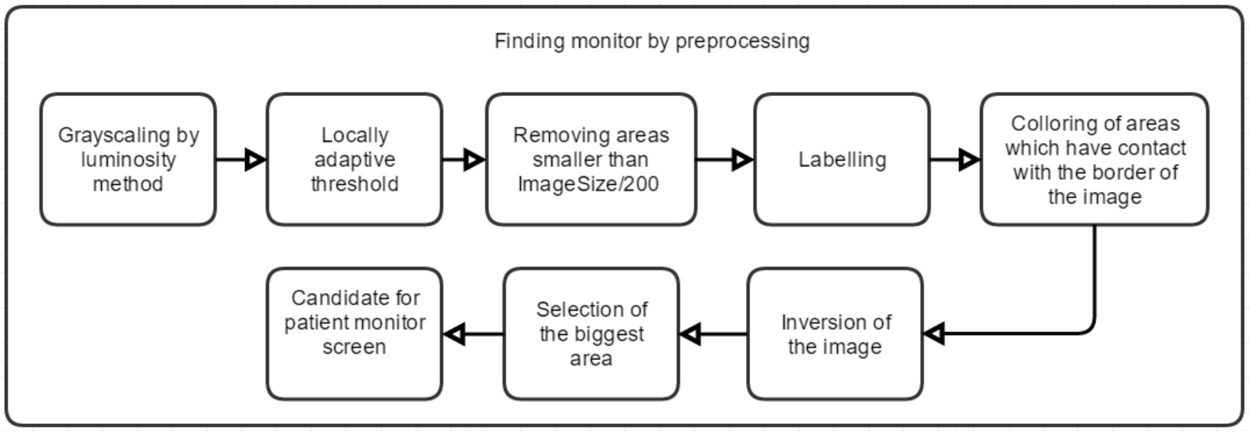
Hough transformation was tested in Matlab and lately also by an EmguCv library in C#. On the original image was applied gray scaling, Canny edge detector and function houghLines. The whole process how to find Hough lines in the image in Matlab is decribed in [16].

5.1.2.1.3 Combination of Hough transformation and corner detector

When we have candidates for corners and most significant lines in the captured scene, we can try to combine these two approaches. That means we have candidates for corners of the patient monitor screen and also we have candidates for borders of the screen. So when a candidate for a corner is lying on two candidates for borders of the patient monitor screen (lines), it is probably the real corner of the patient monitor screen. This process is tested for all intersections of detected lines and its close neighborhood. When the point is located near to this intersection, we can classify it as the corner of the patient monitor screen.

5.1.2.1.4 Finding monitor by preprocessing

There is also a possibility to use a pipeline of operations to extract the patient monitor screen from the captured scene. The whole process is described in the Schema 3. This process was designed by me to keep only the patient monitor screen in the crop of the captured scene. The whole process was designed in Matlab [16].


Finding monitor by preprocessing
Grayscaling by luminosity method
Locally adaptive threshold
Removing areas smaller than ImageSize/200
Labelling
Colloring of areas which have contact with the border of the image
Inversion of the image
Selection of the biggest area
Candidate for patient monitor screen

Schema 3 Pipeline of finding a monitor by preprocessing approach

The first step of this process is gray scaling of the color image. For this operation was used luminosity algorithm which forms the new value by the weighted sum of all three color channels by the formula (7) for every single pixel. Once when we have a grayscale image, we can convert it to black and white image by locally adaptive threshold [19].

$$luminosity = 0.299 * R + 0.587 * G + 0.115 * B \tag{7}$$

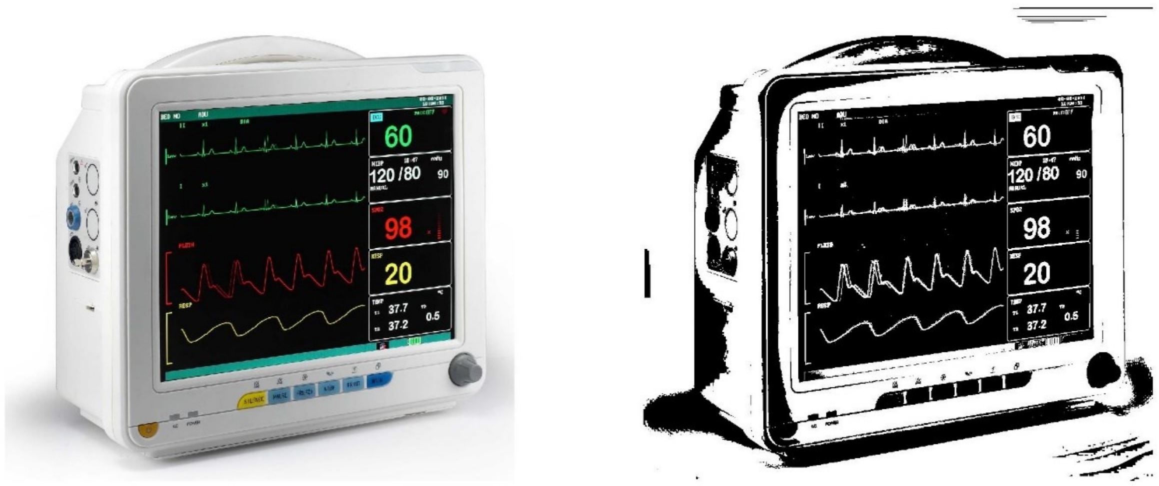


Figure 7 Original image; patient monitor after grayscale and locally adaptive thresholding [80]

Once when we have a black and white image, we can start filtering areas to obtain only the screen of the patient monitor. The first step is the removal of areas with an area smaller than 1/200 of the size of the image. That results in the image without noise. This process always removes areas on the patient monitor screen like curves and symbols.

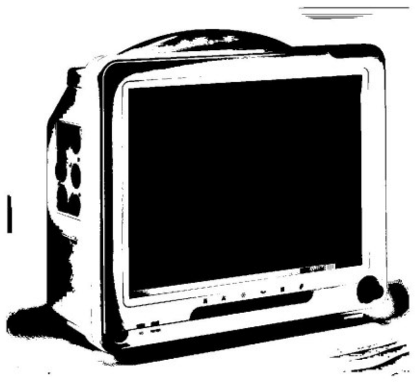


Figure 8 Patient monitor after removing small areas [80]

Next step is the removal of all areas which are in contact with borders of the captured image. Because we in 5.1.1 define the whole monitor has to be captured, there is no possibility that the monitor screen is somehow connected with borders of the image. For this process, it is necessary to mark all areas in the image by connected-component labeling [20]. All labels of areas which are in contact with borders of the image are stored in the list. After that, we can go through the whole image, and if the label of the actual pixel is mentioned in the list, we recolor this area to white. Next step is an inversion of the image. Now we have only few areas in the image. To select the patient monitor screen, we select the biggest area.

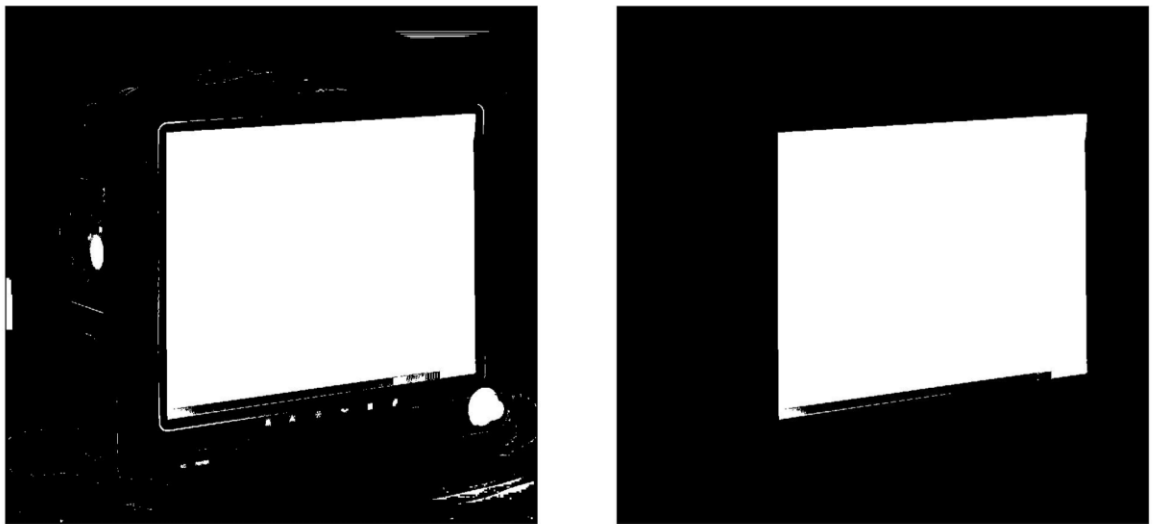


Figure 9 Patient monitor after recoloring all white areas which are in contact with border of image; Selected biggest white area [80]

When we have only one area in the image, we can detect corners by coordinates in the image. The first step is the detection of all positions of pixels in that area. Now we can find where sums of differences of the coordinates in shape are maximized and minimalized by the corners defined as *[1 1; -1 -1; 1 -1; -1 1].* These points are corners of this area. This process is completely done in Matlab [16].

5.1.2.1.5 Definition by user

Corners of the patient monitor screen can also be defined by the user. The advantage of this approach is the precise definition of corners because they are easily recognizable by a human. However, there is a big disadvantage. It is an impossibility to move with the monitor. Whenever someone moves with the patient monitor, the user has to click new corners of the patient monitor in the captured scene. So that means we need a special personal who do this over the whole process of the data acquisition.

5.1.2.2 Projective correction

All objects captured by the camera are distorted. That phenomenon occurs because the camera position in space does not fit with requirement mentioned in 5.1.1 almost all cases. Because we want to detect the location of regions of interest on the screen from every possible angle and distance of the camera, it is necessary to remove this distortion.

For projective correction, it is really useful to use homography, sometimes called as projective transformation. Homography is and an invertible mapping of points and lines on projective plane P2, this is the plane of the camera. That means every 2D point *(x, y)* can be represented as a 3D vector *(x1, x2, x3)* where *x = x1/x3* and *y = x2/x3*. Algebraic definition of homography based on [21] was given by the following proven theorem: *A mapping from P2 ! P2 is a projectivity if and only if there exists a non-singular 3x3 matrix H such that for any point in P2 represented by vector x it is true that its mapped point equals Hx.* That means if we want to sufficiently map each point *x* to the corresponding point *x'* we need to calculate the correspond 3x3 homography matrix. The formula for this fact is shown in the formula (8) where *M'* are homogenous coordinates of a point in the non-distorted image, *M* are homogenous coordinates of a point in distorted image, α is optional non-negative constant and *H* is a homography matrix. When we apply a inverse homography matrix on every distorted pixel we obtain the front-parallel image.

$$\alpha * M' = H * M \tag{8}$$

The homography matrix is composed of geometric transformations. Ordinary the homography matrix is decomposed in three separated matrixes. As you can see in formula (9).

$$H = \ H_S H_A H_P \tag{9}$$

H_S represents a similarity transformation, H_A represents affinity transformation and H_P projectivity. This whole process of the homogenous matrix decomposition is detaily described in [22].

The homogenous matrix contains eight degrees of freedom. All these elements which represent these degrees of freedom are independent. The last element of the homogenous matrix can be described as a scaling factor, that means it is possible to multiply the homogenous matrix by an arbitrary non-zero constant and projective transform still be the same. That means it is necessary to find these eight independent elements.

As you can see from simplified formula (10) without α, and with the knowledge mentioned above we need four points in original image and four corresponding points in a front-parallel image. First four points in the original image was detected by method mentioned in 5.1.2.1. Four points in the front-parallel image can be obtained in the real world, because we need only the aspect ratio of the screen. Size of the screen is not a big problem, because only change is a scale factor in the homogenous matrix. The second alternative how to obtain front-parallel corners is by calculation. For example the simplest one is an average height of patient monitor screen in the captured scene and the same for width. That's give us at least basic information about aspect ratio of the actual screen.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{10}$$

Once when we have exactly four points and their correspondences we can rewrite mentioned formula (10) in two equations for every point as you can see in formula (11). Following equations (12), (13), (14) are only rewriting of the equation (11).

$$x' = \frac{h_{11}x + \ h_{12}y + \ h_{13}}{h_{31}x + \ h_{32}y + \ h_{33}}; y' = \frac{h_{21}x + \ h_{22}y + \ h_{23}}{h_{31}x + \ h_{32}y + \ h_{33}} \tag{11}$$

$$x'(h_{31}x + \ h_{32}y + \ h_{33}) = \ h_{11}x + h_{12}y + \ h_{13} \tag{12}$$

$$y'(h_{31}x + \ h_{32}y + \ h_{33}) = \ h_{21}x + h_{22}y + \ h_{23} \tag{13}$$

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{pmatrix} h = 0 \tag{14}$$

Once when we have equations for all points, we can rewrite them into the matrix which has eight lines. Now we can rewrite the whole equation of homogenous matrix to formula (15) where *h = (h11, h12, h13, h21, h22, h23, h31, h32, h33)*. [23]

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 & -y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 & -x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 & -y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 & -y'_4 \end{pmatrix} h = 0 \tag{15}$$

Once when we have calculated homogenous matrix we can apply it to any pixel with coordinates *(x, y)* and do the projective correction by formula (16).

$$\begin{bmatrix} x'/\lambda \\ y'/\lambda \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{16}$$

This whole process was tested on the image of the patient monitor. As you can see in the Figure 10, the projective correction works quite well, but it is not absolute. That fact occurs, because corners of the patient monitor screen have not been found exactly right and the real size of the patient monitor screen was calculated from the image by average calculation mentioned above.

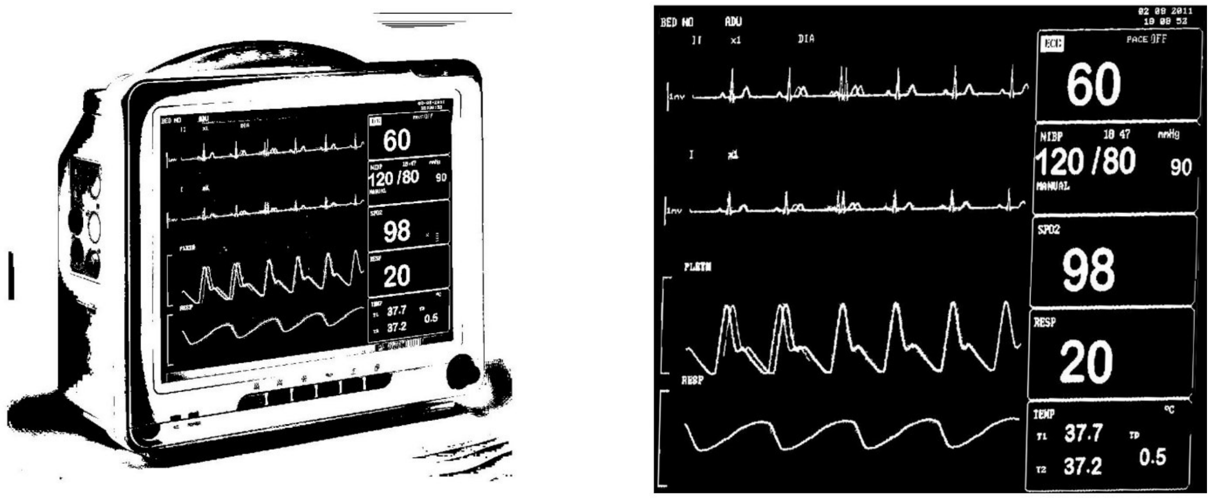


Figure 10 Patient monitor after locally adaptive threshold; projective corrected screen of the patient monitor [80]

5.1.2.3 Selection of region of interest

The patient monitor displays on his screen various types of data. Before we start acquisition process, it is necessary to select which of these data we want to store. Because we already do the projective correction, information will be every time at the same spot independently on the position of the camera.

The possible way how to define regions of interest is via a rectangle selection. As you can see in the Figure 10, the projective correction is done quite good. So, to extract number 60 by the rectangle selection will not be a problem. However, it is necessary to select the region of interest well. For example, if on the monitor would be number 6 instead of 60 during the calibration process and the rectangle selection would be fitted to number with one digit, we will have problems when a number with two or more digits appears in this area. Because of that, it is better to define wider and also little bit higher region of interest, due to an undefined number of digits and also because of small imprecision of a projective correction.

The extension of the mentioned method could be clicking on a shown digit in the selected rectangular selection. Once when we have defined a region of interest, we can mark individual areas inside of it with connected-component labeling like in 6.5. When we click on one of these marked areas we can obtain its height and width, and look inside of the region of interest only for areas with similar characteristics. These characteristics should be updated during the whole acquisition process, because the scale of the digits may vary over time. The second extension could be finding of digits only on the left and right side of the selected digit. The distance on these sides, where other candidates for digits are found, is equal to the width of the firstly selected digit. The definition of the user click can be a little bit tricky during the actual data acquisition process, and I solve it in 6.8.

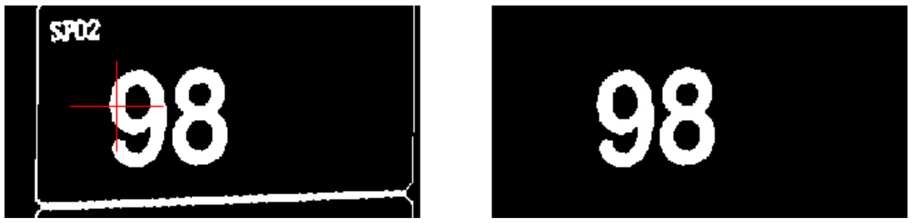


Figure 11 Manual selection of the digit; areas filtered only on appropriate candidates in the distance from selected digit [80]

5.1.2.4 *Classification of selected candidates*

After the selection of the regions of interest, we have candidates for character recognition i.e. classification. The group which deals with these problems is called optical character recognition (OCR). OCR is a process which has as the input pixels of single, or multiple, characters and output is a recognized character. There are several possibilities how to recognize the character. Few of these methods needs to be implemented on your own, but there are also few libraries which deal with this problem. The biggest of them is open source library called Tesseract [11]. This library was used during testing of this approach. The whole topic of OCR is very well researched and people try to recognize characters for a relatively long time [24]. The whole process of classifying is divided into two separated parts: an extraction of features and classifying.

5.1.2.4.1 Feature extraction

The feature extractor derives the features of the character. Features are a set of information which describes the single character. This information should have property, that a feature about one character is different from the feature of the other character. There are several tactics how to obtain these features, and we usually combine many of them for better results. A Bigger number of these features improves the accuracy of a classifier, but it will take a long time to extract all of them. I will introduce few of these features below, for more details, please refer to [25].

5.1.2.4.1.1 *Template matching*

This way how to obtain features is very popular and very simple. The basic one takes one pixel as one part of the feature of a character. During the classification, every pixel is compared with the pixel of learned dataset. If pixels are same, we have a match and vice versa. This approach looks like it has a very high accuracy, but it is not always true. As an example, we can take a letter O in one front and classifier which has learned characters on a different font. In that case, digit 0 can give us a better accuracy than O, but the original letter is O, and that is the problem. Another problem of this method is the high dimension of the feature. Let's have an image with 100x100 pixels. In that case, we have a feature with 10 000 dimensions. That means it could be time-consuming. This method is similar to the convolution of the image with the convolution mask, where convolution mask is newly obtained character and background is a set of characters defined before.

5.1.2.4.1.2 *Structural features*

This method is based on a structural information about the character and decision rules. The structural feature describes the specific properties of the character. Basic properties of characters may be a number of lakes in char, the black-white pixels ratio, the number of corners, etc. There are also more sophisticated features like the distance of the first pixel from the left border, a distance of the last pixel in a row to the right border or number of white-black/black-white crossing in every row. In this case, it is a little bit harder to extract features from a scene, but the process of classifying is faster [26].

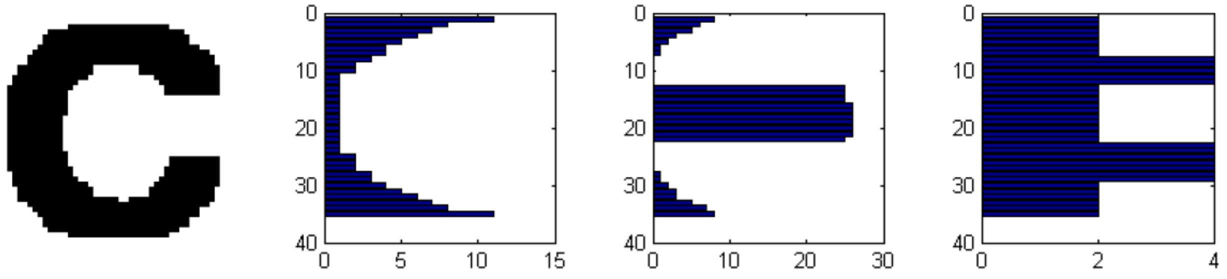


Figure 12 Tested symbol; distance of the first pixel from the left border; distance of the last pixel in row to the right border; number of white-black/black-white crossing in every row [26]

5.1.2.4.2 Classifier

For a classification of digits from the patient monitor, we can choose supervised learning methods or simple convolution which is already mentioned in 5.1.2.4.1.1. Supervised methods use data with known information about its label. For example, we have a figure of a character A and label of that character is information it is A. Besides the information it is A, we also have all extracted features about this character, which we choose.

We know many supervised classification techniques [27]. As I mentioned before in 5.1.2.4 I used the Tesseract library for testing this approach. Overview of Tesseract OCR engine is described in [28]. At following lines, I try to describe methods which look the best for classifying of data from patient monitor.

5.1.2.4.2.1 *Bayesian decision*

Bayesian decision is based on a multivariate Gaussian distribution [29] [30]. Let's consider we have a character described by two-dimensional feature. In that case, we use formula (17) to create a multivariate Gaussian distribution for characters from train data set.

$$\frac{1}{\sqrt{(2\pi)^k|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\,\Sigma^{-1}\,(x\,-\,\mu)\right) \tag{17}$$

When the distribution is built, we can use the posterior probability to choose the best suitable character from learned dataset i.e. distribution for tested char. The best suitable character is a character with the biggest value at the position defined by the feature.

5.1.2.4.2.2 *SVM*

Support vector machine (SVM) is a classification based on hyperplane which divides the space into two separate parts [31]. In the case of the two-dimensional feature vector, the hyperplane is a line. This hyperplane is defined step by step by the training dataset. The advantage of this method are kernel functions. These functions allow us to get features of characters to high dimensions. In high dimensions, it is more suitable to distinguish between two characters. The decision to which class the tested character fall is easily based only on formulas (18), (19) and its result, where one side of the hyperplane is one character and second is the second character.

$$\langle w, x_i\rangle + b \;\geq\; +1 - \,\xi_i, \qquad y_i \;=\; +1 \tag{18}$$

$$\langle w, x_i\rangle + b \;\leq\; -1 + \,\xi_i, \qquad y_i \;=\; -1 \tag{19}$$

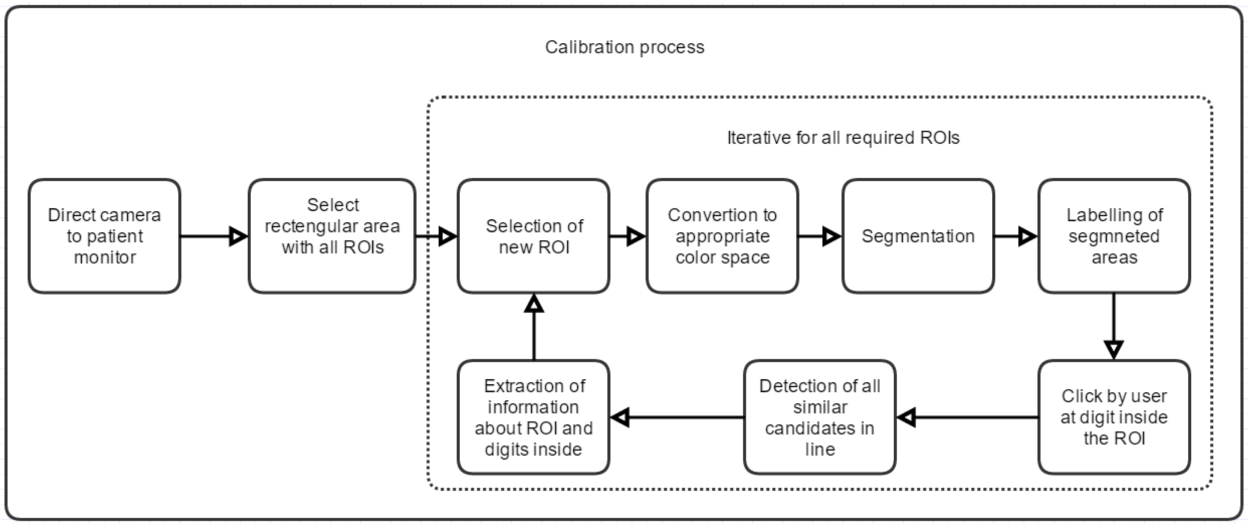
The disadvantage of this method is that SVM is only binary classifier i.e. can distinguish only between two classes. In our case, we need to distinguish between more than two classes. That mean we need to use more complex approaches. Two approaches, how to solve that, are One-vs-rest or all-vs-all, these methods fold in a group called Multi-class SVM [32]. In the first case, we test only one character versus the rest. After testing of all characters at all classifiers, we choose the one with the less lost function. All-vs-all approach needs to build *K*(K-1)/2* classifiers i.e. all possible combinations of characters. After tests of all characters on all classifiers, the one with the most votes win.

5.2 DIGITS BASED APPROACH

The second approach has digits as the main domain, against the patient monitor screen in the previous one. That means it is not necessary to have the whole patient monitor screen in the captured scene. Also, we do not need some light in the room, because when we do not need to find corners of the screen, we do not need distinguishable borders of the patient monitor screen, which is recognizable thanks to light in the room. Digits at the monitor can be described as an own light source, so we can recognize them in almost every light condition in the room. This process also uses less CPU load, because we use image operations only in the small regions of interest. Also, it is not necessary to detect a patient monitor screen in the whole scene whenever we want to capture new frame.

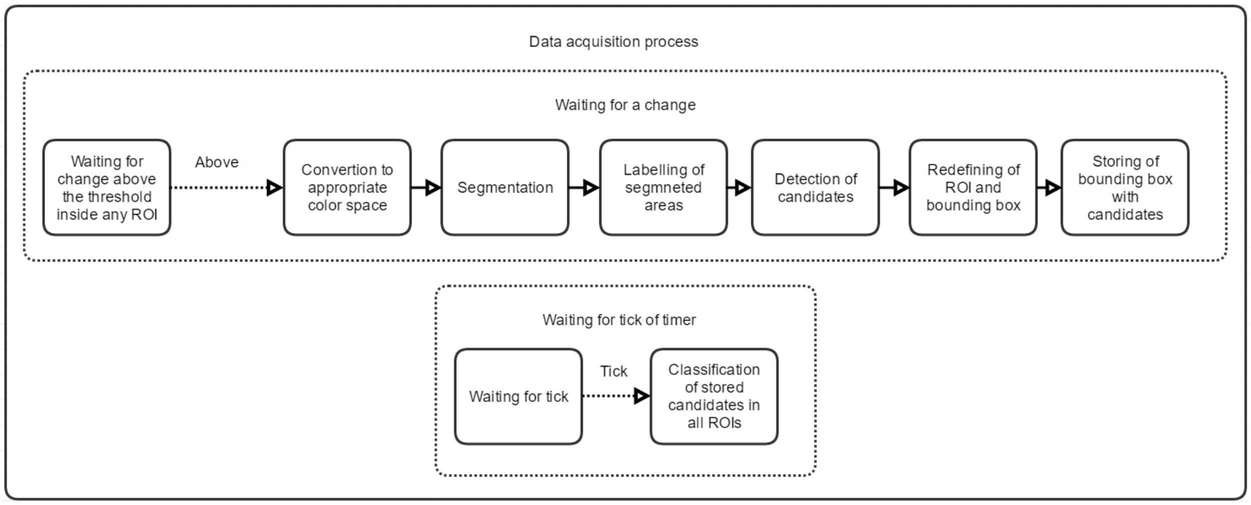
The whole process of this approach is separated into two main parts. The first of them is calibration before the acquisition process and the second one is actual data acquisition process.

The first step of calibration is the selection with all requested information on the patient monitor. Once when we select this selection, we can select the regions of interest with required data. On all of these regions of interest, we apply preprocessing followed by segmentation. All segments are labeled, and user of the software select the required data i.e. digit, in these regions of interest by his own. Afterward candidates for digits are selected. Once when all these steps are done, we can obtain characteristics about the selected digits in all regions of interest and also characteristics about the region itself, and store them for upcoming data acquisition process. The whole pipeline of the calibration process is shown in the Schema 4.


Calibration process
Iterative for all required ROIs
Direct camera to patient monitor
Select rectengular area with all ROIs
Selection of new ROI
Convertion to appropriate color space
Segmentation
Labelling of segmneted areas
Click by user at digit inside the ROI
Detection of all similar candidates in line
Extraction of information about ROI and digits inside

Schema 4 Calibration process for digits based approach

When we have all characteristics of selected regions of interest and also about digits in them, we can start the actual data acquisition process. First few steps are very similar to the calibration process, and that are preprocessing and segmentation. After that, we finding a candidate for a digit in the region of interest. When we find it, we try to find close neighbors which together make up a number. All candidates, that mean the first one and also the neighborhood candidates, need to fulfill conditions defined during the calibration process. Once when we successfully find all candidates we can redefine bounding box for all clusters with candidates, new coordinates and size for the region of interest and characteristics about the selected digits for the next iteration of the acquisition process. Once when we have a bounding box with stored candidates we can use OCR on that bounding box and classify the candidates in it. After that, the whole acquisition process starts new iteration if the difference between two frames in a row has difference bigger than the set threshold. Bounding boxes with stored candidates are classified every time after defined period of time i.e. when timer tick. The whole process is shown in Schema 5.


Data acquisition process
Waiting for a change
Waiting for change above the threshold inside any ROI
Above
Convertion to appropriate color space
Segmentation
Labelling of segmneted areas
Detection of candidates
Redefining of ROI and bounding box
Storing of bounding box with candidates
Waiting for tick of timer
Waiting for tick
Tick
Classification of stored candidates in all ROIs

Schema 5 Data acquisition process for digits based approach

This approach is the selected one to implement as a standalone application. Because of that, every single step and its possible alternatives will be described in separated chapter in 6.

There is also a beta backup process which is used if the process cannot find digits inside the regions of interest or after a defined period of time. This process starts during the calibration where we stored information about the digits in the region of interest and try to find all lines of the text with these characteristics in the whole scene. When we have all these text lines, we can determine its position from the left border of the captured scene and also from the top and save this information to an appripriate region of interest. Once when these definitions are done for all regions of interest, we can start the acquisition process. If no text is detected in the region of interest for the long period of time, or every time after a defined period of time, it probably means that the monitor was moved while the camera was covered by something. At this time beta backup process starts and same as during the calibration process, tries to find all lines of the text in the scene based on stored characteristics of regions of interest. Because we know position from the left and position from the top, we can choose the correct one line in the captured scene, and that is our required region of interest.

5.2.1 Requirements

Five requirements of this approach are exactly same as requirements for the monitor based approach mentioned in 5.1.1. That are following requirements: captured plane of the camera should be as most as possible parallel plane of the patient monitor screen, monitor must be turned on during the process of calibration, symbols on patient monitor should be highly contrasted against the background of it, fonts shown on patient monitor are very similar to synthetic fonts and camera used for capturing have the adequate resolution. All these requirements are described in detail in 5.1.1. in last five paragraphs.

The extra requirement is that is necessary to have all the required data for upcoming data acquisition in a captured scene by a camera. So that means, we need to have all required regions of interest in the captured scene for the whole time when the data acquisition process is in progress.

6 Analysis of digits based approach

In this chapter, I describe every single step of the whole pipeline of a digits based approach. In every subchapter, I mention which method I use in the standalone application and mention if there are any alternatives.

6.1 Selection of area of interest

Nowadays cameras have a various kind of resolution, but the most sold are cameras with FullHD resolution i.e. *1920px X 1080 px*. That means the captured scene contain exactly 2 073 600 pixels. During the ordinary operation with the captured scene we have to visit every single pixel at least once, that is 2 073 600 operations in minimal. In a normal case when we capture the scene with the patient monitor, patient monitor usually does not occupy the whole scene.

The left part of the Figure 13 showing the whole captured scene with the simulated patient monitor and that means already mentioned 2 073 600 points. On the right side of the the Figure 13, we can see the selection with all regions of interest with resolution 328px X 419px, and that is 137 432px. As can be easily calculate, it is only 6.62% of the number of pixels in the whole captured scene. This fact helps us in the case of performance of the application. Because of that, it is practical to work only with this selection.

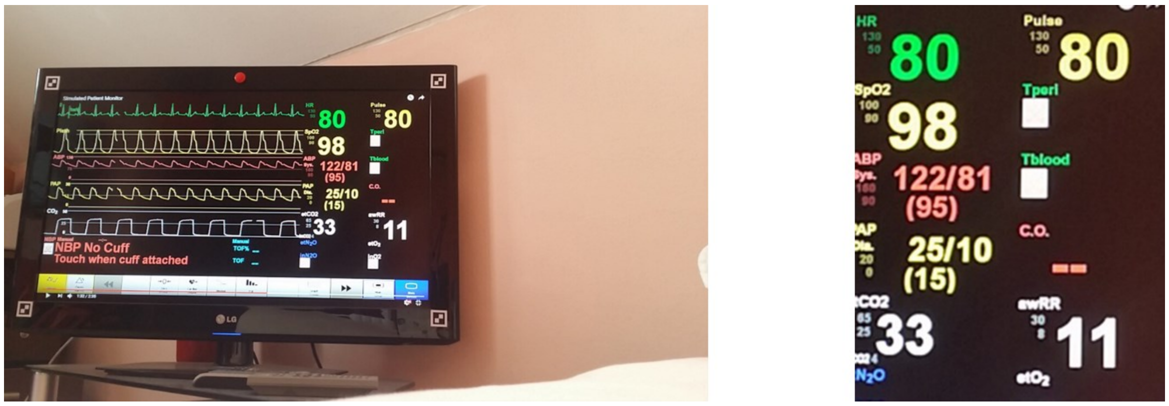


Figure 13 Whole scene with simulated patient monitor; all selected regions of interest

Inside of the software, we need to store the position and the size of this selection. Lately, whenever we want a new frame from the camera, we can crop the image by this selection.

In the implemented application I use this selection only for the selection of smaller regions of interest (ROIs). Once when we have this selection, we can select the individual ROIs with required data. That means if we want to select for example heart rate, pulse, SpO2 and swRR we select every one of these ROIs separately. Between the selection of actual and next ROI with next data application do a bunch of processes on this ROI, these processes are described in following chapters. In the end, the number of selected pixels will be after the whole selection process low. For mentioned ROIs shown in Figure 14 it is equal to the equation (20).

$$(107 * 64) + (115 * 67) + (103 * 62) + (121 * 73) = 29\,772\ px \qquad (20)$$

That is only 1.43% of the number of pixels in the whole captured scene. It is necessary to mention that size of the ROI during the calibration process is defined by the user. Lately during the acquisition process is the size of ROI redefined automatically. This redefinition is described in 6.8.

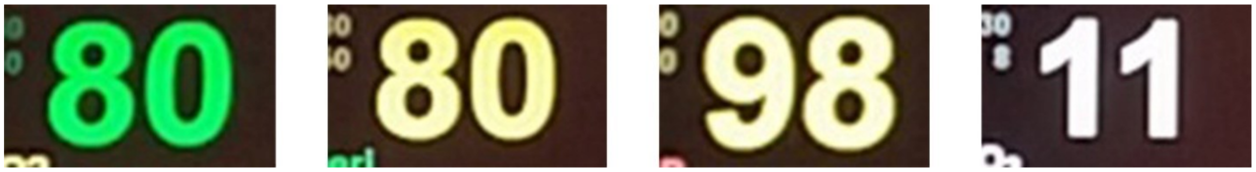


Figure 14 Four selected regions of interest from a patient monitor

6.2 Recalculation of coordinates in program

There is also one problem which is necessary to solve. When we use the application, we want to see the captured scene in the same ratio as it was captured and also as big as it is possible. That means the coordinates on the display of the computer does not correspond to coordinates in the captured scene. Because of that, it is necessary to recalculate the coordinates from the screen to coordinates of the captured scene.

In C#/EmguCv, which I use to implement the software, I used the zoom as a way of displaying of the captured scene. That means there are two possibilities how the captured scene can look. First of them is that imageBox, which is a construct of selected library, has borders on sides of the captured scene. The second one is almost the same, but the imageBox have borders on the top and the bottom of the shown captured scene. These two cases are shown in Figure 15.

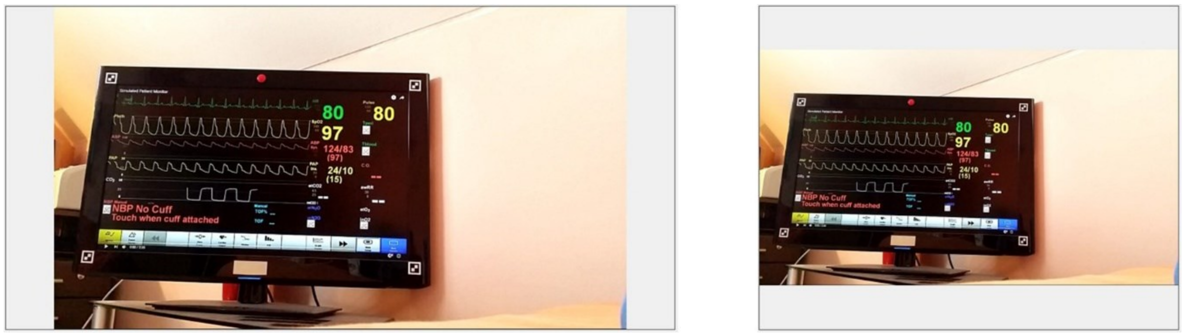


Figure 15 Displaying of captured scene in implemented application: First case borders on sides; Second case borders on the top and bottom

In the first case when borders are on the sides we can recalculate the X and Y coordinates by following formulas (21), (22), (23), (24).

$$Y_{inImage} = Height_{Image} * \frac{Y_{selectedByMouse}}{Height_{ImageBox}} \tag{21}$$

$$scaleWidth = Width_{Image} * \frac{Height_{ImageBox}}{Height_{Image}} \tag{22}$$

$$dx = \frac{(Width_{ImageBox} - scaleWidth)}{2} \tag{23}$$

$$X_{inImage} = \left(X_{selectedByMouse} - dx\right) * \frac{Height_{Image}}{Height_{ImageBox}} \tag{24}$$

For the second case the formulas are very similar and you can see them below in (25), (26), (27), (28).

$$X_{inImage} = \ Width_{Image} * \frac{X_{selectedByMouse}}{Width_{ImageBox}} \tag{25}$$

$$scaleHeight = Height_{Image} * \frac{Width_{ImageBox}}{Width_{Image}} \tag{26}$$

$$dy = \frac{(Height_{ImageBox} - scaleHeight)}{2} \tag{27}$$

$$Y_{inImage} = \left(Y_{selectedByMouse} - dy\right) * \frac{Width_{Image}}{Width_{ImageBox}} \tag{28}$$

6.3 PREPROCESSING

Preprocessing is an essential process before the segmentation. That means we need to select the best preprocessing operation for the best result of the segmentation process. There are several solutions how to do the preprocessing before the segmentation or OCR process [33][34][35]. It is important to mention that this is not classical OCR task, because classical OCR usually works with scanned pages of text and data obtained from the patient monitor sometimes have closer to the detection of the text in the natural scenes. However, this is a bold statement since defining of an ROI location, and its characteristics are defined during calibration.

6.3.1 Color space

When the scene is captured, it has some specific color space. The ordinary camera captures the scene in RGB space which is three-dimensional space with the red, green and blue channel. There are many different color spaces [36]. Between the most known belongs RGB, HSV and grayscale image.

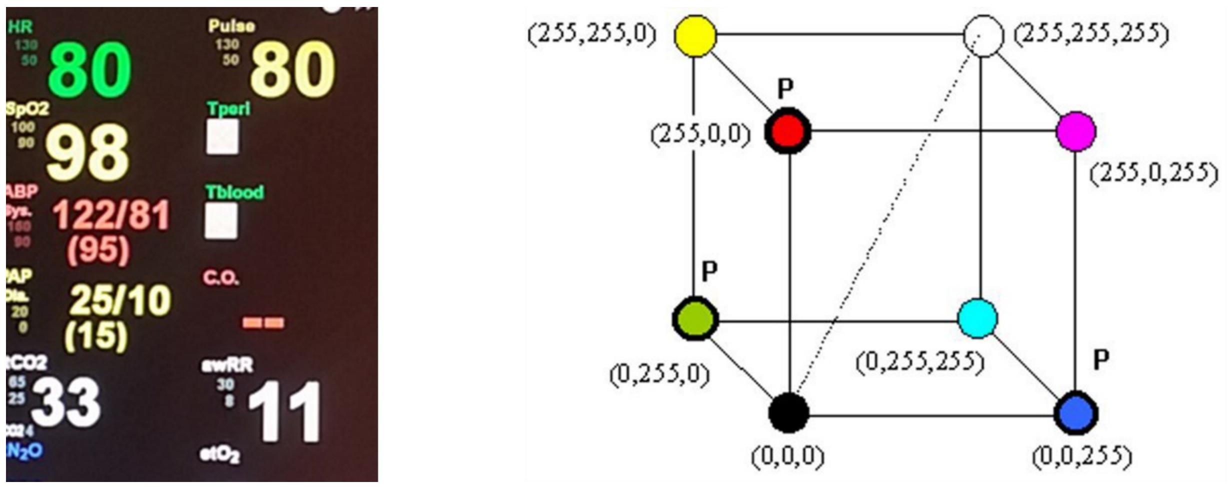


Figure 16 Selected regions of interest in RGB color space; mixing of color channels in RGB space [36]

6.3.1.1 RGB

As I mentioned before, an ordinary scene is captured by the camera in RGB color space. This space has three separate channels where every one of them has a value between 0-255. Zero represent black and with bigger value color become lighter and lighter until the value reaches 255 which is equal to white. Whole RGB space can be represented as a cube in the Figure 16. That means we can decompose this color space in three separate images for every channel. As you can see in the Figure 17 using only one channel of these three channels for segmentation is useless, because some information, for example digit 80 in the top left corner in the red channel, almost completely missing. However in the green channel, all information are quite good readable, but it strictly depends on the color of the digits in the patient monitor, and we cannot count on it. For example, one digit can have a magenta color, and that means with respect to the cube in the Figure 16 that green channel is equal to zero. [37]

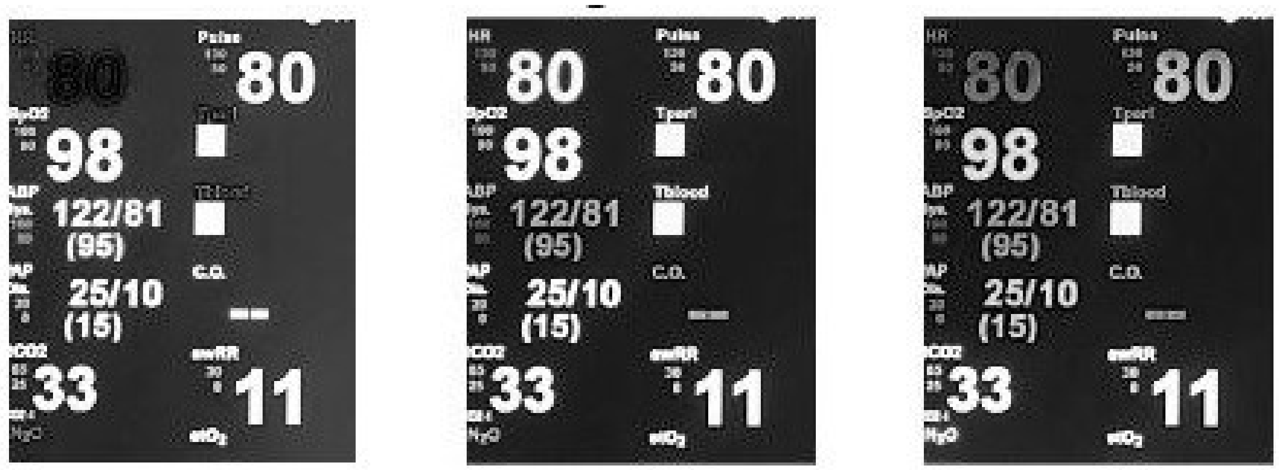


Figure 17 All regions of interest in separated channels for RGB color space: Red; Green; Blue

It is also possible to do segmentation for very channel separately, but this approach consumes at least three times more operations. With this alternative work for example [38] [39].

6.3.1.2 HSV

This color space is quite different against the RGB. Individual letters in its name represent Hue, Saturation and Value. Hue is the color type and represents the own color of the element, it is represented by an angle with a value between 0-360 degrees. This value is ordinary normalized in the range 0-255 where zero is equal to red. Saturation is a representation of the vibrancy of the color. This value is between 0-255. With the lower saturation color becoming more faded. Value channel represents the brightness of the color. Its range is still between 0 and 255. Where zero is fully dark color and 255 is equal to fully bright color. Representation of this color space is cylinder shown in the Figure 18. [40]. Conversion from RGB to HSV is described in [41]

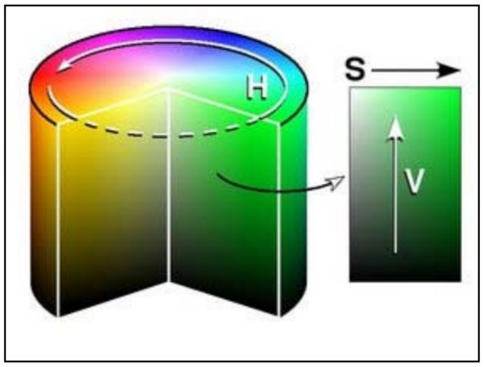


Figure 18 Graphic schema of HSV color space

When we try to decompose image with all regions of interest in these three separated channels, we obtain results shown in the Figure 19. As you can see in hue channel it is almost impossible to, even by a human eye, distinguish between the data and the background. In the saturation channel is situation better, because we can distinguish the data at least by an eye, but for segmentation, it will be problematic because some digits have around yourself white border and some not. Also, almost all of them have a different color. The process of segmentation will be possible, but it is not the easy way with good results. Value channel is really interesting for segmentation because all digits are nicely high contrasted against the black background. When we look in [41] value channel is also the easiest one of these three channels to calculate. It is just a simple maximum value of three channels inside the RGB space. So value channel of HSV space seems to be a nice option, and some papers choose the HSV space as the best solution [42].

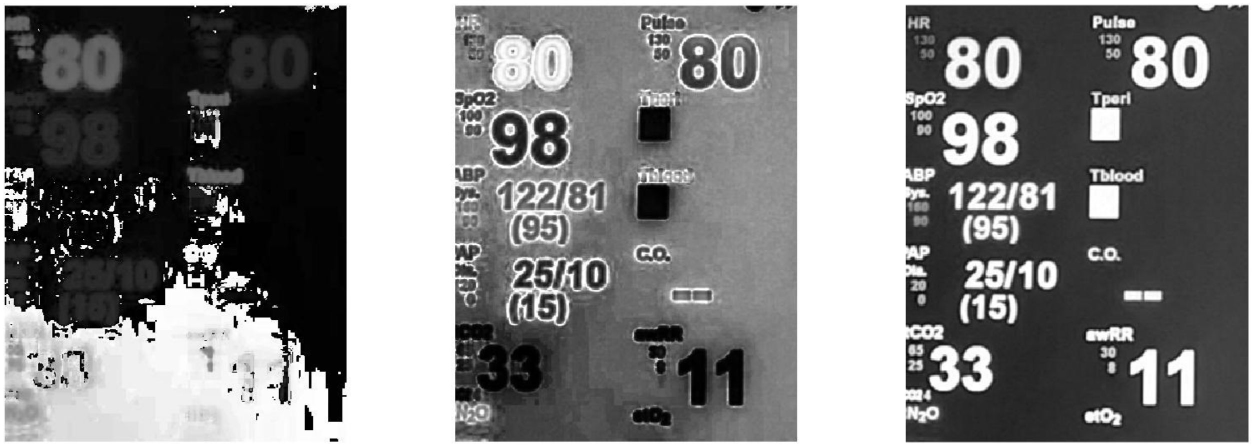


Figure 19 All regions of interest in separated channels for HSV color space: Hue; Saturation; Value

6.3.1.3 Grayscale

Grayscale is a color space with only one channel. There are three basic ways how to compute the new value of the pixel from RGB: lightness, average and luminosity. Lightness works with averaging of max and min value of color channel of the pixel. The average method works with the simple arithmetical mean across all color channels of the pixel. The third one i.e. luminosity is a little bit more sophisticated. It also works with averaging of all color channels, but every single channel is weighted. This weight is based on a number of light sensitive cells for this color in the human eye. Formulas for these conversions are shown in formulas (29), (30), (31). In EmguCV library, which I use during the implementation, is used luminosity method with parameters in the formula (31) [43]. [44]

$$lightness = (\max(R, G, B) + \min(R, G, B))/2 \tag{29}$$

$$average = (R + G + B)/3 \tag{30}$$

$$luminosity = 0.299 * R + 0.587 * G + 0.115 * B \tag{31}$$

In the Figure 20 you can see the difference between the gray scaled image and value channel of HSV space. As you can see, little bit better results give us the value channel from HSV space. However, because EmguCV works inner easier with gray scaled image and digits in the grayscale image are still highly contrasted against the background, I select the grayscale color space for a implementation.

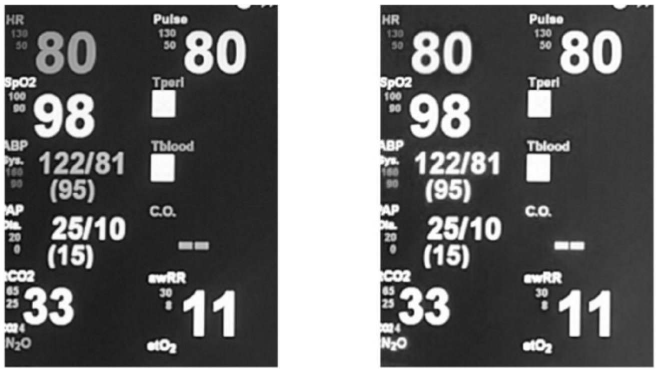


Figure 20 All regions of interest in grayscale image by luminosity; All regions of interest in value channel from HSV space;

6.3.2 Histogram equalization

In converted grayscale picture values of pixels can be from range 0-255. Ordinary an obtained grayscale image does not contain all of these values, and that is the shame. A number of pixels in gray levels can be nicely shown by the histogram of the image in the Figure 21. As you can see in the histogram of this image, image contain only very few values of gray level, and the biggest clusters are relatively distant.

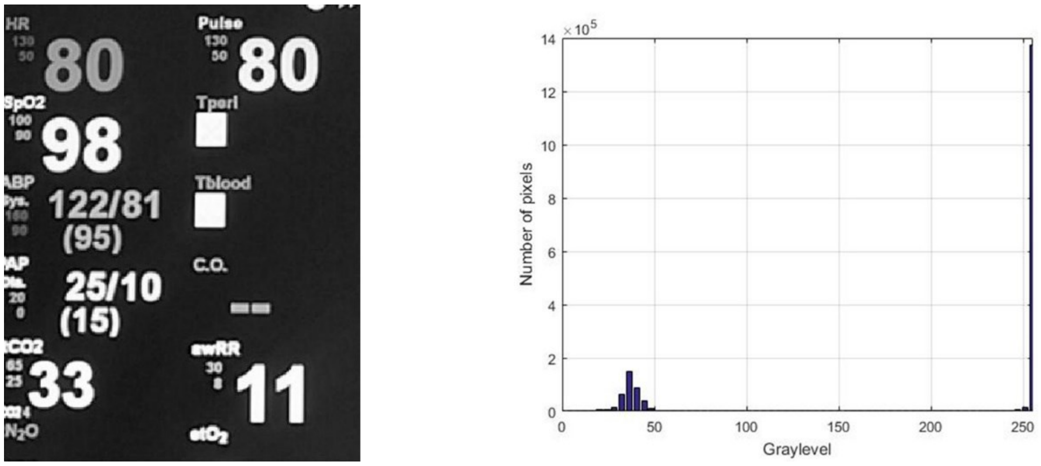
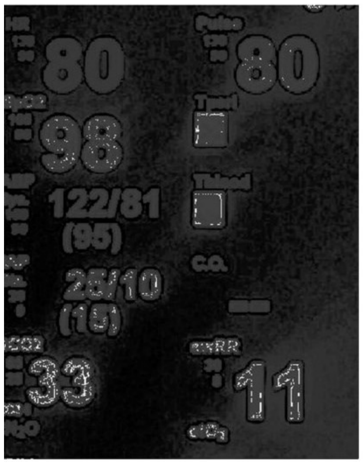


Figure 21 All regions of interest in grayscale; histogram of selection on the left

If we want to use the whole potential of the grayscale color space we can use histogram equalization method. That means we try to spread pixels values over as many pixels values for a given color space as it is possible. For more about this method, please refer to [45]. As you can see in the Figure 22 digits which represent the cluster around the value 45 in the Figure 21 is redistributed in the neighborhood. However, in the image after applying the histogram equalization process is nicely shown that numbers are hardly distinguishable against the background. That means histogram equalization is a bad method before a segmentation process and two clusters with the high distance between them are welcome.



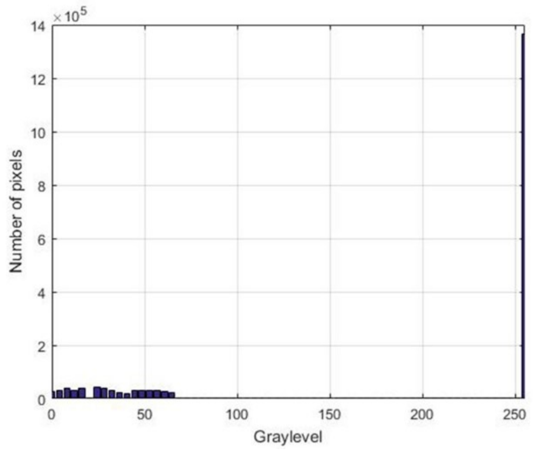


Figure 22 Figure 23 All regions of interest after histogram equalization; histogram of selection on the left

6.3.3 Noise reduction

The patient monitor is very specific. The biggest specific of the monitor is one colored background. This provides us an almost binarized image. Problems of the scene with the patient monitor are artifacts which occur when the scene is captured. There are several artifacts which can make the classification worse. There are two basic types of artifacts or noise. The first one is random i.e. independent. This noise is caused by screeches on lens, impurity or quality of CCD sensor. Another type of noise is dependent. This noise is called Gaussian and it rises systematically depend on something in the scene. Another source of artifacts are sources of light in the scene and turning the image into the grayscale space, which is related to information reduction. We have two types of noise reduction filters i.e. linear and nonlinear.

Grayscale image of patient monitor ordinary does not contain so much noise, because of that only a little briefing through possible solutions will be mentioned.

6.3.3.1 Linear filters

These filters meet the principle of superposition [46]. Usually, before the actual filtration, a scene is projected to the other space. Most often the projection is provided by the fast Fourier transformation which gets an image to space of frequencies [47].

One of the most used filters is low pass filter. This filter exploits the fact that noise in the image always has high frequencies. After application of this filter, high frequencies are removed and after the reverse Fourier transformation, we have a scene without the high-frequency noise. This filter can also be used without the projection to frequency space by convolution of the scene with a convolution matrix [48].

Another favorite filter is Gaussian filter [48]. This filter works in an image space and used convolution mask to filter the scene. This mask is always square and has a structure of Gaussian distribution i.e. the middle part of the mask has the biggest value and more closer to the edge we are, the lower values are on the mask. The negative of this approach is the blurred image after its application. This trend is shown in the Figure 24. That fact is undesirable because digits should have nicely distinguishable borders for segmentation and also for OCR process.

Averaging is a next suitable method of linear filtering [48]. This process is very similar to Gaussian filtering. We use the mask which we put on the current pixel. The new value of the current pixel is the average value of all pixels which are covered by the mask. This filter also makes the image little bit blurred as you can see in the Figure 24.

6.3.3.2 *Nonlinear filters*

In nonlinear filtering does not work the principle of the superposition. The first type of nonlinear filtering is a median filter [49]. This filter works almost the same as mentioned average filter, but the new value for the current pixel is the median of pixels covered by the mask. This method is more robust than an average filter because it does not count with outliers. This filter looks like the best solution, but as I already mention before, using of filters is not necessary in our case.

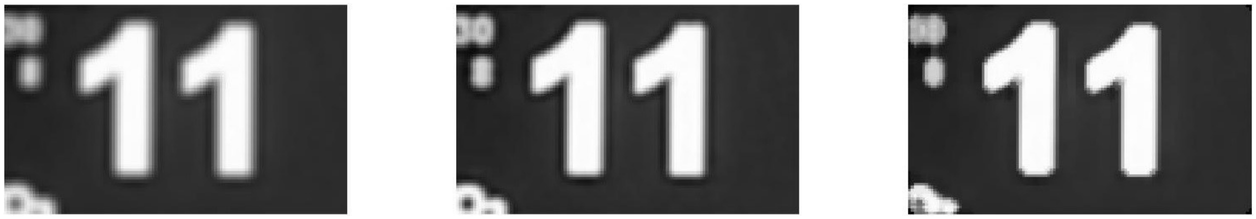


Figure 24 Region of interest after applying filters with kernel 5x5 px. Gaussian; average; median

6.4 SEGMENTATION

For segmentation of the image i.e. distinguishing between the background and the foreground of the image, we can use one of the four groups of segmentation methods which are: threshold based segmentation, edge based segmentation, region based segmentation, matching and clustering techniques. If you want to check the list of various segmentation techniques, please refer to [50]. Because we already have almost black and white image, the best solution seems to be one of thresholding methods. One little bit advanced method will be mentioned, and it is MSER.

6.4.1 MSER

One of the most favorite methods is called maximally stable extremal regions (MSER) [51]. This method is ordinary used to detection of text in real scenes [26]. It can also be used for detection of traffic signs or other things in the captured scene [52]. The principle of detection of MSER can be described as follows. A grayscale image is gradually thresholding with all possible thresholds. Pixels below the threshold are colored to black and pixels above the threshold are colored to white.

If you imagine series of thresholding planes with the value begins at 0 at ends with 255 as a video. The video starts as a completely white image. With a higher threshold, black areas start to appear and begin to grow until the threshold reaches value 255. In that case, the image is completely black. All closed areas which appear during the this process are maximal regions. Stable regions are moreover stable with a high range of threshold values. For a formal description of MSER please refer to [51].

These MSER have some useful properties. *Invariance to affine transformation of image intensities. Covariance to adjacency preserving (continuous) transformation $T : D \rightarrow D$ on the image domain. Stability, since only extremal regions whose support is virtually unchanged over a range of thresholds is selected. Multi-scale detection. Since no smoothing is involved, both very fine and very* *large structure is detected. The set of all extremal regions can be enumerated in O(n log log n), where n is the number of pixels in the image.[49]*

As you can see in the Figure 25 all digits were nicely founded by this method. However, this method seems to be unnecessarily complicated in EmguCV, although it is one of the greatest alternative how to deal with a segmentation process. The Figure 25 was made by a function detectMSERFeatures in Matlab [53].

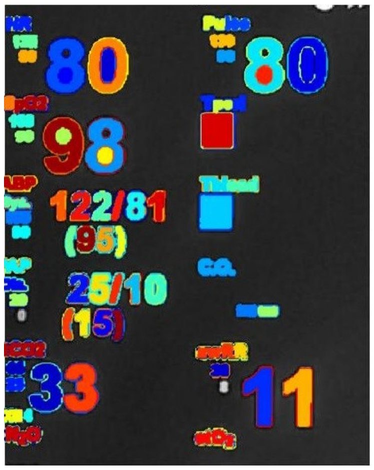


Figure 25 Segmented input image in separated segments by MSER method

6.4.2 Global thresholding

Global thresholding is the easiest way how to separate the image into the foreground and the background. It can be described as one comparison with a threshold plane as well as in MSER mentioned in 6.4.1. Pixels below the threshold are colored to black and pixels above the threshold are colored to white.

The threshold value can be chosen by a user or calculated from the image. The first option is useful for scenes with the similar light condition for the whole acquisition process. Because the conditions can be changed during the acquisition process, it is better to choose some more sophisticated method for selecting this value. The optimal value for thresholding is based on the histogram of the image. There are several techniques how to find the threshold value [55].

One of the most favorite techniques is Otsu method [56]. In this method, we look at the points like a two separated clusters with a diverse range of values. Ordinary these two ranges overlap each other. The main goal of this method is to find the threshold where the sum of the foreground and background ranges is at its minimum. This method is iterative, and for every possible threshold value do these steps:

1) Separate all pixels into two groups separated by threshold value

2) Calculate the mean in both groups

3) Difference of means square by factor two

4) Multiplying by the number of pixels in each group

Result: Threshold value is that which has max value $\sigma_b^2(T)$

These steps can be described in term of formulas (32-36)

$$\sigma_b^2(T) = n_b(T)n_0(T)[\mu_b(T) - \mu_0(T)]^2 \tag{32}$$

$$n_b(T) = \sum_{i=0}^{T-1} p(i) \tag{33}$$

$$n_0(T) = \sum_{i=T}^{N-1} p(i) \tag{34}$$

$$\mu_b(T) = \sum_{i=0}^{T-1} \frac{ip(i)}{n_b(t)} \tag{35}$$

$$\mu_0(T) = \sum_{i=T}^{N-1} \frac{ip(i)}{n_0(t)} \tag{36}$$

This process is relatively high time consuming, so there is a possibility to update only *n* a μ in the next iteration of the process when we know how histogram looks. How to calculate the next iteration is shown in formulas (37-40) where n_T the frequency of intensity is equal to the threshold value.

$$n_b(T+1) = n_b(T) + n_T \tag{37}$$

$$n_0(T+1) = n_b(T) - n_T \tag{38}$$

$$\mu_b(T+1) = \frac{\mu_b(T)n_b(T) + n_T T}{n_b(T+1)} \tag{39}$$

$$\mu_0(T+1) = \frac{\mu_0(T)n_0(T) + n_T T}{n_0(T+1)} \tag{40}$$

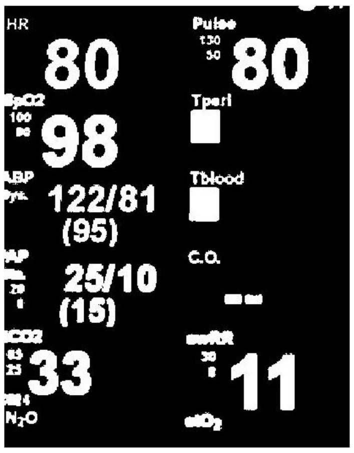


Figure 26 All selected regions of interest after binarization by Otsu method with a level set to 0.4902

As it is shown in the Figure 26, Otsu method gives a nice result. This method seems to be almost the best solution for the further implementation. However, there is a problem with EmguCV which crashes whenever I try to use Otsu binarization method, many people at official forums of EmguCV also notice this. I hope it will be fixed in upcoming version. Because of that I use Bradley + Wellner methods which is mentioned below in 6.4.3.

6.4.3 Locally adaptive threshold

The locally adaptive threshold is similar to the classic global thresholding. Pixels below the threshold are colored to black and pixels above the threshold are colored to white. The big difference is that threshold for every single pixel is computed separately from his neighborhood. For this purpose is used Bradley local image thresholding method [19]. This method is more robust to illumination changes in the image than a classic global thresholding. That is a very pleasant fact because we are capturing data from a patient monitor screen where reflections can occur.

This method using integral images. This integral image is a neighborhood according to which decides whether the resulting pixel will be black or white. First of all, we need to compute the sum of grayscale level over a rectangular region of the image. *Without an integral image, the sum can be computed in linear time per rectangle by calculating the value of the function for each pixel individually. However, if we need to compute the sum over multiple overlapping rectangular windows, we can use an integral image and achieve a constant number of operations per rectangle with only a linear amount of preprocessing. [19].*

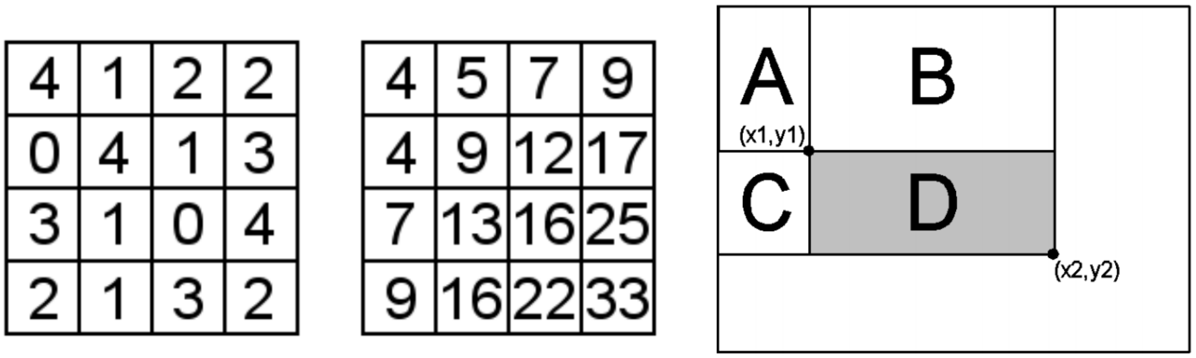
For this purpose we create a new matrix *I* where on position <x, y> will be the sum of all grayscale levels to the left and above the pixel <x, y> from the original grayscale image. This can be calculated by dynamic programming by the formula (41). The process of calculation is shown in the Figure 27.

$$I(x,y) = f(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1) \qquad (41)$$

Once when we have the whole matrix, we can simply calculate the sum of grayscale levels in any rectangular selection between <x1, y1> (left-top corner) and <x2, y2> (right-bottom corner) by following calculation (42) from the matrix *I*.

$$\sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} f(x,y) = I(x_2,y_2) - I(x_2,y_1-1) - I(x_1-1,y_2) + I(x_1-1,y_1-1) \tag{42}$$

The whole process of calculation of sums is nicely shown in the following Figure 27.



4	1	2	2
0	4	1	3
3	1	0	4
2	1	3	2

4	5	7	9
4	9	12	17
7	13	16	25
9	16	22	33

Figure 27 The integral image. A simple input of image values; the computed integral image; using the integral image to calculate the sum over rectangle D [19]

If we have the whole matrix *I* and also the rectangle selection we can use the Wellner's method now [57]. *The main idea in Wellner's algorithm is that each pixel is compared to an average of the surrounding pixels. Specifically, an approximate moving average of the last s pixels seen is calculated while traversing the image. If the value of the current pixel is t percent lower than the average then it is set to black, otherwise it is set to white. [57]* In modern systems like a Matlab [16] or EmguCV [58] we can use the different function above the rectangular selection instead of the average. It could be Gaussian function or median. For this method is essential size of the neighborhood.

In my application, I choose Bradley + Wellner method. As a function above the rectangular selection was selected Gaussian function with size of neighborhood equal to 101. This value was selected for its best results during the capturing of the video with camera resolution 1280px X 720px. As you can see in the following Figure 28 difference of average (mean) or Gaussian function is almost unrecognizable. When we look at the median function, the noise around the digits is undesirable for upcoming areas labeling.

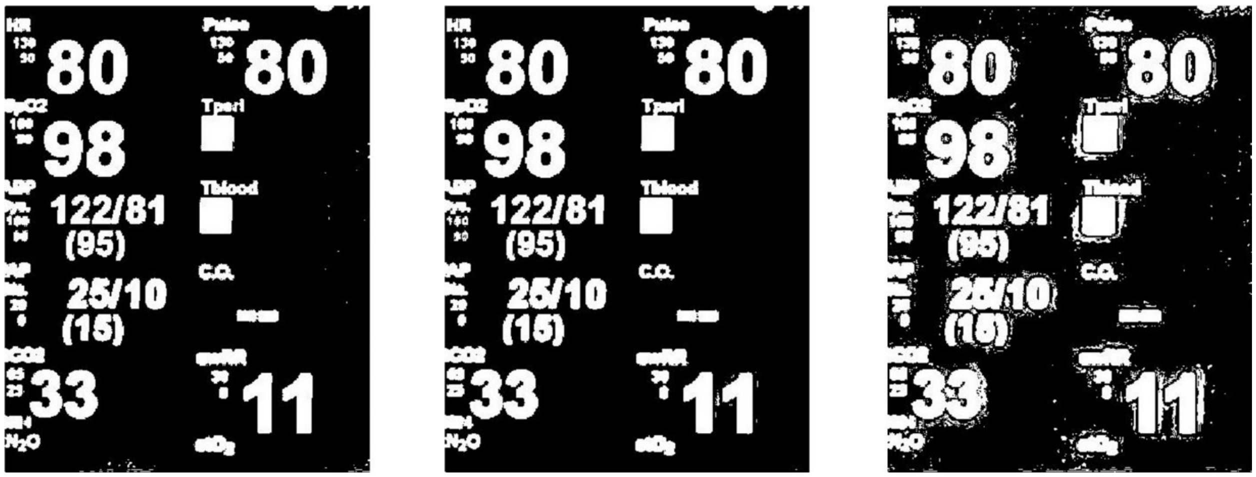


Figure 28 All regions of interest after locally adaptive thresholding by Bradley + Wellner method, with different function above them. mean; Gaussian; median

6.5 Connected-component labeling

After the segmentation process, we have two separate groups of pixels. One of them represents background and the other one foreground i.e. data. Because we want to separate every single digit, we have to mark every closed white area in the image. For this purpose, we can use connected-component labeling algorithm which is sometimes called blob extraction or region labeling. This approach is the algorithmic application of graph theory where subsets of connected components are uniquely labeled based on a given heuristic. This method is based on two pass system [59].

During the first pass, algorithm travel through an image pixel by pixel from the top left corner. Also, we made a copy of the original black and white image. If the pixel is foreground, it checks its 8 or 4-neigborhood. In case no labeled pixel is in this neighborhood, actual pixel gets the new value for a label. If there is one already labeled pixel in this neighborhood, actual pixel gets the same value as the founded pixel. If there a two or more pixels with different labels, actual pixel gets the value of the smallest one and all other founded labels are stored in pairs <parent, child> in the memory. Where a parent is the smallest value of the label in 8-neighborhood and child is another founded label.

If this happens again for a next label, for example, label 2 is a child of label 1 and we now obtain that label 3 is a child of label 2. We check the parent list for information if label 2 is a child of some label. Because it is true, label 3 is stored as a child of label 1. Another way how to do this is simply store that label 3 is a child of the label 2. Moreover, do the second pass multiple times.

In the second pass, we connect adjoining areas with different labels into one. We go through the labeled image and check if actual pixel label value is a child of some other label. If not, we move to the next pixel. However, if it is true we relabel the value of the actual pixel to the value of the founded parent.

After these two passes, or after the first pass and multiple iterations of the second pass, we obtain the image with a unique label for every closed area in the image. An easy example is provided in the Figure 29.

$$\begin{bmatrix}1&1&1&0&1\\0&0&0&0&1\\1&0&1&0&1\\1&0&0&0&1\\1&1&1&1&1\end{bmatrix} \qquad \begin{bmatrix}1&1&1&0&2\\0&0&0&0&2\\2&0&3&0&2\\2&0&0&0&2\\2&2&2&2&2\end{bmatrix}$$

Figure 29 Example of labeling. Original binary image; image after labelling

6.6 Detect the closest right candidate

If we have labeled closed areas inside the image, we can detect the candidates for digits. If we look at the Figure 30, there are still more areas which could be misinterpreted as a digit. Also, the region of interest is during the calibration process selected by a user. That means ROI can be really big and there could be plenty of candidates for digits and also many groups of digits. Because of that, it is necessary to select the correct one group of digits.

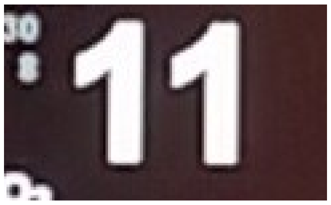


Figure 30 Selected region of interest

This selection is done by a user. The user is during the calibration process, immediately after selection of ROI with data, asked to click on, or as close as possible, to required data i.e. digit. When the pixel in the ROI on a clicked position is labeled i.e. it is not a background, this candidate is tested if it is suitable candidate for a digit or not. These tests are described in 6.6.1.

If a candidate does not pass these tests or on the clicked position is a background, application tries to find the next candidate as close as possible to a clicked point. This process can be described as a snail method. Because the order of tested pixels looks like a snail's shell. That means we go to the top from the clicked point, further one to the right, two down, two left, three top and so on. This process continues until the candidate which meets the tests conditions is found. If candidate which meets the test is not found, the user probably chose the ROI badly, or monitor is turned off

Once when a candidate is successfully founded, we store average color, width and height of the candidate. Because we know the position of all pixels of the selected label in the ROI and also the position of the ROI in the whole captured scene, we can easily obtain the average color from the original colored scene. We also store the position and size of the smallest possible rectangular selection around this candidate i.e. bounding box.

6.6.1 Test for a closest candidate

It is a waste of time and resources to try to classify by OCR all labeled areas inside of the ROI. That means we need to select only these which could be digits. Digits have some characteristics which make it possible to describe them. All selected candidates are tested if meets these characteristics.

These characteristics can be formed in conditions which build the whole decision tree. When we want to compare the new candidate to these characteristics we need to compute them. Because of that, a decision three has the characteristics which can be easiest calculated closer to the root of the three. With the increasing complexity of the characteristics, the distance from the root is bigger. All these tests are independent to the size of the candidate.

In following subchapters I describe these characteristics. All these characteristics were calculated on all digits i.e. 0-9 and diameters ()/ and I call it lately dataset. For testing, all candidates were separated to as small as possible rectangular area with pixels equal only to this label. This area is called bounding box. Values of these characteristics are stored in the Table 1. It's necessary to mention that these characteristics will be different for different fonts and also for a different position of the camera in the space. Conditions in tests were set with a certain reserve.

Table 1 Characteristics of digits on simulated patient monitor

Symbol	Aspect ratio	White ratio	Number of lakes	Avg. BW and WB steps	Maximum number of BW, WB steps in one row
1	2.190	1.875	0	2.054	4
2	1.586	1.748	0	1.760	4
3	1.586	1.793	0	2.173	4
4	1.382	2.054	1	2.276	6
5	1.533	1.707	0	2.173	6
6	1.566	1.600	1	2.574	6
7	1.491	2.336	0	1.648	6
8	1.533	1.549	2	2.782	6
9	1.533	1.604	1	2.510	6
0	1.533	1.627	1	2.695	6
(	3.428	1.647	0	1.479	6
)	3.50	1.649	0	1.1510	6
/	2.473	2.072	0	1.765	6

6.6.1.1 *Aspect ratio*

Digits and also all letters are higher than wider. Aspect ratio is the height of the candidate divided by its width. As I already mentioned digits are higher than wider, that means aspect ratio will be bigger than 1. Aspect ratio was calculated, and condition for a pass has been set as follows. It should be bigger than 1.1 and also smaller than 4. If it's not, it's not an appropriate candidate for a digit.

6.6.1.2 *White ration*

This characteristic is calculated as the ratio of white pixels in the bounding box. The result of that ratio is calculated by the formula (43). White ration should be bigger than 1.1 and smaller than 2.5.

$$WR = width * \frac{height}{NumberOfNonlabeledPixels} \tag{43}$$

6.6.1.3 *Number of lakes*

Some digits can contain one or more unlabeled areas surrounded by them. These areas are called lakes. Symbols in our dataset can contain 0-2 lakes. Lakes is detected by labeling of the inverted bounding box. After that, area with the candidate was expanded by few pixels on all sides. Without this step, we detect for example for number 8 up to 8 labeled areas, but after the expanding the number of lakes will be equal to three. One extra lake is the white background of the digit.

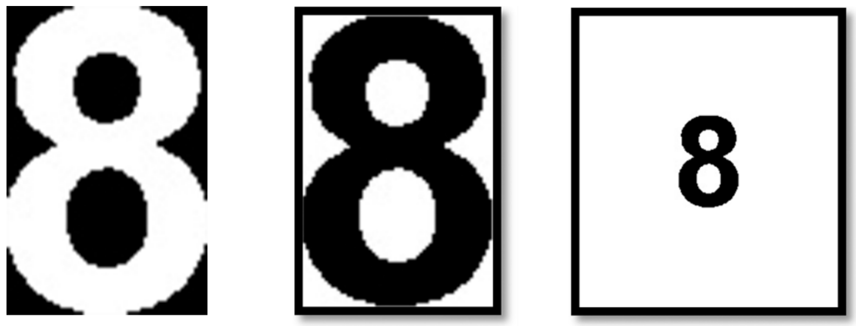


Figure 31 Counting number of lakes in digit process. White digit on black background; inverse digit with eight lakes; inverse digit with added borders with three lakes

6.6.1.4 *Black and white steps*

Digits and all other symbols in the dataset are solid. That means they don't have a high number of small lakes inside of them and also they are not noisy. Because of that, we can compute the number of black-white and white-black steps on an each row. Once when we have this, we calculate the average number of these steps in the whole bounding box and also store the maximum of these steps. A number of these steps should be bigger than one and smaller than 3.5. A maximal number of these steps for a dataset should be smaller than 8.

6.7 SELECTION OF CANDIDATES CONNECTED TO SELECTED ONE

Digits almost never appear on the patient monitor independently. That means we need to look for some neighboring candidates around the already selected candidate. Digits which belong together should be in the line and also close enough.

When we already have one candidate for a digit we can look on the left in the distance equal to founded symbol height. If we found some labeled area in this distance, we test it by tests in 6.6.1 and 6.7.1. If it passes through these tests, we redefine reference width and height for founding the next candidate on the left of the newfound candidate. That is done because, with more digits in one line and projective distortion, the very left digit should have a quite different size against the first candidate. That would not meet the test conditions. In the other case, i.e. candidate doesn't go through these tests, we still try to find a correct candidate in the previously defined distance. Because bad candidate could be only some kind of noise between the digits and there could still be a digit in the previously defined distance. This whole process is iterative until we don't find any candidate on the left which pass through these tests.

Once we found all the candidates on the left from the first founded candidate, we try to find all possible candidates for digits on the right. Before that, we need to redefine values of reference height and width by firstly selected candidate values. Furthermore, the process is the same as for the process to the left, just proceed to the right.

After we find all the candidates for digits on the left side and also on the right side, we redefine new bounding box i.e. smallest possible rectangular selection with the whole line of candidates.

6.7.1 Test for candidates in line

At first candidate for a digit was tested by a simple test for a size. After that, we test the similarity to the firstly selected candidate and in the end, the candidate was tested by all tests in 6.6.1.

6.7.1.1 Size

Because in binarize image could occur noise and other artifacts, it is useless to test all candidates by sophisticated tests. If a candidate is smaller than 20px, it is probably not a digit. If it is a digit, it means it have 5 x 4px and that is almost impossible to classify by OCR. An example of resized digit to that size is shown in the Figure 32.

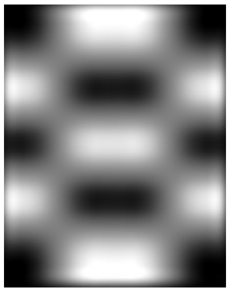


Figure 32 Digit with area equal to 20 px

6.7.1.2 Similar size to chosen candidate

We already have one candidate for a digit. That means we can compare some characteristics of that candidate to the tested one. The most basic characteristic is width and height of the candidate. Candidate should be higher than 0.75* reference height and smaller than 1.5 * reference height. Also, the width of the candidate could be smaller than 1.5* reference height.

6.7.1.3 Similar color to chosen candidate

Reference average color was stored after successful found of the first candidate. Once we have this reference color, we can split RGB image in separate channels and compare every one of them to a reference value. If tested value is inside +/-50 interval in all three channels, candidates pass this test.

6.8 DETECTION OF CHANGE

After all previous steps, we have completed calibration. The camera is focused on all ROIs respectively on all bounding boxes inside of these ROIs. In my application, it is possible to obtain new frame several times during the one second. A patient monitor ordinary doesn't change the value of digits on the screen more than once in a second. That means it will be a waste of time and resources to do all image manipulation operations on every frame which camera provides to us.

In a further application, I store black and white contain, after an adaptive threshold, of every bounding box. When the camera captures the new frame, this frame is cropped by rectangular selection with a size and location of stored bounding box. After that adaptive thresholding is applied. After that last stored contain of the bounding box is compared to the actual cropped black and white selection. If there is a change in an occurrence which is bigger the five percent, it means something is happening. All frames with this value below five percent were obviously the same.

This detection is done during the acquisition process. That means we already know some characteristics about regions of interest and digits in it. Because of that, we also test the first founded candidate to series of test in 6.6.1 with respect to the stored reference values.

6.8.1 Change of number

The first alternative what happen inside the bounding box is a just simple change of the number. Because we have the whole number which is composed of multiple digits inside the bounding box, change one of them cause difference bigger than five percent. In that case, first of all, we redefine selected region of interest. Top left corner is now lying on different coordinates and size is also different. The new size of the ROI is defined by following formulas (44-47).

$$TopLeft_x = TopLeftBB_x - 2 * BB_{height} \tag{44}$$

$$TopLeft_y = TopLeftBB_y - BB_{height} \tag{45}$$

$$ROI_{width} = BB_{width} + 4 * BB_{height} \tag{46}$$

$$ROI_{height} = 3 * BB_{height} \tag{47}$$

If one of these coordinates cross the border of the captured scene, the new value is set to this border. That ROI is defined only for a testing, and it is not stored inside of class as characteristic of ROI for now. Now we have new ROI and can continue with the definition of the location of user click. New click is automatically set to the center of newly defined ROI. After that we continue with similar steps as during calibration process i.e. connected-component labeling, detect the closest right candidate and selection of candidates connected to selected one. If after all of these steps, we found candidates for digits inside of the new ROI we store the new position of the ROI, new bounding box, newly calculated reference height, weight and color of the digits inside of bounding box. After that, contain of the bounding box is ready to send to classification step and application again waiting for the next difference bigger than five percent.

6.8.2 Movement of the number

In the second case, someone is gently moving with the patient monitor. When a digit is moving inside of the bounding box, the newly captured image inside of that selection is whole moved in some direction. That will result again in difference bigger than five percent. In that case, we define new ROI by calculation mentioned in 6.8.1. Because during the gently moving with the patient monitor we can still find candidates for digits in this ROI. After the application found them, it defines all parameters corresponding to the region of interest and digits inside, which are mentioned above in 6.8.1, and store them.

When the patient monitor is moving, almost in every case, we need to do the whole process of redefinition of the region of interest in several frames one by one until movement with this monitor ends and scene is again stable. The application could have problems when movement is too quick. That is because candidates for digits will be blurred and don't pass tests for these candidates in 6.6.1 and 6.7.1.

6.8.3 Area of interest is covered by something

Another case what can cause difference bigger than five percent could be something or someone who covers the monitor, or when the monitor is turned off. In that case, we again redefine new ROI by formulas in 6.8 and try to find candidates for digits. Because there are no candidates which pass through tests in that case, we know something is happening with the patient monitor i.e. monitor is off or something covering the region of interest. The application does not store the new position of the ROI, new bounding box, newly calculated reference height, weight and color of the digits inside of bounding box. However, stay with the old ones and wait on the same position until candidates for digits appear again. That change again causes the difference bigger than five percent.

6.9 Classification of candidates

Once when we have selected candidates for digits in bounding box we can try to classify them. Theory of possible methods is mentioned in 5.1.2.4 so here I describe how it work inside of the application. In the application I use two-step classification system. Where first step is Tesseract library. If this library fails in classification, the application tries to classify candidates by difference with stored masks.

Classification in the application is in use only when timer tick. That means all inner processes which are caused by change detection 6.8 are independent to the classification. This could be a problem when timer tick and we want to classify the candidates in the stored bounding box, but one bounding box contains a digit during the redrawing. In that case, the classifier gives us probably a bad result. This problem occurs very rarely, but there is still this possibility. This problem is shown in the Figure 33.

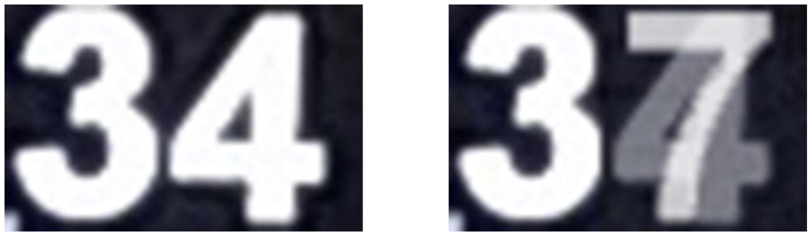


Figure 33 Candidates for digits obtained correctly; candidates for digits obtained incorrectly between two values

6.9.1 Tesseract

At first, the application tries to classify candidates inside of the bounding box by Tesseract library [11]. This library is relatively sophisticated and tries to do some preprocessing processes before the actual classification. At first, the library is trying to find the lines of the text inside of the image. Because our bounding box is as small as possible rectangular selection with all candidates for digits, we need to expand this bounding box. This process was simply done by expanding of the bounding box in all directions by a black border with size 100px, because we already have white text on a black background. If we keep candidates too close to the border of the bounding box, Tesseract library will a have really big problem with classifying of these candidates.

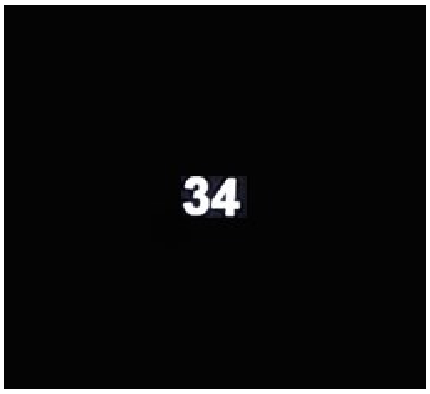


Figure 34 Expanded bounding box for one region of interest

Another step is word recognition. Because we have only one world in the bounding box, this step is unimportant for us. After that library continues by trying to chop connected candidates, extracting features, and actual classification. For more about all processes, please refer to [28].

The big advantage of this library is already completed dataset of training data for the classifier. Because on the patient monitor, as I mentioned in 2.1, occurs almost only synthetic fonts I use classical dataset for English which is included in Tesseract. It is also possible to learn Tesseract directly on our dataset, that gives us better results for the concrete monitor, but this application should work on different types of monitors.

Another advantage of Tesseract library is that we can define which characters could occur in the image. Because patient monitor has a limited set of symbols, which are mentioned in 6.6.1. Tesseract tries to classify candidates only to these symbols. If a match is not above the threshold defined by Tesseract, Tesseract returns an empty string. In that case, the second classification which is a difference with stored masks is called.

6.9.2 Difference with stored masks

If classification by Tesseract fails, the application tries to classify candidates by using similarities with stored masks. Before this process, we need to make masks with the stored digits. Each of these masks should contain only one white letter on the black background inside of the bounding box. Once when we have these masks, we need to define which mask contain which symbol. This option is for now done internally in the code by a List. Now we have defined and stored all mask for upcoming comparison.

The bounding box which contains candidates for classification should by separated to individual candidates at first. This can be easily done by connected-component labeling inside of the bounding box. After that, we separate every label with the corresponding new bounding box.

Now we need to resize contain of every newly created bounding box to image with size 35 x 35px with maintaining the same aspect ratio. Because symbols are higher than wider, we resize every candidate to height 35px and width will be calculated automatically with respect to aspect ratio. After that, we add a black border with the same width to the both sides of resized candidate to complete the square to 35 x 35px. This process is shown in the Figure 35.

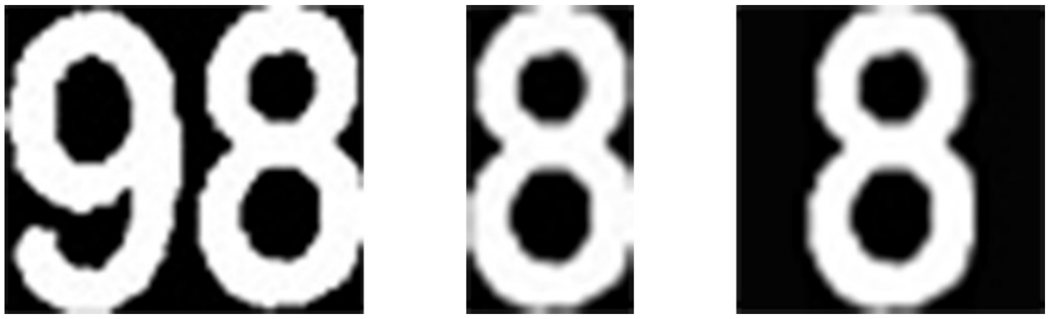


Figure 35 region of interest in bounding box; one separated candidate for digit; one candidate resize to 35x35 px with the same aspect ratio as original

Comparison of the newly created candidate in 35 x 35px box and mask could begin. If the mask does not have the same size as compared candidate, we apply on this mask same resize process which is mentioned above. Now we use a simple absolute difference between the mask and candidate. Because we know that borders of the digit captured by camera ordinary are not smooth we apply morphological operator called open on it [60]. Morphological open is using in our case circle kernel with a diameter equal to 2px. Comparison of mask and candidate is shown in the Figure 36.

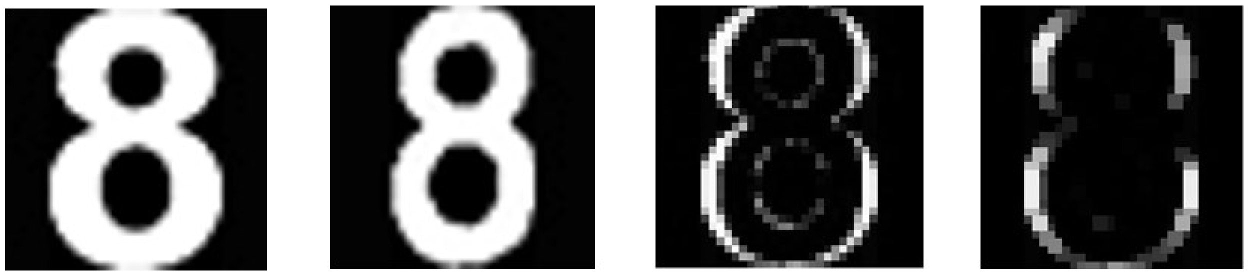


Figure 36 Stored mask with digit 8; candidate for digit; difference between the stored mask and a candidate; difference after using morphological open

If a candidate has a difference with a mask after these processes smaller than 2 percent, we add it to the list of possible digits. After testing of all digits, we choose the one with the smallest value of the difference from this list. This process is done for all founded candidates in the original bounding box with several candidates.

6.10 Saving to file

Once when we have classified data, we can store them. At the moment all data are storing in the CSV file [61]. This shortcut means comma separated values. On the first line of the file, you can find names of all ROIs separated by commas. On all other lines are results of classification for these ROIs. The column with the results corresponds to the column with the mane of the ROI. Data are stored in the CSV file whenever a timer tick. That means every time after the selected period of time.

6.11 Backup process

In chapter 6.8 I mentioned three different kinds of changes, but there is the fourth possible type of change. Change when the monitor is covered by something and during that, someone move with the patient monitor or a camera. After this situation happened, the application does not found any candidates in the new ROI which corresponds to previous ones. That means the process cannot go successfully through tests in 6.6.1 and 6.7.1. It is obvious because when no candidate is detected inside of the ROI, ROI still be waiting at the same coordinates in the whole scene and waiting until candidates again appear as it mentioned in 6.8. However, required data which we previously acquired is moved away while the monitor was covered.

This type of change is the most complicated for digits based approach. My solution which is mentioned below does not work well because there are few problems. Because of that this solution is not recommended and is not included in the tests in 8.3. This kind of change is much better solved in the monitor based approach 5.1. In that approach, if we locate the monitor in the captured scene we can easily locate the concrete information after the projective correction i.e. it does not matter if we move with the covered monitor or visible monitor.

6.11.1 Problems

There are few reasons which make it complicated. We do not want to obtain any data from the monitor, we want the concrete one. That means we cannot just find the closest fine candidate as close as possible to the previous location of ROI. As you can see in the Figure 37 on the patient monitor could be more data with the similar candidates. Because of similarity, these candidates could pass tests in 6.6.1 and 6.7.1. That means if we change the location of data in the scene, when patient monitor is covered, wrong data can be selected as required ones, when patient monitor is finally revealed.

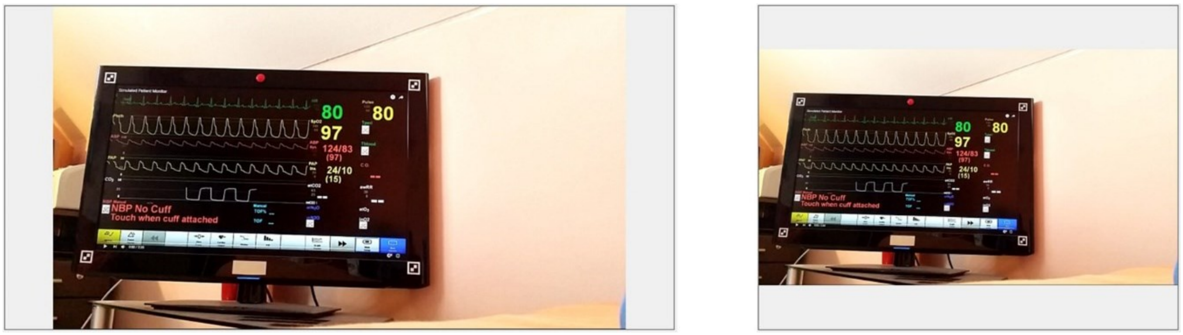


Figure 37 Captured simulated patient monitor by camera from same spot in both cases. Regions of interest are at different positions. Red target is at the same position in both images.

The second problem is the color of the candidates. If the monitor is moved away, there could occur some reflection, change of light conditions in the room or other changes which change the value of separated color channels quite a lot.

Third and also the biggest problem is a number of clusters with the candidates. The patient monitor is a dynamic hardware. So it is possible that when the patient monitor was covered, medical personal activates or deactivate some other parameter on the monitor. That means there is one more, or one less, cluster of candidates in the captured scene. Another option is when the patient monitor was covered, and someone put the next similar monitor in the scene. That is also a new source of clusters with candidates. There are more situations when new candidates can occur or disappear.

6.11.2 Definition of possible candidates

This process starts immediately after the calibration and also can be done at some interval during the acquisition process. Once when we have all characteristics of regions of interest and also about digits inside it, we can start this process. It necessary to find all possible clusters of candidates in the whole scene and their position in the scene before the actual acquisition process starts. A cluster of candidates is set, if minimally two candidates are standing in the same lane. All steps mentioned below are done for all defined regions of interest by a user.

First of all, it is necessary to find all possible candidates in the whole captured scene which are similar to digits inside of the ROI. This step is very similar to the whole process of digits based approach 5.2. The first step is grayscale of the whole captured scene same as in 6.3.1.3. After that, we binarize the scene by locally adaptive threshold 6.4.3. Now we have two separated layers i.e. foreground and background. All areas in the binarized scene are labeled by a connected-component labeling 6.5. Afterward tests in 6.6.1 and 6.7.1 are applied to labeled areas. At the end of this pipeline, we have all candidates for digits which are similar to digits inside of the ROI.

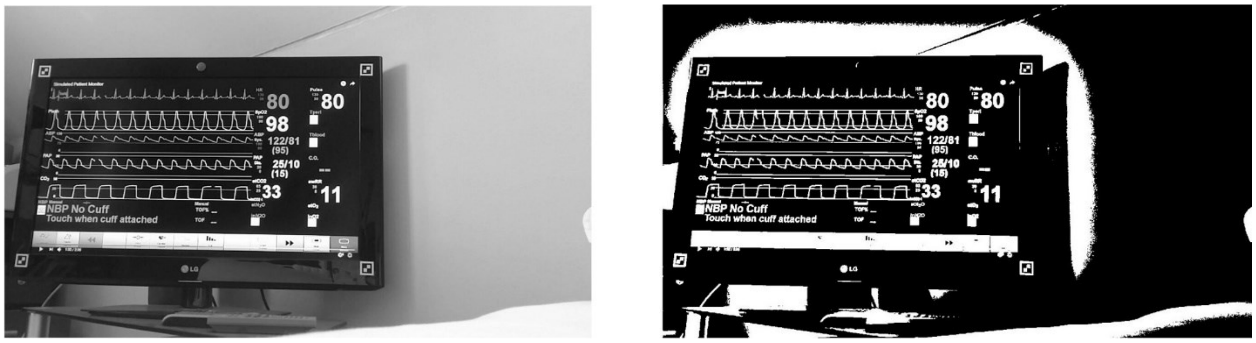


Figure 38 Grayscale captured scene; binarized scene by locally adaptive thresholding

Almost on every patient monitor, digits are shown in at least pairs. We connect together candidates for digits in clusters i.e. lines. Every single candidate has its own centroid. The centroid is the arithmetic mean position of all the points in the shape. Candidates in one line should be relatively close to each other horizontally, if patient monitor is horizontally located. So if the distance between two candidates in the horizontal axis is lower than two times size of the candidate they should be in one line. This distance was calculated by the formula (48).

$$Distance = |x_1 - x_2| \tag{48}$$

Another condition that must be met is position in the vertical axis. If centroid of a newly founded candidate for a neighbor on the line is located between the most top and most bottom coordinate of the second candidate, they are on the same line. After testing all candidates to all candidates, we have candidates separated in clusters i.e. lines.

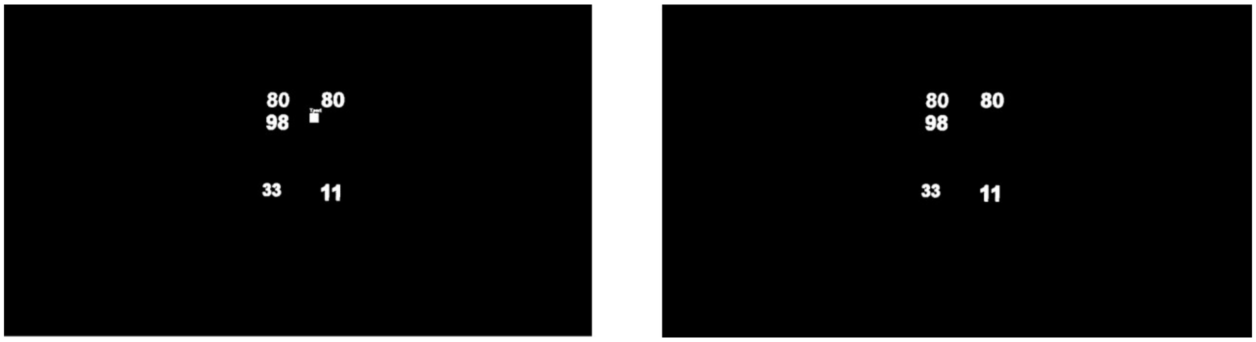


Figure 39 Selected candidates with characteristics similar to candidates in actual region of interest; candidates filtered to only candidates in clusters i.e. lines

Every one of these clusters can contain two or more candidates. Because we want to locate the whole cluster in the captured scene, we need some characteristic of this cluster. The fine parameter is the centroid of the whole cluster. The centroid is calculated by formulas below (49), (50).

$$GroupCentroid_x = \frac{\sum^{candidatesInGroup} centroid_x}{count(candidatesInGroup)} \tag{49}$$

$$GroupCentroid_y = \frac{\sum^{candidatesInGroup} centroid_y}{count(candidatesInGroup)} \tag{50}$$

Now we can determine the position of clusters in the vertical and horizontal axis in pairs <horizontalPosition, verticalPosition>. We obtain both of this information, but the position in the horizontal axis is more important for us. Because the position in vertical axis should change during the rotation around the vertical axis with the patient monitor in the scene. The rotation around the horizontal axis appears highly unlikely. This fact is nice to see in the Figure 37. Where on the left figure both number 80 are on the same line, but in the second one yellow 80 is definitely above the green one.

Once when we have information about the positions of the clusters, we compute the distance of all clusters to the centroid of actual ROI by formula (51). Once we found the closest one we store in this ROI position in vertical and horizontal axis for the upcoming backup process.

$$Distance_i = |x_i - x_{ROI}| + |y_i - y_{ROI}| \tag{51}$$

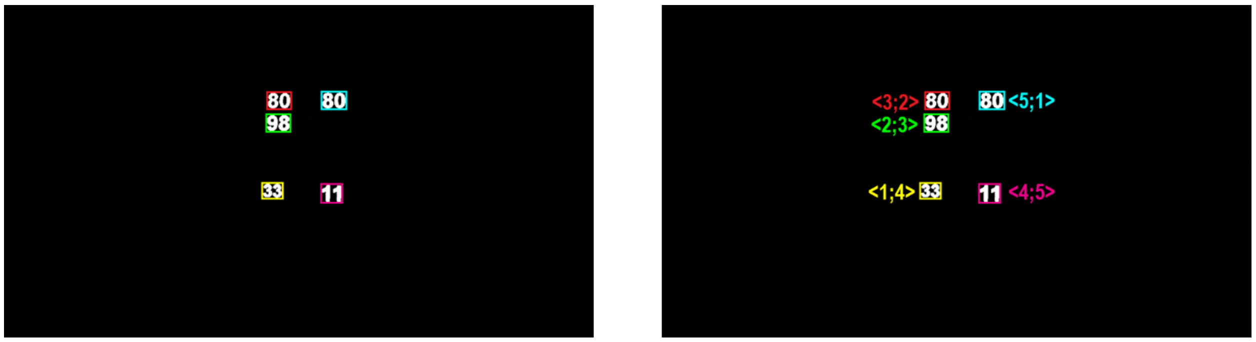


Figure 40 Selected groups of candidates; selected groups of candidates with its position in scene <horizontal, vertical>

6.11.3 Backup process

The backup process can be activated every time after a defined period of time, for control if every ROIs are located right and looking on a right data in the scene. On the other side the backup process could be activated only if some ROI does not contain candidates for digits for a long period of time. In implementation was chosen the first option.

When backup process is activated the same process as during definition of possible candidates 6.11.2 are done on the whole captured scene i.e. grayscale, binarization, connected-component labeling, filtration of candidates by tests, construction of clusters, the definition of their centroids, the founding of position in both axis. This is the last same step.

After that, we look in the ROI for location in the horizontal and vertical axis. That position was stored there during the definition of possible candidates process. At first, we look which cluster is at the same position in the horizontal axis. If this cluster has neighbors which have centroid located inside of its width in the horizontal axis. All these clusters are candidates for the right one. If not, selected cluster is the right one.

Now we can look at the positions of centroids of selected candidates to right clusters. We look on vertical position extracted from ROI, where this information was stored before, and select that cluster. If other selected candidates to right clusters centroids are located inside of the height of this cluster in the vertical axis. We still have more than one candidate to right cluster. In the other case, we select the right one.

If we still have more than one candidate for a right cluster, we select that one which has the best position in the horizontal axis. Because as I already mentioned above in 6.11.1, position in vertical axis could be incorrect. The process of founding green 80 is nicely shown in the Figure 41 and in the Figure 42.

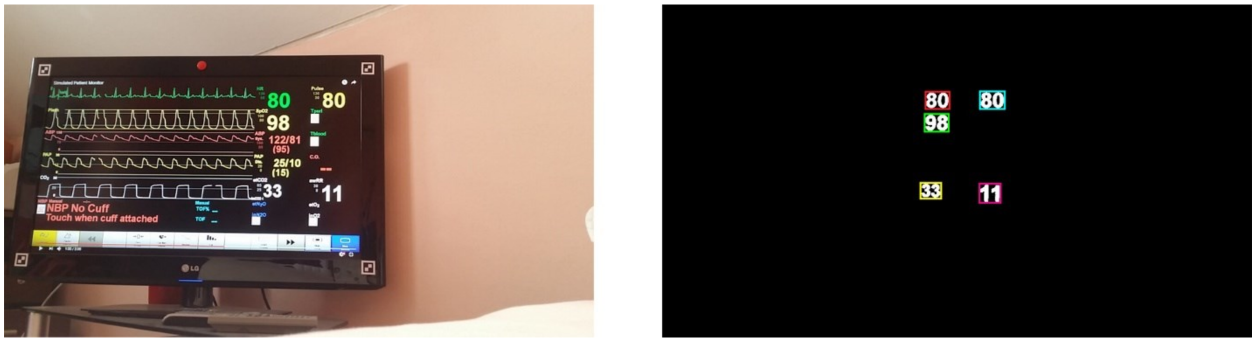


Figure 41 Original captured scene; selected candidates for clusters with similarities to digits in lost region of interest

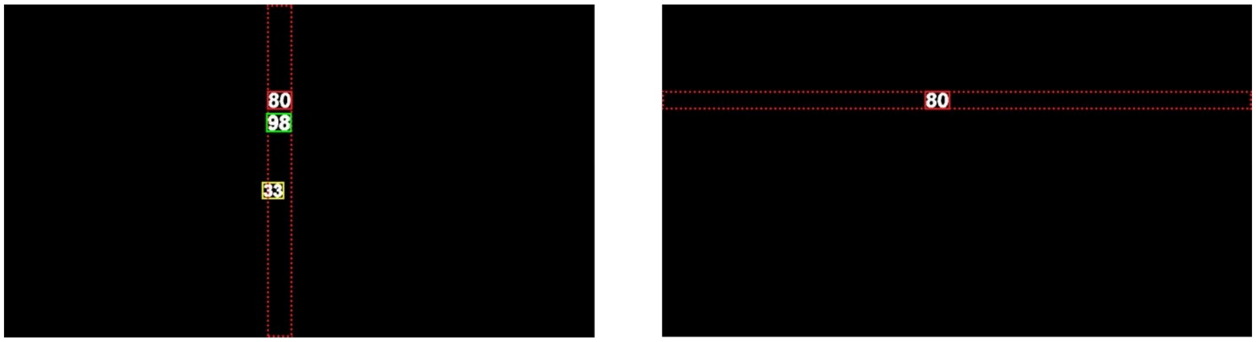


Figure 42 All possible candidates in a horizontal axis; all possible candidates in a vertical axis after a horizontal axis filtering

If we select the right cluster, we can redefine location and size of the ROI by parameters of the cluster. The new size of ROI is calculated from the size of the cluster because the size of the cluster represents in essence bounding box for digits. The size of the new ROI is calculated same as in 6.8.

6.12 Summary of used methods in upcoming application

In chapter 6 was presented several methods and algorithms which can be used during the digits based approach. Few of them are ideal for data acquisition from the patient monitor and few not. Following chapter sums finally selected methods and processes.

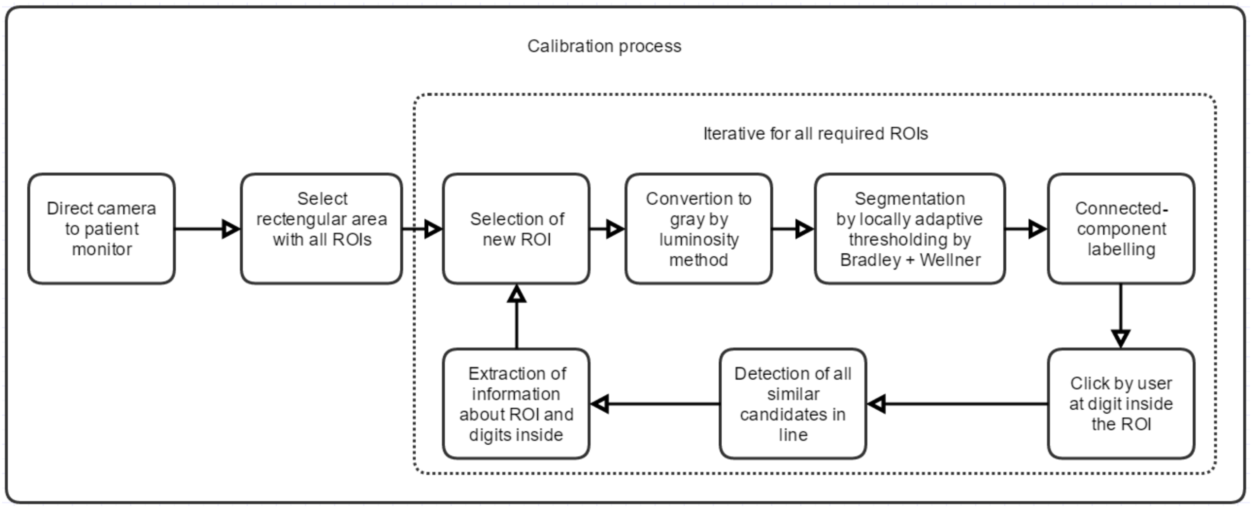
6.12.1 Calibration

At first, a rectangle area with all required regions of interest on the patient monitor screen is selected. For this step, there are no alternatives. This process is described in chapter 6.1.

After this step, it is necessary to select each region of interest. When one region of interest is selected by rectangular selection, it is converted to grayscale by luminosity method mentioned in 6.3.1.3. Once we have a grayscale region of interest, we convert it to black and white image by locally adaptive thresholding by Bradley + Wellner. As a function above the rectangular selection was selected Gaussian function with the size of neighborhood equal to 101. This method is mentioned in 6.4.3. After locally adaptive thresholding of selected region of interest, we segment it to foreground and background.

The foreground of the region of interest contains more than just candidates for digits. This could be noise, artifacts and other candidates for digits. We mark each area in the foreground by a unique identifier. For that purpose is used the Connected-component labeling mentioned in 6.5.

After the labeling, the first correct candidate is selected by the user, and its neighbors on the line are selected exactly as it described in 6.6 and 6.7.1. After that, we have all what is necessary for upcoming data acquisition process. All processes mentioned in three paragraphs above included this one, are done for all selected regions of interest. The whole process is shown in the Schema 6.


Calibration process
Iterative for all required ROIs
Direct camera to patient monitor
Select rectengular area with all ROIs
Selection of new ROI
Convertion to gray by luminosity method
Segmentation by locally adaptive thresholding by Bradley + Wellner
Connected-component labelling
Click by user at digit inside the ROI
Detection of all similar candidates in line
Extraction of information about ROI and digits inside

Schema 6 Calibration process for digits based approach with selected methods

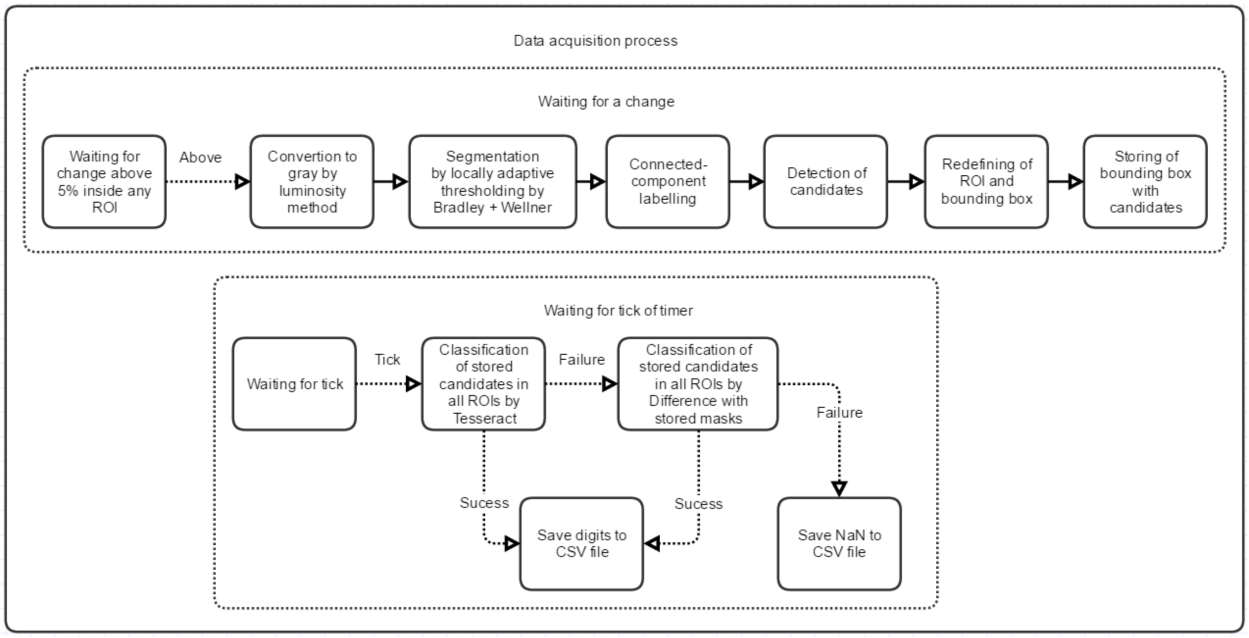
If the beta backup process is used, the application finds all possible candidates for all regions of interest. That means each region of interest contain information about its position in vertical and horizontal axis before the actual data acquisition. Beta backup process is described in detail in 6.11.

6.12.2 Data acquisition process

During the actual data acquisition is detected a change in each region of interest. If this change is above the selected threshold, an appropriate action is done. Used detection of change is exactly same as a detection mentioned in 6.8.

When timer tick at user defined time, the application tries to classify each region of interest by Tesseract. If Tesseract fails, the application tries to use difference with stored masks as a classifier. If this method also fails the result of that region of interest is equal to NaN. This is done for all regions of interest. For more detail, please refer to 6.9.

Classified values are saved to the CSV file. For more detail about CSV please refer to 6.10. The whole pipeline of data acquisition process is shown in the Schema 7.



Data acquisition process
Waiting for a change
Waiting for change above 5% inside any ROI
Above
Convertion to gray by luminosity method
Segmentation by locally adaptive thresholding by Bradley + Wellner
Connected-component labelling
Detection of candidates
Redefining of ROI and bounding box
Storing of bounding box with candidates
Waiting for tick of timer
Waiting for tick
Tick
Classification of stored candidates in all ROIs by Tesseract
Failure
Classification of stored candidates in all ROIs by Difference with stored masks
Failure
Sucess
Save digits to CSV file
Sucess
Save NaN to CSV file

Schema 7 Data acquisition process for digits based approach with selected methods

If the beta backup process is in use and tick violate its use, methods in 6.11.3 are used. That means we try to find the appropriate cluster of candidates at the newly captured scene by the position in the horizontal and vertical axis. This position was saved to all regions of interest during the calibration.

7 FUTURE EXTENSIONS

This diploma thesis is just a first step into the optical data acquisition via camera from the patient monitor. There was mentioned two approaches how to obtain textual information from the patient monitor. However, there are still many things which could be included in the application as an extra, like a multi-camera approach or curves acquisition. Other things can be done more precisely. These things are for example a classification and methods to separate digits in the captured scene. In following subchapters, I go through possible extensions with the greatest attention to the multi-camera approach and curves acquisition.

7.1 CURVES ACQUISITION

On the patient monitor are not only digits but also curves. These curves are useful for medical personal and can show many different trends. The most known is Electrocardiography i.e. EKG or ECG [62]. On EKG doctor can check, for example, if PQRST cycle looks like the one for the health person. At all, information stored in curves stands for depositing. Once when we deposit them, we can use some statistical analysis on them.

Curves on the patient monitor can be described as graphs. There is software which can extract data from images of graphs. For example im2graph [63], WebPlotDigitalizer [64] and others. This software is very similar to what we want to do as a future extension, but there are few differences. All this software use the defined axes, or you need to define them before the acquisition. Also, these programs work on static scenes, that means we obtain the data from them only once with the same ratio. There are also some more problems like that, these problems are described in 7.1.2.

7.1.1 Curves on patient monitor

On the market, and in hospitals, are many different types of patient monitors. They use different ways how to show curves on the patient monitor. Some of them contain some kind of axis, some of them contain multiple axes, when some of them do not contain any axes. An example of different styles of curves displaying is in the Figure 43. When I do a survey through a web, the most used style is the style without axes. However this survey it is not conclusive.

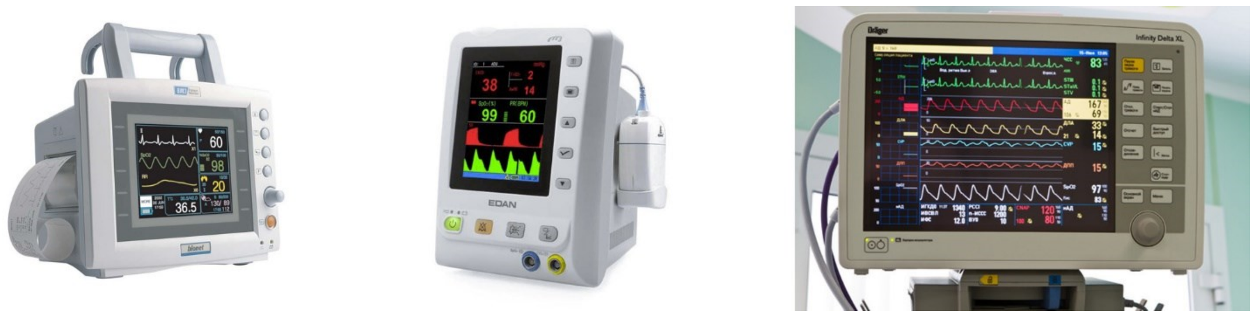


Figure 43 Different types of patient monitors with different way how to show curves on their screens [77], [78], [79]

First what should be mentioned is the style of curve redrawing on the patient monitor. We have two basic types of redrawing. The first one and more present in patient monitors is when the curve is redrawn gradually by a small black window. This small window travels iterative from left to right along the curve. Whenever that window appears above a curve, new information is shown on the left side of that window. This window can be called as some kind of redrawing window, and it is represented by a black window which covers the curve. This redrawing is nicely shown in the Figure 44.

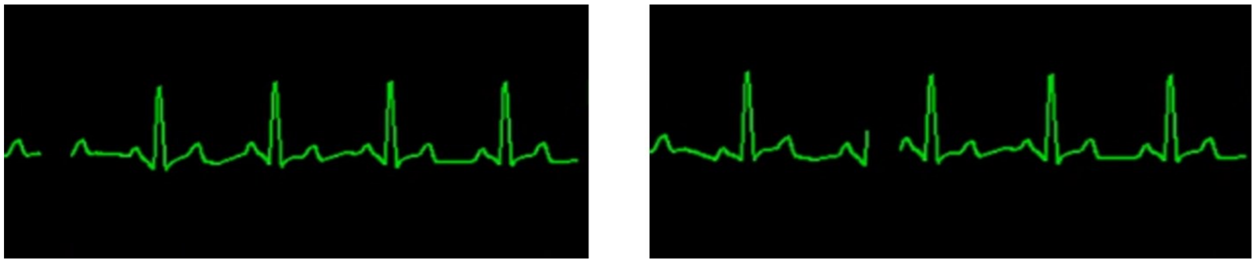


Figure 44 EKG curve with a different position of redrawing window

The second possible approach of redrawing is a movement of the whole curve on the patient monitor. That means when we need to show a new information in the form of the curve we move the whole curve to the right and show the new information on the left. Same amount of curve which is added to the left is deleted from the right side of the curve. At all, it looks like a whole curve is moving to the right.

7.1.2 Problems with curves

Curves on the patient monitor are not regular static graphs, and that cause many problems. In this chapter, I describe these problems and show possible solutions how to deal with them.

7.1.2.1 Axis problem

Patient monitors at almost all cases do not have any axes for curves. As you can see in the Figure 43 the patient monitor can contain some lines which look like axes, but ordinary these axes do not have any description about the values, units and scale. Some monitors do not have any lines on the screen. Because of that, we cannot simply subtract the value from the curve because we do not know the value in the horizontal and vertical axis.

The first solution seems to be the definition of axis for all curves by borders of the captured scene. This will be ok, but we must meet few conditions. Monitor and camera have to be set perfectly parallel to each other. That means top and bottom border of the captured scene by a camera must by parallel to the top and bottom edge of the patient monitor screen. That is related to that scene must be static i.e. no one can ever move with the monitor or camera during the acquisition process. Once when we meet these conditions, we can extract curve value in horizontal and vertical axis, where the pixel of the curve is located. In the end, this approach is bad for normal use because the scene is not ordinary static.

The second solution is based on the definition of axes for all curves by borders of the screen. This approach seems to be very ok, if we use a monitor based approach 5.1. The monitor is projective corrected on every frame, so borders and curves should be at the same position in these frames. If we move with the monitor, or camera, projective correction set screen back to the rectangle every time and we can simply extract the value of the curve in both axes. The projective correction also solves the scale problem when patient monitor is closer or further. Because every time we correct the patient monitor screen to the rectangular with the same size. This approach seems to be the most suitable solution and the user can define the scale of the both axes.

The last solution is the definition of every single axis for every region of interest manually by a user. This approach is usually used in mentioned programs in 7.1. All processes mentioned below are done on the projective corrected image. We simply set two points for the horizontal axis and two points for the vertical axis with the corresponding values. Because we do not know these values, as the best solution seems to be (0, 1) interval. Once we defined all axes for all curves, we can just simply extract information from them because these axes will also be projective corrected whenever patient monitor screen is corrected. This approach is fine, but because we do not know the values on axes, we can use the previous one. In the previous one there is less work for the PC and also for a user.

7.1.2.2 *Crossing curves*

On some monitors, curves can cross with axes or other lines on the patient monitor. This problem is really important because these lines look almost like a required curves. Shown curves ordinary do not look like a line, but for example, when the heart rate is equal to zero the curve will be the line, and that does not mean it is not a required curve. These lines should be misinterpreted as a required curves. These lines are nicely shown in detail in the Figure 45.

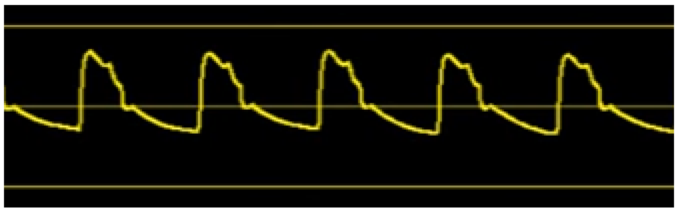


Figure 45 Crossing curves from background of the patient monitor with required curve which we want to obtain [79]

That what is typical for these lines is, they are all the time at the same position. That mean whenever we try to acquire data from the patient monitor these lines will be the same and that is the solution. We can store a mask before the required curves appear and use that mask as a background which we every time subtract from the actually captured frame. After that, the image obtains only the curve with some artifacts after the subtraction. Sometimes it is not possible to acquire mask without the curves. In that case, we use a bunch of frames from previous seconds and do a unification on them. All these frames should have same only these lines, which means we got the mask and we can continue with the process same as mention above.

7.1.3 Curves acquisition

The curves acquisition is quite similar to the monitor based approach 5.1. In this approach I contemplating a patient monitor with lines on the screen and the redraw window. There are more approaches how to obtain curves and this is only one of them. At all, we can use more methods which are mentioned in previous chapters, or some others, in every single step of curve acquisition.

First of all, we need to find corners of the monitor. If we have corners of the monitor, we can do a projective correction. After that correction, the user defines regions of interest with curves and application set left and bottom edge of the patient monitor screen as a vertical and a horizontal axis and the user set their scale.

Now we move to the actual acquisition and the need to extract a curve from a concrete ROI. At first, we grayscale contain of the ROI or use the value channel from HSV space. If we are done with this, we use some kind of binarization method. Because the curve could be relatively long, and some part of the patient monitor screen could be occupied by light reflection. Locally adaptive threshold 6.4.3 seems to be the best option. Sometimes the curve can be discontinuous. That problem will be solved by morphological close which connects close objects.

Now we have all candidates for curves in the ROI, so we need to filter some of them out. At first, we try to filter constant lines in the ROI. These lines can be removed by the method mentioned in 7.1.2.2. After that, curve can be cut in several pieces by removed lines. That means we want to connect them back. For that, we again use a morphological close. Another thing that is typical for curves is their bounding box. Area of their bounding box is much bigger than occupied area inside of it. Also, the size of the bounding box should cover at least approximately 1/20 of the ROI. Yes, this condition should cut out the small pieces at the end, or start, of the required curve in some situations. However, we obtain curves at least every second, and redraw time of the whole curve is longer. Another way how to filter curves is by a stroke width variation. Curves have this characteristic almost the same all along the curve. This method is used ordinarily for extracting text in natural scenes [65].

After these basic filters, we should have only two curves separated by a redraw window mentioned in 7.1.1. This window is essential for the upcoming storing process. We want to store only new data from every frame from the curve. So, we need to connect the old one curve which is already stored, to the new one obtained from the actual frame. The redraw window helps us with this problem. We store the position of this window whenever we save data from the curve to the database. We know this window is moving from the left to the right. We also know that curve between the old position of the redrawing window and the new position of that window is a new part of the curve. So we can store only the new data.

Filtrated curves on the patient monitor is not smooth, and there could still be some other parts in the ROI which are connected to curves. The value in the horizontal and vertical axis is simply calculated by following formulas (52), (53).

$$X = horizontalPositon * \max(horizontalAxis)/Width_{wholeCorrectedScreen} \tag{52}$$

$$Y = verticalPositon * \max(verticalAxis)/Height_{wholeCorrectedScreen} \tag{53}$$

Horizontal position is the median of a point in a curve from the horizontal neighborhood with size equal to the appropriate value which depends on the resolution. Vertical position is defined in the same way but in the vertical neighborhood. We also can use mean value, but the median value is more robust to outliers. If we solve this for one point, we move one pixel further in the horizontal axis.

The storage of the curve is opened problem. Because when we want to storage curve point by point it takes a huge place on the hard drive. This is a general problem of time series. For example, one hour of EKG takes 1 Gigabyte [66]. We can save only some features about the curves like max value, min value, autocorrelation, zero crossings, ARIMA, etc. or space the curve with lines. Sometimes is necessary to storage the whole data point by point.

7.2 MULTI-CAMERA USAGE

With the multiple sources of data, we obtain more robust results. That also work for the multi-camera approach. If we use multiple cameras for data acquisition from the patient monitor, we can use multiple data sources in every single step of this acquisition process. For more cameras which are focused to the patient monitor is better to use a different position in the room where the patient monitor is located. If both cameras will be on the almost same spot, that means all processes on obtained frames will also be almost the same. Locate cameras in different positions in the space, also provide the bigger possibility that if the patient monitor is covered by something for the first camera, this obstacle probably does not block the view of the second one.

7.2.1 Possible situations

Every single camera of multi-camera approach work itself i.e. all of these cameras has its own thread. At the end of the acquisition pipeline, results from all cameras are compared and best option for the result is select. After the actual acquisition can occur more situations. Depending on the situation we select the best option. All these options are mentioned below. All situations will be described for the two camera system, but usage in the multi-camera system is almost the same.

7.2.1.1 Two same results

If both threads after the classification contain the same result, the solution is easy. The final result of the classification is the value which is classified by both threads.

7.2.1.2 Two different results

If one camera classifies one value and the other, classify different one we have to aggregate results. Simplest way is a comparison of classification error of both threads. Afterward, we select the result from the classifier with lower classification error. The more sophisticated solution is classifying of both bounding boxes with candidates by another type of a classifier. After that, we compare the sums of classification errors for all obtained results. Option with the lowest sum is afterward select as the final result.

7.2.1.3 One result and no candidates

This situation means that one camera source is covered by something, or the monitor is off, and the other not. The final result is the result of the thread which contains the result.

7.2.1.4 One result and candidates without label

In this situation, one camera could not classify and the other on yes. In that case, we use one extra classifier to classify both bounding boxes again. After that, we compare the sums of classification errors for all obtained results. Nonempty option with the lowest sum is afterward select as a final result.

7.2.1.5 Candidates without label and no candidates
In that case, we try to use a new classifier to the candidates without a label. If that classifier gives us a result we use that one as a final result, if not the result is NaN.

7.2.1.6 No candidates at all
That means both cameras are covered by something or the monitor is turned off. In that case, the final result is NaN.

7.2.2 Use during the backup process
The multi-camera system could also be useful during the backup process 6.11.3. If one camera for a long period of time does not extract any candidates for digits, camera or monitor has probably moved away, while something is covering it. That means we try to use the backup process.

At the end of this process, we have separated clusters of candidates for ROIs. One of these clusters is selected for a concrete ROI. Once when this cluster is selected, we can switch this ROI to the test mode for few upcoming second. If classified data from the test ROI for one camera correspond to classified data from the second camera, the backup process selects the right ROI for a cluster and we switch this ROI to the normal mode. If results from these two ROIs are different, we redefine test ROI to the new cluster from the backup process. This process continues until all clusters are correctly paired with their ROIs.

7.3 OTHER EXTENSIONS
In this chapter, I try to introduce other possible extensions. All of these extensions are almost in every case connected to the problem in the application.

7.3.1 Separation of candidates
Separation of candidates is, in my opinion, the biggest problem of this diploma thesis. Because when we are testing candidates by tests in 6.6.1 and 6.7.1. We assume that these candidates are separated characters with specific characteristics. In case that two or more characters are connected we do not accept these candidates.

The solution for this could be erosion with the appropriate kernel. This solution is useful only in cases if the connection between the characters is significantly smaller than stroke width of the letters. If stroke width has a similar size as the connection between characters. Erosion starts to erode also characters itself.

A Nice solution of candidates separation was published by Lukas Neumann [26]. In this method, we measure the distance between the top and low pixel in every column, in a projective corrected candidate. After that, we find local minimums and try to cut candidates gradually in these extremes and classify newly created candidates. An example is nicely shown in the Figure 46.

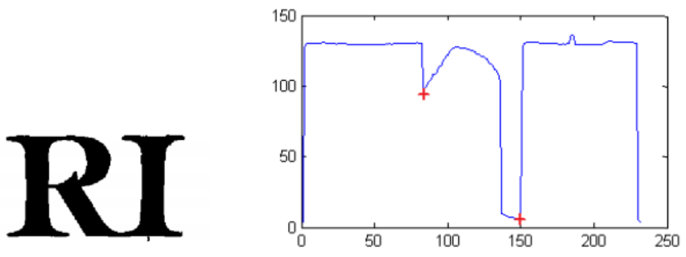


Figure 46 Connected candidates for symbols; graph of distance between top and low pixel in every column with local minimums [26]

7.3.2 Combining of both approaches

Both of mentioned approaches for data acquisition have its advantages and disadvantages. If we combine these two approaches in one and use advantages from both of them, we obtain better results.

Monitor based approach is better, if someone is moving with the patient monitor, while something is covering it. Also for curve acquisition mentioned in 7.1, this approach is much more suitable. On the other side, this approach is relatively time and source consuming and we can use it only when borders of the patient monitor screen are recognizable.

Digits based approach is faster and should work with similar accuracy in the case of digits acquisition. We can also use it during the night, when the light condition in the room is dark. However, curve acquisition is almost impossible in digits based approach and the backup process does not work well.

In the end, that means if we want to extract only digits from the patient monitor and no backup process is needed we use the digits based approach. If it is necessary to use the backup process we use the monitor based approach, if borders of the patient monitor screen are recognizable. In the other case, we use backup process from digits based approach. In the end, if we want to acquire curves as well we need to use monitor based approach.

Another option is to use both approaches separately on two separated threads and combine the data from both of them in essential parts like results of classifiers.

7.3.3 Communication along the network

Now all classified digits from the patient monitor are stored to the CSV file [61]. It will be nice to make a client and server for this application. The client will be installed on a distance PC which could set calibration and start acquisition process on the server. The server will be installed on the gizmo with camera, in the room where the patient monitor is located. Once when client set all required calibration on the host, actual acquisition process begin. Client-server could be implemented as a thick or fat client [67]. Because sending the image via the internet or internal network is a big amount of data, the thick client looks like a better option. All acquired data, or frames, will be sent to the client PC via TCP/IP [68].

That means it will not be necessary to visit the room with the patient monitor whenever we want to start data acquisition process.

7.3.4 Detection of separators

On the patient monitor can appear, in addition to digits, also separators. In this thesis, I work with three kinds of them. These are brackets and slash. On the patient monitor can occur some other types of separators. Most important are comma and dot. The problem with both of these separators is their big similarity to noise in the scene. That means it is necessary to implement filters which try to find these separators.

8 IMPLEMENTATION

Digits based approach was implemented as a standalone application in programming language C#. Some third-party libraries were used, they are mentioned below in 8.2. For graphical user interface (GUI) of the application was selected windows forms. A Newer version of making GUI in C# called Windows Presentation Foundation is not supported by an EmguCV library. GUI is resizable and should show information readable on almost all computer screens.

The main goals of the application were two. First of them is the quality of classified digits from patient monitor. The user does several settings during calibration process every time before the actual acquisition, for more robust results. The second one is performance, because in the near feature, it will be fine to use Mono C# compiler [69] and transform this application to mini PC like Raspberry Pi with a connected camera.

Raspberry Pi is a credit-size single board computer developed in the United Kingdom by Raspberry Pi Foundation to promote teaching on basic computer science in schools and in developing countries [70]. Five million Raspberry Pis have been sold before February 2015, that makes this mini PC best-selling British computer ever [74]. Average price for the newest model Raspberry Pi 3 is about $40 (3.1.2017). Camera, case and other accessories are not included with Raspberry Pi. Raspberry Pi has classical USB 2.0 connector, so it is possible to connect the camera via this connector. On this mini PC it is possible to run many types of operating systems, for example Raspbian, Fedora, Ubuntu Mate, Kali Linux, Windows 10 for IoT, etc. Raspberry Pi does not have a performance as currently sold average personal computers. That is the reason why all image transformation are done on the smallest possible selections. Raspberry Pi 3 Model B contain ARMv7 Processor rev 4 (v7l), which has performance equal to 1.09 GFLOPS per one core [75][76]. PC mentioned in 9, on which experiments was conducted has 4.35 GFLOPS per core [76]. For more details about Raspberry Pi, please refer to [70].

The application was implemented for 64bit Windows systems. This application also should work with 32bit Windows systems, but it is necessary to download the 32bit version of libraries mentioned below in 8.2. If you want to run this application on other systems, it is necessary to use different libraries for selection of camera source and make appropriate adjustments in code.

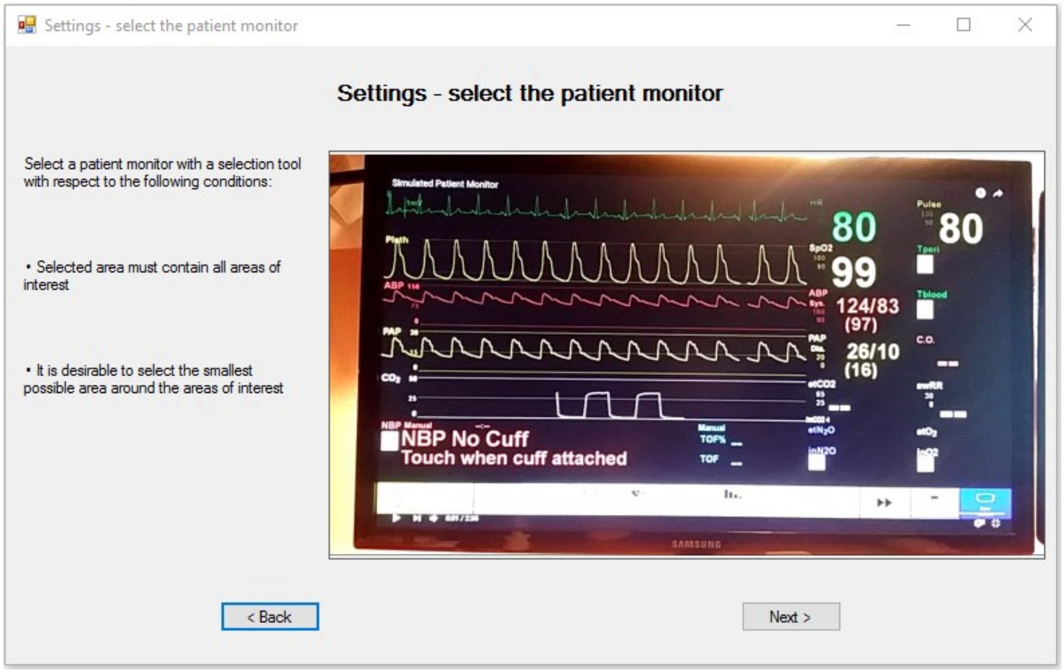
The camera is selected from all successfully installed cameras on the system.

8.1 BRIEF DESCRIPTION OF IMPLEMENTED CODE

This chapter describes most essential classes implemented inside the application.

8.1.1 Graphical interface

The graphical interface of implemented application is composed of seven windows forms. Every one of them has a different function in the application.


Settings - select the patient monitor

Settings - select the patient monitor

Select a patient monitor with a selection tool with respect to the following conditions:

• Selected area must contain all areas of interest

• It is desirable to select the smallest possible area around the areas of interest

Figure 47 Example of the GUI from implemented application

8.1.1.1 *frmMain*

This form works like the main menu of the whole application. There are only two possibilities what to do. First is the start of the calibration process before the data acquisition and the second is the exit of the application.

8.1.1.2 *welcomeSettings*

At this form user of the application is welcome and should continue.

8.1.1.3 *selectCameraSettings*

In this form is the user prompted to select one installed camera from shown combo box. Information about installed cameras on the system is provided by library DirectShowNet mentioned in 8.2.2. For this purpose is called class cameraSettings 8.1.2.1. When the camera is selected, its socket is saved to the new instance of class camera 8.1.2.2.

8.1.1.4 *setCameraInSpaceSettings*

In this form, the user can see video signal provided by the selected camera and recommendations for setting up a camera in space. Frames are captured whenever the application is in an idle state. The frame acquisition and its show in imageBox provide methods from EmguCv library 8.2.1.

8.1.1.5 *roiSettings*

In this form, the user selects all ROIs in captured scene by a rectangular selection. For this purpose are used action click, unclick and move while holding left mouse button. Left click by a mouse is defined as the start of the selection. Move while holding left mouse button is recognize as a size change of the selection. Unclick of the left mouse button is defined as the end of the selection. Coordinates are recalculated to the regular image by a manipulation mentioned in 6.2. This manipulation is implemented inside the class coordinatesManipulation mentioned in 8.1.2.5. The final selection with its coordinates is saved to the appropriate instance of class camera 8.1.2.2.

8.1.1.6 additionSettings

In this form user sets the name of the upcoming CSV file, the rate of classification and if beta backup process should be turned on or not. All these options are saved to the appropriate instance of class camera 8.1.2.2.

8.1.1.7 addAreasSettings

This form provides what is described in Schema 6 inside the part with name *Iterative for all required ROIs*. Whenever a new ROI is selected, this form call class labelSettings 8.1.2.3 which create a new instance of class label 8.1.2.4. All internal image manipulation are done externally in the actual label class 8.1.2.4. If next button is clicked, actual data acquisition process starts.

8.1.1.8 checkTrackingSettings

This form provides us information about actual contain of selected ROIs. All internal processes are called from this class. These are: wait for ticks, checking if change inside any ROIs is not above the selected threshold and saving to the CSV file. If button End session is clicked, the whole data acquisition process ends.

8.1.2 Internal classes

These classes ensure the functionality of the application.

8.1.2.1 cameraSettings

This class is a factory for a class camera 8.1.2.2. This class also provide a list of all cameras installed on the system by library DirectShowNet mention in 8.2.2. Information about the index of the active camera is stored in this class.

8.1.2.2 camera

Each instance of this class represents one camera. In this class are stored information about the calibration process.

8.1.2.3 labelSettings

This class is a factory for a class label 8.1.2.4. This class can add and remove selected ROIs.

8.1.2.4 label

Each instance of this class represents one selected ROI. All image manipulation processes which are associated with concrete ROI are done in this class. This class also contain all information about the ROI and digits inside of it. Function in this class corresponds to methods in 6.12, except the classification and saving to the file.

8.1.2.5 coordinatesManipulation

This class do exactly what is described in the chapter 6.2.

8.1.2.6 saveToFile
Methods in this class are called when the timer in application tick. That means application tries to classify contain of all bounding boxes by Tesseract. If Tesseract fails, the application tries to classify candidates by a difference with stored masks. Work of this class is described in the Schema 7 in a part called *Waiting for tick of timer.*

8.1.2.7 backUpProcess
This class do what is described in the chapter 6.11.

8.2 LIBRARIES

8.2.1 EmguCv
Because program works with images manipulation all the time, I use the cross platform .NET wrapper for the OpenCV library called EmguCV [58]. This library pack contains several image manipulation functions. Because it is not clear OpenCV, but only a wrapper, EmguCV sometimes did not work as should. There is also a problem with functions from this library.Some of them are missing in the used version for implementation, but older versions contain them. Used version was 3.1.0.2282

8.2.2 DirectShowNet
DirectShowNet library [71] provide the application all successfully installed cameras on the Windows system. Version of used library is 2.1.0.0.

8.3 TESTS
Testing of monitor based approach methods and also testing for the second approach was done in software called Matlab from company Mathworks [16]. All these tests were done on static images. The big advantage of Matlab is an image processing toolbox. This toolbox contains almost all image manipulations methods used in this thesis.

8.4 HOW TO START A SOFTWARE
The application has been built as a standalone. That means it is not necessary to compile it, before the actual use. The application is located on attached CD in folder StandAloneApplication_x64. The application was built to run on Windows systems with x64 architecture. For the successful start of the application, it is necessary to install .NET Framework 4.5.2. All libraries for the successful running of the application are included in the mentioned folder.

To start the application, please use file Diploma.exe. Once when the application is started, user obtain instructions what to do in every individual step of the calibration process. These steps correspond to individual windows forms mentioned in 8.1.1 and pipeline mentioned in 6.12. File with acquired data is localized in mentioned folder under the user defined name.

Folder also contains two additional folders. First of them is called Tessdata. This folder contains train data for Tesseract library. The second one is called TemplatesTestMonitor. This folder contains masks for classifying by Difference with stored masks.

9 EXPERIMENTS

The implemented application was tested on a simulated patient monitor. For this purpose was chosen Laerdal software [72]. This tool is a complex bundle of applications which can simulate various situation during the whole medical process. The patient monitor simulated by this software has the same occurrence as a classical patient monitor.

All experiments were captured by the 720p camera (1280x720 px) which transmitted a signal to the personal computer via Wi-Fi. For all tests were used the same simulation of the patient monitor with length equal to 156s. Tests were done under various conditions and during the different daytime. That is the reason why I choose a Laerdal solution, because we can run a simulation of patient monitor how we want.

During the experiments was tracked five sources of data, parameters, on the patient monitor. These data correspond to heart rate (HR), pulse, blood oxygen level (SpO2) and ambulatory blood pressure (ABP) in two options. First three parameters are easier to recognize, last two are more complicated. Last two parameters are smaller and do not contain only digits but also delimiters. Figures of all parameters will be shown in every experiment with information about their size.

9.1 TRACKED OUTCOMES

During experiments was evaluated several outcomes. In this chapter, I describe them and show how they are calculated. The reference data was obtained from the simulated patient monitor by a human. After that, collected values was checked five times for minimalizing error of the human factor.

9.1.1 Success rate

This outcome is a percentage of corrected classified values during the whole simulation.

$$SR = \left(\frac{count(correctResults)}{count(allResults)}\right) * 100 \tag{54}$$

9.1.2 Success rate after correction

In experiments, I obtained that delimiters are a big problem of classification. For example, a value which should be equal to 123/80 is classified as 123180. Because of that, I do an easy post-processing on data and correct these facts. It is possible because every ROI have some specific structure and for example value 123180 is absolute nonsense for ABP. Systolic pressure must be bigger than diastolic that means 123/180 is a bad result. Also, another position of slash in this result is nonsense. The final result is that slash was misclassified as a digit one. All following outcomes are calculated on post processed results.

The success rate after correction is a percentage of corrected classified values during the whole simulation after post processing. This outcome is the most important outcome at all, and gives us an idea of the result of experiments.

$$SRAC = \left(\frac{count(correctResultsAC)}{count(allResults)}\right) * 100 \tag{55}$$

9.1.3 No text

This outcome is a percentage of results with none founded candidates for digits.

$$NT = \left(\frac{count(NoText)}{count(allPPResults)}\right) * 100 \tag{56}$$

9.1.4 No result

This outcome is percentage of results which have after Tesseract and difference with stored masks no result.

$$NR = \left(\frac{count(NoResult)}{count(allResults)}\right) * 100 \tag{57}$$

9.1.5 Difference with stored masks

This outcome is a percentage of results which was classified by difference with stored masks after Tesseract classifier fails.

$$DSM = \left(\frac{count(testedByDWSM)}{count(allResults)}\right) * 100 \tag{58}$$

9.1.6 Difference with stored masks – success rate

This outcome is a percentage success rate of difference with stored mask on results which were classified by this classifier.

$$DSMSR = \left(\frac{count(testedByDWSM \;\&\; correctResultsAC)}{count(testedByDWSM)}\right) * 100 \tag{59}$$

9.1.7 Tesseract

This outcome is a percentage of results which was classified only by Tesseract.

$$T = \left(\frac{count(testedByTesseract)}{count(allResults)}\right) * 100 \tag{60}$$

9.1.8 Tesseract – success rate

This outcome is percentage success rate of Tesseract on results which was classified by this classifier.

$$TSR = \left(\frac{count(testedByTesseract \: \& \: correctResultsAC)}{count(testedByTesseract)}\right) * 100 \qquad (61)$$

9.2 EXPERIMENTS

9.2.1 One meter – light on

In this experiment, the camera was located one meter far from the simulated patient monitor. The plane of the patient monitor was almost parallel to the captured plane by the camera. Light in the room was turned on. During the whole experiment the camera and the patient monitor were on the same position.

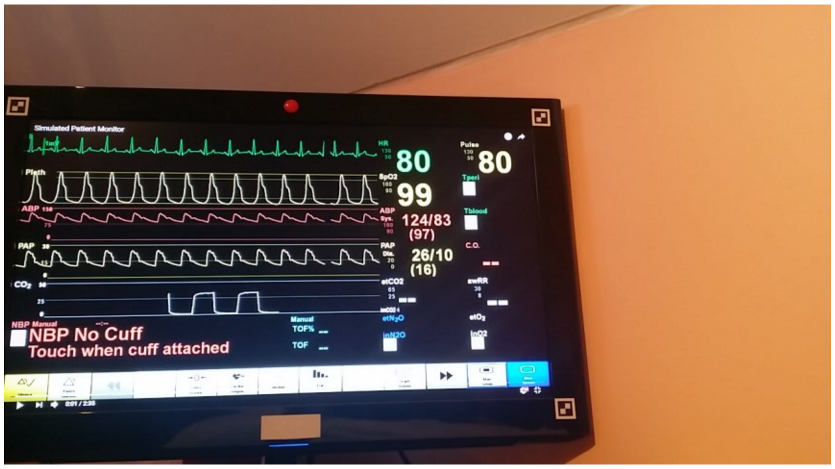


Figure 48 Captured scene from one meter with light on

9.2.1.1 *Scanned parameters*

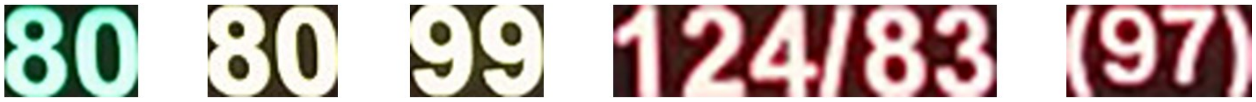


Figure 49 All scanned parameters in their bounding boxes from one meter with light on the frame before acquisition process. HR; Pulse; SpO2; APB_1; APB_2

Table 2 Size of digits for experiment from one meter with light on

Parameter	Avg. height of digit	Avg. width of digit
HR	34	24
Pulse	35	24
SpO2	36	24
APB_1	18	12
APB_2	16	11

9.2.1.2 *Results of experiment*

Table 3 Results of experiment from one meter with light on

Parameter	SR	SRAC	NT	NR	DSM	DSMSR	T	TSR
HR	100	100	0	0	0	NaN	100	100
Pulse	99.367	99.367	0	0	0	NaN	100	99.367
SpO2	100	100	0	0	0	NaN	100	100
APB_1	1.266	84.177	11.392	0	0	NaN	88.608	95
APB_2	87.342	87.342	0	0	0	NaN	100	87.342
Total	**77.595**	**94.177**	**2.278**	**0**	**0**	**NaN**	**97.722**	**96.342**

9.2.1.3 *Conclusion*

From results of this experiment in the Table 3, you can see that success rate is sufficient. For bigger numerical values without delimiters, SR is above the 99 percent. An interesting result is a difference between the success rate before and after correction at APB_1. This big difference means that almost all slashes were misinterpreted as digit one. No text at ABP_1 was caused by connection of a digit with the slash. As you can see in the Table 2, the distance between these characters is approximately not bigger than 3px.

9.2.2 One meter – light on – angle

In this experiment, the camera was located one meter far from the simulated patient monitor. The plane of the patient monitor screen was placed under the angle equal to 45 degrees to the plane of the scene captured by the camera. Light in the room was turned on. During the whole experiment, the camera and the patient monitor were on the same position.

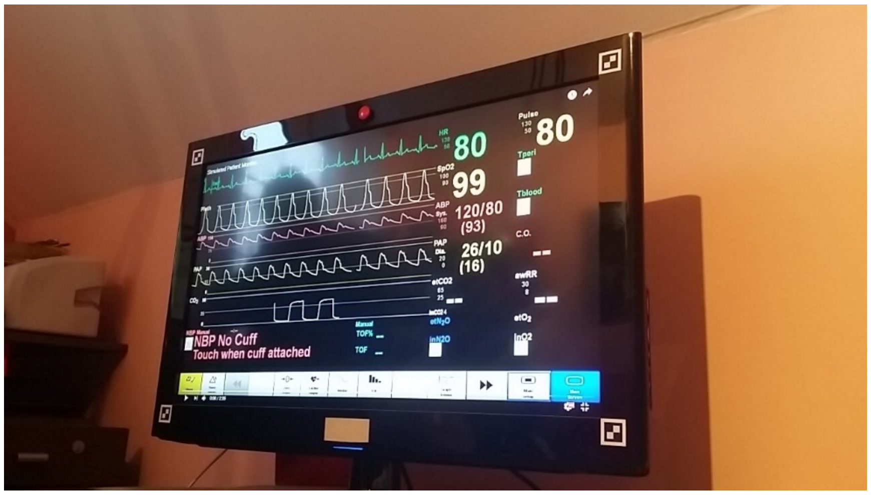


Figure 50 Captured scene from one meter with light on under angle 45 degrees

9.2.2.1 *Scanned parameters*

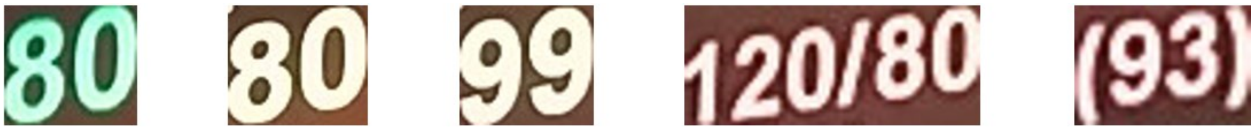


Figure 51 All scanned parameters in their bounding boxes from one meter with light on under angle 45 degrees captured before acquisition process. HR; Pulse; SpO2; APB_1; APB_2

Table 4 Size of digits for experiment from one meter with light on under angle 45 degrees

Parameter	**Avg. height of digit**	**Avg. width of digit**
HR	36	21
Pulse	39	25
SpO2	38	22
APB_1	19	11
APB_2	16	10

9.2.2.2 *Results of experiment*

Table 5 Result of experiment from one meter with light on under 45 degrees

Parameter	**SR**	**SRAC**	**NT**	**NR**	**DSM**	**DSMSR**	**T**	**TSR**
HR	97.468	97.468	1.266	0.633	0	NaN	98.101	99.355
Pulse	89.873	89.873	1.266	3.797	0	NaN	94.937	94.667
SpO2	98.101	98.101	0	1.266	0.633	0	98.101	100
APB_1	68.354	69.620	10.759	0	0	NaN	89.241	78.014
APB_2	93.671	93.671	0	0	0	NaN	100	93.671
Total	**89.494**	**89.747**	**2.658**	**1.139**	**0.127**	**0**	**96.076**	**93.141**

9.2.2.3 *Conclusion*

In this experiment, digits were affected by projective distortion. As you can see in the Table 5 overall success rate is lower than in case 9.2.1. What is really interesting is recognized slash in almost all cases. Another interesting fact is the lower success rate at parameter Pulse than HR and SpO2, because Pulse is the biggest and closest one to the camera.

9.2.3 One meter – light off

In this experiment, the camera was located one meter far from the simulated patient monitor. The plane of the patient monitor was almost parallel to the captured plane by the camera. Light in the room was turned off. During the whole experiment the camera and the patient monitor were on the same position.

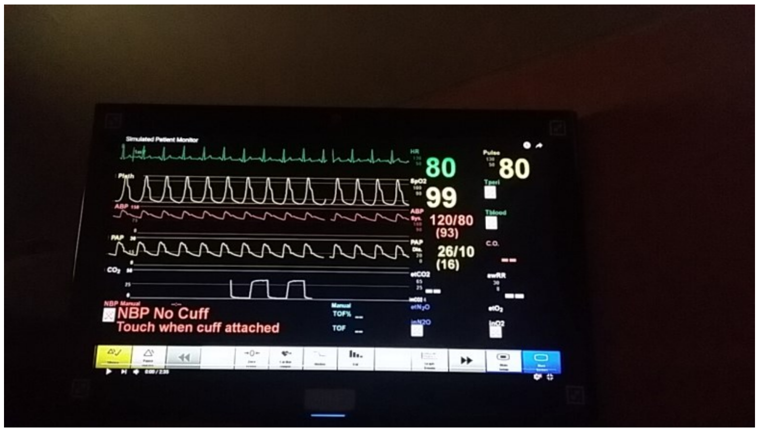


Figure 52 Captured scene from one meter with light off

9.2.3.1 *Scanned parameters*

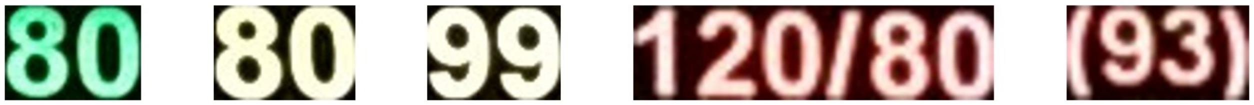


Figure 53 All scanned parameters in their bounding boxes from one meter with light off captured before acquisition process. HR; Pulse; SpO2; APB_1; APB_2

Table 6 Size of digits for experiment from one meter with light off

Parameter	Avg. height of digit	Avg. width of digit
HR	34	22
Pulse	34	23
SpO2	35	23
APB_1	17	11
APB_2	16	11

9.2.3.2 *Results of experiment*

Table 7 Results for experiment from one meter with light off

Parameter	SR	SRAC	NT	NR	DSM	DSMSR	T	TSR
HR	98.101	98.101	1.2666	0	0	NaN	98.734	99.3589
Pulse	99.367	99.368	0	0	0.633	0	99.367	100
SpO2	84.177	89.241	0.633	0	13.924	36.634	85.443	98.519
APB_1	48.734	54.430	22.785	0	0	NaN	77.215	70.492
APB_2	42.405	42.405	0	0	0	NaN	100	42.405
Total	**74.557**	**76.708**	**4.937**	**0**	**2.911**	**18.180**	**92.152**	**82.154**

9.2.3.3 *Conclusion*

When light conditions in the room are low, the overall success rate is smaller than in the case when the light is on. As you can see from the Table 7, the biggest influence on bad results has parameter APB_2 and its misclassification by Tesseract. Another issue is a connection of slash with digits which evaluates to NoText results at APB_1 parameter, because connected symbols do not fulfill test conditions in 6.6.1 and 6.7.1.

9.2.4 One meter – light off – angle

In this experiment, the camera was located one meter far from the simulated patient monitor. The plane of the patient monitor was placed under the angle equal to 45 degrees to the plane of the scene captured by the camera. Light in the room was turned off. During the whole experiment the camera and the patient monitor were on the same position.

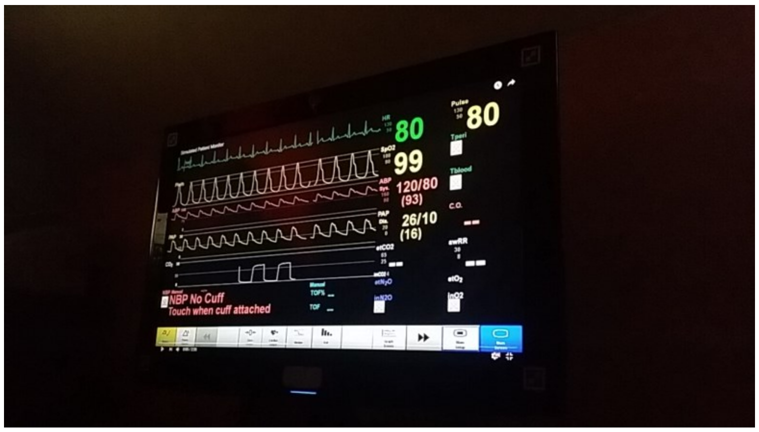


Figure 54 Captured scene from one meter with light off under angle 45 degrees

9.2.4.1 *Scanned parameters*

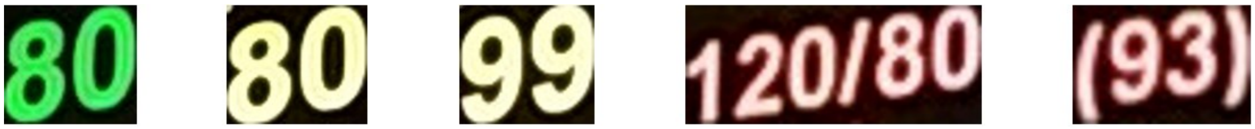


Figure 55 All scanned parameters in their bounding boxes from one meter with light off under angle 45 degree captured before acquisition process. HR; Pulse; SpO2; APB_1; APB_2

Table 8 Size of digits for experiment from one meter with light off under angle 45 degrees

Parameter	Avg. height of digit	Avg. width of digit
HR	37	22
Pulse	39	25
SpO2	37	22
APB_1	19	10
APB_2	16	9

9.2.4.2 *Results of experiment*

Table 9 Results of experiment from one meter with light off under angle 45 degrees

Parameter	SR	SRAC	NT	NR	DSM	DSMSR	T	TSR
HR	100	100	0	0	0	NaN	100	100
Pulse	98.101	98.101	1.900	0	0	NaN	98.101	100
SpO2	98.101	98.101	1.900	0	0	NaN	98.101	100
APB_1	63.924	68.987	23.418	0	0	NaN	76.582	90.083
APB_2	65.822	65.822	21.519	0	0	NaN	78.481	83.870
Total	**85.190**	**86.202**	**9.747**	**0**	**0**	NaN	**90.253**	**94.790**

9.2.4.3 *Conclusion*

In that case, Tesseract works really well. Overall success rate of Tesseract is above 94 percent and also parameter APB_2 which has a low Tesseract success rate in the case 9.2.3 has score above 83 percent in this test. That means, if correct candidates were founded in ROIs, they were almost every time correctly classified. The only problem was again fulfillment of test conditions in 6.6.1 and 6.7.1.

9.2.5 Two and half meter – light on

In this experiment, the camera was located two and a half meter far from the simulated patient monitor. The plane of the patient monitor was almost parallel to the captured plane by the camera. Light in the room was turned on. During the whole experiment the camera and the patient monitor were on the same position.

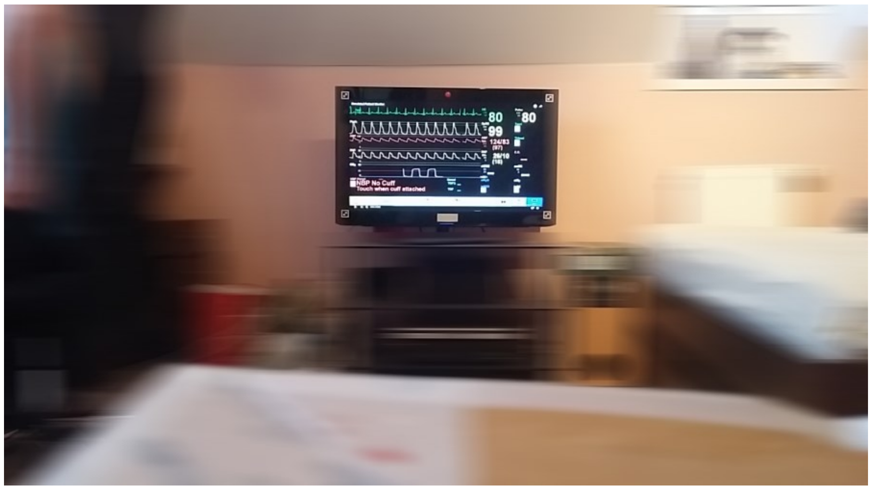


Figure 56 Captured scene from two and half meter with light on. Surroundings was blurred because of privacy.

9.2.5.1 *Scanned parameters*

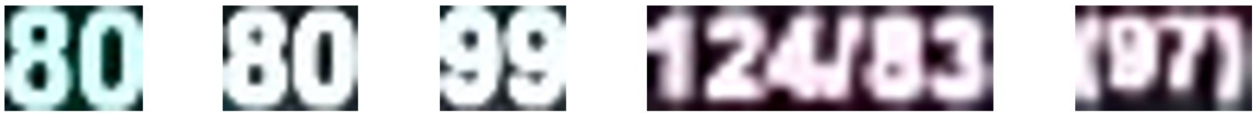


Figure 57 All scanned parameters in their bounding boxes from two and half meter captured before acquisition process. HR; Pulse; SpO2; APB_1; APB_2

Table 10 Size of digits for experiment from two and half meter with light on

Parameter	Avg. height of digit	Avg. width of digit
HR	15	9
Pulse	16	10
SpO2	15	6
APB_1	7	4
APB_2	6	4

9.2.5.2 *Results of experiment*

Table 11 Results of experiment from two and half meter with light on

Parameter	SR	SRAC	NT	NR	DSM	DSMSR	T	TSR
HR	59.494	59.494	8.861	27.848	0	NaN	63.291	94
Pulse	20.886	20.886	63.291	1.899	0	NaN	34.810	60
SpO2	20.886	20.886	67.089	0.633	0	NaN	32.279	64.706
APB_1	0	0	99.367	0	0	NaN	0.633	0
APB_2	0	0	99.367	0	0	NaN	0.633	NaN
Total	**20.253**	**20.253**	**67.595**	**6.076**	**0**	NaN	**26.329**	**54.677**

9.2.5.3 *Conclusion*

Overall success rate of this experiment is really low. At first, the success rate of APB_1 and APB_2 is equal to zero. These parameters are even hardly recognizable by a human as you can see in the Figure 57. Moreover, they are connected all the time, so they do not fulfill test conditions in 6.6.1 and 6.7.1. Also, three first parameters look really bad for classification, and they do not fulfill these test conditions. In the end, that means we need a better camera or data acquisition from this distance is useless.

9.2.6 One meter – light on - cover

In this experiment, the camera was located one meter far from the simulated patient monitor. The plane of the patient monitor was almost parallel to the captured plane by the camera. Light in the room was turned on. During the whole experiment camera was covered three times, overall for 25% of a simulation time.

9.2.6.1 *Scanned parameters*

Scanned parameters is very similar to parameters in 9.2.1.1.

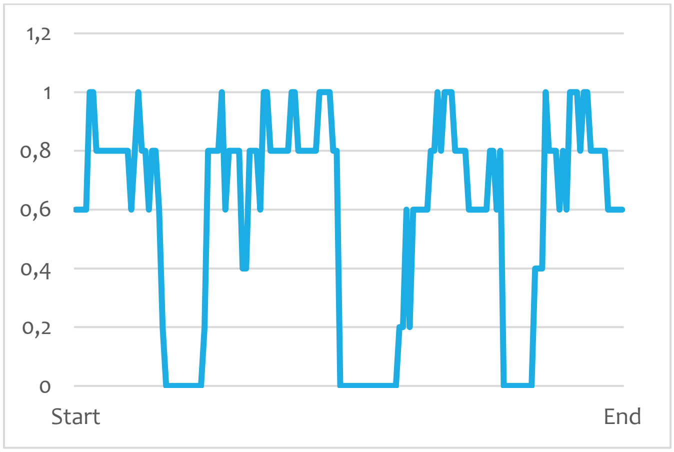
9.2.6.2 *Results of experiment*

For this experiment was calculated one extra outcome. This outcome is success rate after correction only when patient monitor is exposed.

$$SRACE = \left(\frac{count(correctResultsACE)}{count(ExposedResults)}\right) * 100 \qquad (62)$$

Table 12 Results of experiment from one meter with light on and covering the patient monitor screen three times during the experiment

Parameter	SR	SRAC	NT	NR	DSM	DSMSR	T	TSR	SRACE
HR	67.089	67.089	31.012	0	0	NaN	68.987	97.248	89.452
Pulse	70.253	70.253	24.051	0	0	NaN	75.949	92.500	92.500
SpO2	29.747	29.747	26.582	0	37.342	64.407	36.075	82.456	39.663
APB_1	0	30.380	43.038	0	0	NaN	56.962	53.333	40.507
APB_2	64.557	64.557	25.949	0	0	NaN	74.051	87.179	86.079
Total	**46.329**	**57.215**	**30.127**	**0**	**7.468**	**64.407**	**62.405**	**82.543**	**69.640**



Graph 1 Absolute success rate throughout the experiment from one meter with light on and covering the patient monitor screen three times during the experiment

9.2.6.3 *Conclusion*

As you can see in the Table 12 total outcome of NT is equal approximately to 30. That is fine because 25 is the minimal value, because the monitor was covered for 25% of the simulation. NT value for Pulse parameter is lower than 25 by one percent. That is because something on the thing which cover the monitor fulfills test conditions in 6.6.1 and 6.7.1 for a limited time.

Overall success is equal to 46 percent, if we recalculate this value to the monitor without covering, the success rate is equal approximately to 70 percent. This result is that low because the application needs some time to recalibration and also some parameters does not have good success rate from Tesseract and difference with stored masks. For example DSM result of parameter APB_1 is only 64%.

From a Graph 1 you can see that regular success rate of exposed patient monitor is between 60 and 100 percent. If the monitor, or the camera, is covered by something the success rate is equal to 0. That trend occurs three times during the simulation.

9.2.7 One meter – light on - move

In this experiment, the camera was located one meter far from the simulated patient monitor. Light in the room was turned on. During the whole experiment, the patient monitor screen plane was rotated around the vertical axis between 0 to 45 degrees. This angle is between the plane of the patient monitor screen and the plane captured by the camera.

9.2.7.1 *Scanned parameters*

Scanned parameters are a combination between 9.2.1.1 and 9.2.2.1 during the experiment. It depends on the actual angle of the patient monitor screen.

9.2.7.2 *Results of experiment*

Table 13 Results of experiment from one meter with light on and moving with patient monitor between angle 0-45 degrees during the whole experiment

Parameter	SR	SRAC	NT	NR	DSM	DSMSR	T	TSR
HR	53.165	53.165	27.215	0	0	NaN	72.785	73.043
Pulse	61.392	61.392	16.456	0	0	NaN	83.544	73.485
SpO2	74.684	74.684	0	0	5.696	33.334	39.873	79.194
APB_1	2.532	9.493	56.329	3.165	0.633	0	39.873	23.810
APB_2	4.430	4.430	55.063	2.532	1.266	0	41.139	11.112
Total	**39.240**	**41.013**	**31.013**	**1.139**	**1.519**	**11.111**	**66.329**	**52.129**

9.2.7.3 *Conclusion*

From the Table 13 is evident that success rate of all parameters is low. That is caused by actual movement with the patient monitor during the whole simulation. If we look at the example in the Figure 58 everything is now clearer. Symbols are connected and also does not have its specific characteristics. If the move is relatively slow, the application still can recognize candidates for digits and send them to the classifier.

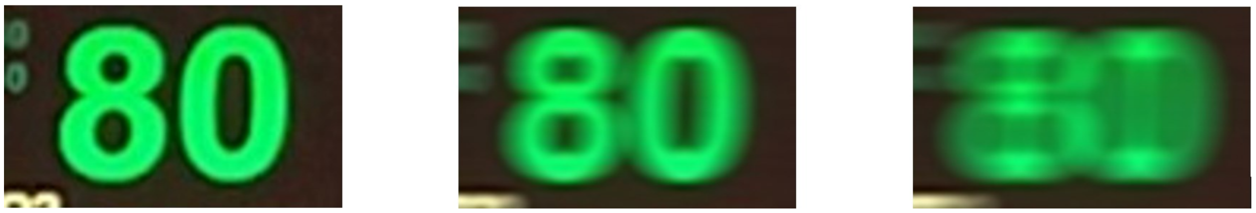


Figure 58 Captured digit in static scene; Captured digit during gently moving with patient monitor or camera; Captured digit during quick moving with patient monitor or camera

9.2.8 One meter – light on – move/pause

In this experiment, the camera was located one meter far from the simulated patient monitor. Light in the room was turned on. During the experiment, the patient monitor screen plane was rotated three times around the vertical axis between 0 to 45 degrees. This angle is between the plane of the patient monitor screen and the plane captured by the camera.

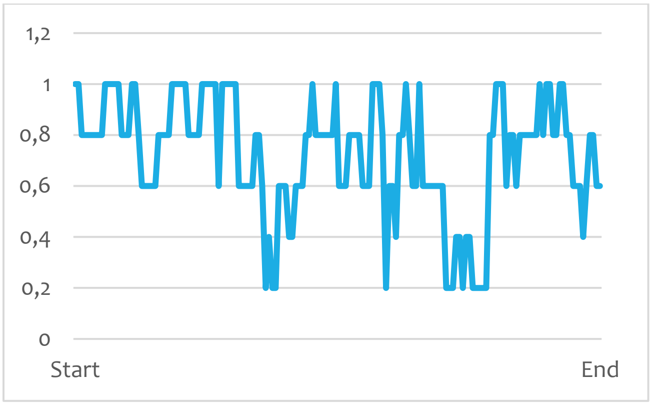
9.2.8.1 *Scanned parameters*

Scanned parameters are a combination between 9.2.1.1 and 9.2.2.1 during the experiment. It depends on the actual angle of the patient monitor screen.

9.2.8.2 *Results of experiment*

Table 14 Results of experiment from one meter with light on and moving with patient monitor between angles 0-45 degrees three times during the experiment

Parameter	SR	SRAC	NT	NR	DSM	DSMSR	T	TSR
HR	81.646	81.656	13.924	0	0	NaN	86.076	94.853
Pulse	79.114	79.114	17.722	0	0	NaN	82.278	96.154
SpO2	86.076	86.076	0	0	11.392	33.334	88.608	97.143
APB_1	37.975	49.367	17.0886	0	0	NaN	82.911	59.542
APB_2	61.392	61.392	21.519	0	0	NaN	78.481	78.226
Total	**69.241**	**72.278**	**14.051**	**0**	**2.278**	**33.334**	**83.671**	**85.183**



Graph 2 Absolute success rate throughout the experiment from one meter with light on and moving with the patient monitor between angles 0-45 degrees three times during the experiment

9.2.8.3 *Conclusion*

At first, from the Graph 2 you can see that regular success rate of the exposed patient monitor is between 60 and 100 percent. If the monitor is moving success rate, fall down. That trend occurs three times during the simulation.

As you can see in the Table 14, parameter SpO2 has candidates for digits for the whole simulation. However, almost same amount of NoText for HR and Pulse, fall for SpO2 in a difference by stored masks, because candidates did not have sufficient quality for Tesseract.

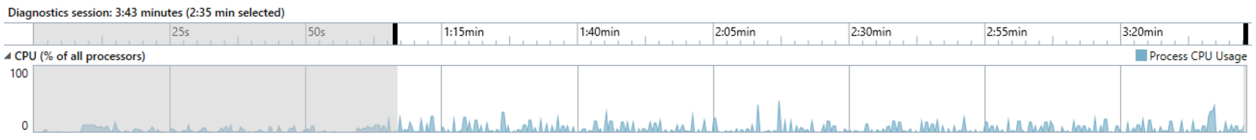
9.3 PERFORMANCE

All experiments was tested on personal computer with CPU Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz (8CPUs) and 16384MB ram. This processor has performance equal to 4.35 GFLOPS per core [76]. In upcoming graphs, you can see the CPU load and memory usage in different situations.

9.3.1 One meter – light on

Performance shown below was tested on experiment similar to 9.2.1.

9.3.1.1 *CPU load*

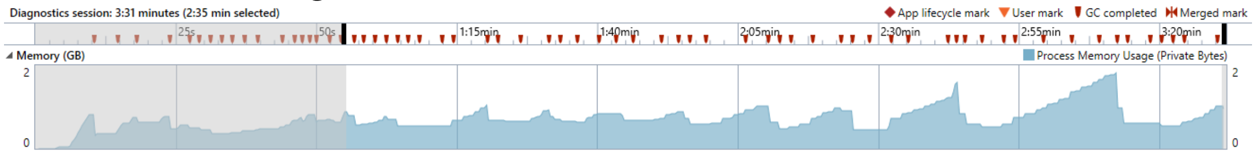


Graph 3 CPU load of all processors during the experiment from one meter with light on. Gray area is CPU load during the calibration and after the experiment

As I mentioned before, data are classified at concrete ticks. As you can see in the selected area these ticks correspond to peaks. These peaks have high between 8-30%. Their average value is approximately 18%. If no tick is detected, the usage of the application is close to zero. That means detection of change and connected processes are low CPU consuming.

CPU load for situations during the low light conditions and from different angles is quite similar.

9.3.1.2 *Memory usage*



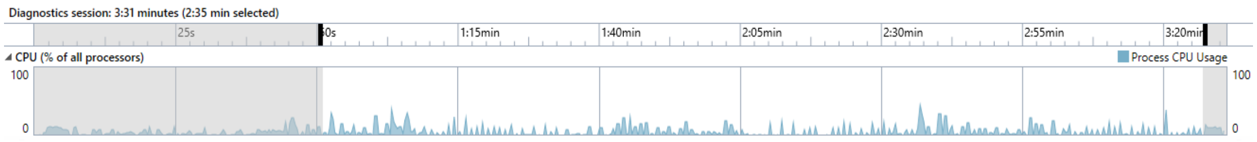
Graph 4 Memory usage for experiment from one meter with light on. Gray area is memory usage during the calibration and after the experiment

In the Graph 4, you can see that the amount of memory used during the acquisition process is relatively high. However, when we look at the documentation of a C# garbage collector we can found that *Garbage collection occurs when the system has low physical memory* [73]. That means if we have less psychical memory on our system, the garbage collector tries to solve this problem by the suitable calling of itself. Because of that, I do not show memory usage for following performance tests.

All usage of the garbage collector are shown in the Graph 4 on the time axis above the memory usage by a red mark.

9.3.2 One meter – light on – cover

Performance shown below was tested on experiment similar to 9.2.1. Time of covering is 1:20-1:40 and 1:50-2:10.

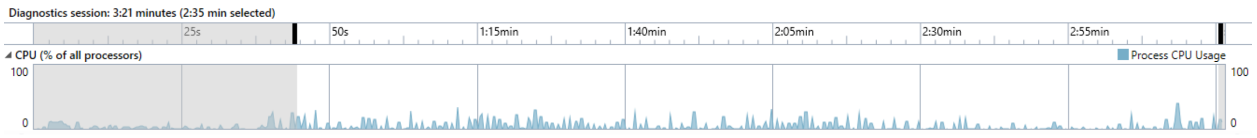


Graph 5 CPU load of all processors during the experiment from one meter with light on and covering of patient monitor screen two times during the experiment. Gray area is CPU load during the calibration and after the experiment

In the Graph 5 above, you can see that there is not any magnificent difference in CPU load while the monitor is covered by something. Peaks have similar height as in case 9.3.1.

9.3.3 One meter – light on – move

The performance shown below was tested on experiment similar to 9.2.8. Time of the movement is 1:10-1:30 and 2:00-2:20.



Graph 6 CPU load of all processors during the experiment from one meter with light on and moving patient monitor two times during the experiment. Gray area is CPU load during the calibration and after the experiment

From the Graph 6 we can see that peaks have similar height as in previous cases 9.3.1 and 9.3.2. When the user moves with the patient monitor, no magnificent difference in CPU load appears. Only peaks are a little bit wider, but no so significantly.

10 CONCLUSION

In this thesis, I at first look at current solutions of data acquisition from the patient monitor. Nowadays there are more approaches how to extract data from the patient monitor. First of them is memory inside the patient monitor. This alternative is most comfortable, but only some of new patient monitors have this possibility. The second possible solution is a connection of the cable to the patient monitor interface. These solutions are usually commercial and have appropriate certification, but there are also some non-commercial solutions. The problem with these solutions is a legal aspect because all connected devices to medical devices need to be used in accordance with supplied manual, at least in the Czech Republic. Manufacturers usually do not include connection of third party devices to their patient monitors in their manuals. The last solution is the optical extraction of data. There are two possible ways how to acquire data from the optical video signal. First of them is via video grabbers connected to the patient monitor. The advantage of this solution is no distortion of the video signal and disadvantage is the mentioned legal aspect. The last way how to acquire data absolutely noninvasively is via camera directed to the patient monitor screen. This solution does not break any law, but video signal is affected by different kinds of distortion.

In this thesis, I designed two approaches how to acquire data, especially digits, from the patient monitor via camera. As a final approach which was implemented as a standalone application, was selected digits based approach. This approach was primary selected because it can be used for data acquisition when light conditions in the room, where the patient monitor is located, is low. Compared to that monitor based approach needs distinguishable borders of the patient monitor screen during the whole acquisition process. These borders are in case of low light conditions in the room unrecognizable.

During the design of both approaches was mentioned and tested several methods. All of these methods were applied to scenes with the patient monitor. After that, I discuss the suitability of these methods for data acquisition from the patient monitor. For the implemented approach were selected best suitable methods. These methods give us best results in the concrete step of the approach, or it is the best solution for used programming language and libraries.

The whole process starts with the rectangular selection of all regions of interest by the user. This selection is recalculated from coordinates on PC screen to actual coordinates in a captured scene. Afterward, the user selects iterative all regions of interest on the patient monitor screen. On all these regions is applied conversion to gray by luminosity method, segmentation by locally adaptive thresholding by Bradley + Wellner and connected-component labeling. After that, the user is prompted to select a candidate for a digit inside the ROI. As next, the application tries to find all possible candidates similar to the selected one on the same line. At the end of the calibration process, software store information about selected ROIs and digits in it.

During the actual data acquisition process, the application waiting until difference inside any bounding box, with stored data, against the newly captured contain of this bounding box, does not exceed defined threshold. If the difference exceeds this threshold, the application redefines size of the region of interest and bounding box and store a new contain of it. Classification is done every time after a user defined time i.e. at the tick. At this tick, the application tries to classify a contain of each stored bounding box by Tesseract. If Tesseract fails, the application tries to use difference with stored masks as a classifier. If this method also fails, the result is equal to NaN for the concrete bounding box.

Data acquisition from the patient monitor via camera is a suitable solution, but there are few conditions which must be met. At first, it is a resolution of obtained digits. If we do not have appropriate resolution and separable digits, the application does not work well. This can be achieved by different things. First of them is a small distance from the camera to the patient monitor. As you can see in experiments from one meter, the success rates of these tests are relatively high. In fact, that means camera should be placed on a tripod close to the patient monitor, but this is a bad solution because tripod will bother personnel during the work. Another solution is a handle mounted straight on the patient monitor. This handle will capture data from the top of the patient monitor. The second possible way how to achieve appropriate resolution is to use a high-quality camera. In that case, we can capture patient monitor from a distance. In my opinion, if we spend money on high-quality camera we can spend them rather on video grabbers. Obtained video signal from them is not affected by any distortion. That results in better quality of classification. With a small recode of the implemented application, the application can be applied to the video signal obtained by grabbers, and we get universal acquisition tool. However, it is important to mention legal aspects, because grabbers are physically connected to the patient monitors.

The application was also tested for classical situations in the room where the monitor is located. These situations are covering of the patient monitor, moving with monitor and moving with the monitor, while something covers it. Covering of the monitor works quite well, unless the thing which covers the patient monitor, does not contain candidates for digits. The second situation is solved relatively well if movement with the monitor is gentle. The last situation is solved in the beta backup process. This process has several problems and in real situations will not work well. Because of that, I suggested certain extensions.

First mentioned extension is a combination of both designed approaches in one because each of them is better in some situations. For example the monitor based approach has no problem with the movement of the monitor while something covers it, but has higher CPU load. Another extension is focused on curves acquisition. Curves are an important part of data shown on the patient monitor, because of that I discuss their characteristics, problems and a possible approach how to acquire them. The last big mentioned extension is the multi-camera approach. Each camera connected to this system will work on individual thread and results of classifiers will be combined. This causes higher robustness of the entire application. In the end, other small extensions was mentioned like: separation of candidates, communication along the network and detection of separators.

In the end, data acquisition from the patient monitor via the camera is possible and suitable solution, but the results are highly dependent on the quality of captured video and situation in the scene. Because of that, it is important to consider what is important for the concrete user. If its quality of acquired data it is better to use commercial certified solutions. These solutions are also appropriate to make patient based decisions. However, If acquired data are for academic use and we have a low budget, implemented solution is a good alternative.

11 REFERENCES

[1] Zákon č. 268/2014 Sb., Zákon o zdravotnických prostředcích a o změně zákona č. 634/2004 Sb., o správních poplatcích, ve znění pozdějších předpisů

[2] Bionet BM5 (2012) Available at: http://www.kardiobtl.cz/download/1322666993_a640/BM5.jpg (Accessed: 3 January 2017).

[3] Promos spol. S r.o. - iM20 (2016) Available at: http://www.promos-vm.cz/im20.php (Accessed: 3 January 2017).

[4] technika, B. zdravotnická (2016) BTL získala certifikaci ISO 9001 a 13485. Available at: http://www.kardiobtl.cz/produkty/pacientske-monitory/bionet-bm3-plus/ (Accessed: 3 January 2017).

[5] ARO - DIP | ARO (2015) Available at: http://aro.fnplzen.cz/sites/default/files/users/aro/IMG_6771.jpg (Accessed: 3 January 2017).

[6] N, K.P. (2016) View details of Philips IntelliBridge enterprise. Available at: http://www.usa.philips.com/healthcare/product/HC830070/intellibridge-enterprise-interoperability-solution (Accessed: 3 January 2017).

[7] MediCollector (2014) Record bedside patient monitor vital signs data. Available at: http://www.medicollector.com/ (Accessed: 3 January 2017).

[8] Guo, F. and Loparo, K.A. (2014) Development of the real-time data acquisition system for Philips patient. Available at: https://etd.ohiolink.edu/!etd.send_file?accession=case1405963966&disposition=inline (Accessed: 3 January 2017).

[9] Rockstroh, M., Wittig, M., Franke, S., Meixensberger, J. and Neumuth, T. (2015) Video-based detection of device interaction in the operating room, Biomedizinische Technik. Biomedical engineering, 61(5), pp. 567–576.

[10] Copyright (2002) Gynecological ultrasound video capture and storage. Available at: https://www.epiphan.com/solutions/high-resolution-ultrasound-video-capture-using-epiphan-video-grabbers/ (Accessed: 3 January 2017).

[11] GitHub (2016) Tesseract-ocr. Available at: https://github.com/tesseract-ocr (Accessed: 3 January 2017).

[12] tesseract-ocr (2016) Tesseract-ocr/tesseract. Available at: https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality (Accessed: 3 January 2017).

[13] Collins, Robert (2015) Lecture 6: Harris Corner Detector, CSE486, Penn State.

[14] Parks, D., Gravel, J. (2002) Corner Detectors, Available at: http://www.cim.mcgill.ca/~dparks/CornerDetector/index.htm (Accessed: 8 December 2016)

[15] HÝNA, Petr. Detekce rohů v obraze. [s.l.], 2007. 54 s. VUT Brno. Bakalářská práce.

[16] MathWorks, T. (1998) Makers of MATLAB and Simulink - MATLAB & Simulink. Available at: http://www.mathworks.com/ (Accessed: 3 January 2017).

[17] 09gr820 (2009) Line detection by Hough transformation. Available at: http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09.pdf (Accessed: 3 January 2017).

[18] J. Matthews., L. G. Roberts (2002) An introduction to edge detection: The sobel edge detector. Available at: http://www.generation5.org/content/2002/im01.asp (Accessed: 6 December 2016)

[19] Bradley, D. and Roth, G. (2007) Adaptive Thresholding using the integral image, Journal of Graphics, GPU, and Game Tools, 12(2), pp. 13–21.

[20] Haralick, Robert M., and Linda G. Shapiro, Computer and Robot Vision, Volume I, Addison-Wesley, 1992, pp. 28-48.

[21] R. Hartley and A. Zisserman. Multiple View geomerty in Computer Vision. Cambridge University Press, second edition, 2003

[22] Dubrofsky, E. (2007) Homography estimation, Carleton University. Available at: https://www.cs.ubc.ca/grads/resources/thesis/May09/Dubrofsky_Elan.pdf (Accessed: 3 January 2017).

[23] Fisher, Bob (2006) Computing plane projective Transformations Available at: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/EPSRC_SSAZ/node11.html (Accessed: 3 January 2017).

[24] S. Mori, C. Y. Suen and K. Yamamoto, "Historical review of OCR research and development," in Proceedings of the IEEE, vol. 80, no. 7, pp. 1029-1058, Jul 1992.

[25] TRIER D., ØIVIND, TAXTT, TORFINN in JAIN K., ANIL (1995) Feature extraction methods for character recognition-a survey. Oslo: Department of Informatics, University of Oslo. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi= 10.1.1.51.7439&rep=rep1&type=pdf (Accessed: 9 November 2016)

[26] Neumann, L. (2010) Vyhledávání a rozpoznání textu v obrazech reálných scén, Czech technical university in Prague. Available at: https://cmp.felk.cvut.cz/awards/neumann-msc10.pdf (Accessed: 3 January 2017).

[27] Richards, J.A. (2012) Supervised classification techniques, in Remote Sensing Digital Image Analysis. Springer Nature, pp. 247–318.

[28] R. Smith. An Overview of the Tesseract OCR Engine, Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2 (2007), pp. 629-633.

[29] Zanibbi, Richard "Bayesian decision Theory" CSE486, Rochester Institute of Technology. Lecture, Available at: https://www.cs.rit.edu/~rlaz/PatternRecognition/slides/Bayesian.pdf

[30] Koski, T. (2013) Multivariate Gaussian distribution auxiliary notes for time series analysis, SF2943 spring 2013., KTH Royal Institute of Technology, Stockholm. Available at: http://www.math.kth.se/matstat/gru/sf2943/gaussvect2943.pdf (Accessed: 3 January 2017).

[31] N. Cristianini and J. Shawe-Taylor. An introduction to Support Vector Machines. Cambridge University Press, March 2000.

[32] Wang, Z. and Xue, X. (2014) Multi-class support vector machine, in Support Vector Machines Applications. Springer Nature, pp. 23–48.

[33] Bieniecki, W., Grabowski, S. and Rozenberg, W. (2007) Image Preprocessing for improving OCR accuracy, 2007 International Conference on Perspective Technologies and Methods in MEMS Design

[34] Fast, B.B., Allen, D.R., B, B., Allen and R, D. (1995) Patent US5594815 - OCR image preprocessing method for image enhancement of scanned documents. Available at: https://www.google.com/patents/US5594815 (Accessed: 3 January 2017).

[35] Alginahi, Y. (2010) Preprocessing techniques in character recognition, in Character Recognition. InTech.

[36] Joblove, G.H. and Greenberg, D. (1978) Color spaces for computer graphics, ACM SIGGRAPH Computer Graphics, 12(3), pp. 20–25.

[37] Color and image processing (2003) Available at: http://courses.cs.vt.edu/~cs4624/s98/sspace/imgproc/ (Accessed: 3 January 2017).

[38] Forssén, P.-E. (2011) Maximally stable Colour regions for recognition and matching. Available at: http://www.cs.ubc.ca/~perfo/papers/forssen_cvpr07.pdf (Accessed: 3 January 2017).

[39] Kristensen, F., Nilsson, P. and Öwall, V. (2006) Background segmentation beyond RGB, 3852, pp. 602–612. doi: http//dx..org/10.1007/11612704_60.

[40] Converting from RGB to HSV (2005), University of Illinois at Urbana-Champaign. Available at: http://coecsl.ece.illinois.edu/ge423/spring05/group8/finalproject/hsv_writeup.pdf (Accessed: 3 January 2017).

[41] Ford, Adrian, Roberts, Alan (1998) Colour space conversions. Available at: http://www.poynton.com/PDFs/coloureq.pdf

[42] Jeon, G. (2013) Measuring and comparison of edge detectors in color spaces, International Journal of Control and Automation, 6(5), pp. 21–30.

[43] (2013) Miscellaneous image Transformations — OpenCV 2.4.13.2 documentation. Available at: http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html#void%20cvtColor%28InputArray%20src,%20OutputArray%20dst,%20int%20code,%20int%20dstCn%29 (Accessed: 3 January 2017).

[44] Kanan C, Cottrell GW (2012) Color-to-Grayscale: Does the Method Matter in Image Recognition? PLoS ONE 7(1): e29740.

[45] Enchance contrast using histogram equalization - MATLAB histeq (2012) Available at: https://www.mathworks.com/help/images/ref/histeq.html?searchHighlight=histeq&s_tid=doc_srchtitle (Accessed: 3 January 2017).

[46] Girod, Bernd (2013) Linear image processing and filtering, Stanford university. Available at: http://web.stanford.edu/class/ee368/Handouts/Lectures/2014_Spring/Combined_Slides/8-Linear-Image-Processing-Combined.pdf (Accessed: 3 January 2017).

[47] Gvynn, Earl, F. (2017) Fourier analysis and image processing, Stowels institute for medical research. Available at: http://research.stowers-institute.org/efg/Report/FourierAnalysis.pdf (Accessed: 3 January 2017).

[48] Corso, Jason (2014) Linear filters and image processing, University of Michigan. Available at: https://web.eecs.umich.edu/~jjcorso/t/598F14/files/lecture_0924_filtering.pdf (Accessed: 3 January 2017).

[49] (2010) Image filtering, The University of Auckland. Available at: https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Image%20Filtering_2up.pdf (Accessed: 3 January 2017).

[50] Poppe, Ronald (2015) Segmentation, University Utrecht. Available at: http://www.cs.uu.nl/docs/vakken/ibv/reader/chapter10.pdf (Accessed: 3 January 2017).

[51] J. Matas, Robust wide-baseline stereo from maximally stable extremal regions, Image Vis. Comput., vol. 22, no. 10, pp. 761–767, Sep. 2004

[52] Greenhalgh, J. and Mirmehdi, M. (2012) Real-time detection and recognition of road traffic signs, IEEE Transactions on Intelligent Transportation Systems, 13(4), pp. 1498–1506.

[53] Detect MSER features and return MSERRegions object - MATLAB detectMSERFeatures (2012) Available at: https://www.mathworks.com/help/vision/ref/detectmserfeatures.html (Accessed: 3 January 2017).

[55] Morse, Bryan, S. (2000) Lecture4: Thresholding, Brigham Youth University. Available at: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf (Accessed: 3 January 2017).

[56] Otsu, N. (1979) A threshold selection method from gray-level Histograms, 9(1), pp. 62–66.

[57] Wellner, P.D. (1993) Adaptive Thresholding for the DigitalDesk.

[58] Emgu CV: OpenCV in .NET (C#, VB, C++ and more) (2008) Available at: http://www.emgu.com/ (Accessed: 3 January 2017).

[59] Sinha, U. (2016) Connected component Labelling: Pixel neighbourhoods and connectedness - AI Shack - tutorials for OpenCV, computer vision, deep learning, image processing, neural networks and artificial intelligence. Available at: http://www.aishack.in/tutorials/pixel-neighbourhoods-connectedness/ (Accessed: 3 January 2017).

[60] CS/BIOEN 4640 (2012) Morphological Operators, The University of Utah, Available at: http://www.coe.utah.edu/~cs4640/slides/Lecture11.pdf (Accessed: 3 January 2017).

[61] CSV, comma separated values (RFC 4180) (2012) Available at: http://www.digitalpreservation.gov/formats/fdd/fdd000323.shtml (Accessed: 3 January 2017).

[62] Bioengineering i6460 Bioelectricity (2012) The University of Utah. Available at: http://www.sci.utah.edu/~macleod/bioen/be6460/notes/W12-ECG.pdf (Accessed: 3 January 2017).

[63] Vaingast, S. (2015) Im2graph. Available at: http://www.im2graph.co.il/ (Accessed: 3 January 2017).

[64] Rohatgi, A. (2010) WebPlotDigitizer - extract data from plots, images, and maps. Available at: http://arohatgi.info/WebPlotDigitizer/ (Accessed: 3 January 2017).

[65] Epshtein, B., Wexler, Y. and Ofek, E. (2010) Stroke width transform - Microsoft research. Available at: https://www.microsoft.com/en-us/research/publication/stroke-width-transform/ (Accessed: 3 January 2017).

[66] Keogh, Emonn, Time series, University of Califormia (Accessed: 24 November 2016).

[67] McKenna, Frank, Thin-client vs fat-client computing (2002) Knowledgeone Corporation, Available at: http://www.knowledgeonecorp.com/news/pdfs/Thin%20client%20vs%20Fat%20client%20Computing.pdf (Accessed: 3 January 2017).

[68] Joseph, Davies G., Thomas, Lee F., Microsoft Windows Server 2003 TCP/IP Protocols and Services

[69] Project, M. (2014) C# compiler. Available at: http://www.mono-project.com/docs/about-mono/languages/csharp/ (Accessed: 3 January 2017).

[70] Raspberry Pi (2017) in Wikipedia. Available at: https://en.wikipedia.org/wiki/Raspberry_Pi (Accessed: 3 January 2017).

[71] DirectShowNet library (2005) Available at: http://directshownet.sourceforge.net/ (Accessed: 3 January 2017).

[72] Medical, L. (2010) Products and promotions. Available at: http://www.laerdal.com/ (Accessed: 3 January 2017).

[73] Microsoft (2011) Fundamentals of garbage collection. Available at: https://msdn.microsoft.com/en-us/library/ee787088(v=vs.110).aspx (Accessed: 3 January 2017).

[74] Gibbs, Samuel (2015-02-18). Raspberry Pi becomes best selling British computer, The Guardian. (Retrieved : 28 December 2016).

[75] Ltd, P.F. (2009) A comprehensive raspberry pi 3 benchmark. Available at: https://www.element14.com/community/community/raspberry-pi/blog/2016/02/29/the-most-comprehensive-raspberry-pi-comparison-benchmark-ever (Accessed: 4 January 2017).

[76] Sebastien (2015) CPU performance. Available at: http://wuprop.boinc-af.org/cpu_list.php (Accessed: 4 January 2017).

[77] technika, B. zdravotnická (2012) BTL získala certifikaci ISO 9001 a 13485. Available at: http://www.kardiobtl.cz/produkty/pacientske-monitory/bionet-bm3/ (Accessed: 5 January 2017).

[78] EDAN M3B ‘NEW’ CAPNOGRAPH MONITOR (2014) Available at: http://stores.portlandsurgicalsales.com/edan-m3b-new-capnograph-monitor/ (Accessed: 5 January 2017).

[79] Emmerich, M. (2013) Imko GmbH - Medizintechnik. Available at: http://www.imko-med.de/leistungen/projektgeschaeft/patientenueberwachung.php?lang=de (Accessed: 5 January 2017).

[80] AM-2000H 12.1 inch patient monitor - 东辰科技有限公司 (2010) Available at: http://www.advance-med.com/ProductView.asp?ID=50 (Accessed: 5 January 2017).