České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Jan J e ř á b e k

Studijní program: Kybernetika a robotika (magisterský)

Obor: Robotika

Název tématu: Online rozpoznávání obličeje v radiometrické sekvenci z termokamery

Pokyny pro vypracování:

1. Seznamte se s problematikou bezdotykového měření teploty a s multispektrálním bikamerovým systémem Thermal Vision PRO společnosti Workswell pro zobrazování termovizního a viditelného 2D obrazu.
2. Navrhněte a v LabVIEW implementujte algoritmus online rozpoznávání obličeje v radiometrickém obraze z termovizního senzoru. Stanovte povrchovou teplotu obličeje a diskutujte nejistoty měření. Navrhněte postup pro snížení nejistoty měření a dle výsledků zvolte nejvhodnější místa měření teploty dle úhlu natočení obličeje. Diskutujte možnost umístění černého tělesa v měřené scéně.
3. Navrhněte a implementujte algoritmus pro fúzní zobrazení obou obrazů (termovizního a viditelného spektra) a jednotlivé scény zkombinujte do jediné. Diskutujte možné kombinace zobrazování (termovizní na viditelném, reálný na termovizním) vzhledem k co nejefektivnějšímu odhalení konkrétní osoby v měřené scéně (více osob ve scéně). Lokalizované osoby označte ve výsledném fúzním obraze a zaznamenejte jejich teplotní údaje včetně fotografie do tabulky. Maximalizujte přesnost překryvů obrazů a minimalizujte negativní důsledky fúze vlivem paralaxního umístění senzorů a rozdílných pozorovacích úhlů objektivů.
4. Diskutujte možnost vylepšení algoritmu využitím dodatečného příznaku, tj. povrchové teploty lidské tváře a optimalizujte tak výslednou homografii, zaměřte se přednostně na oční oblasti.
5. Navrhněte v LabVIEW prototyp vyhodnocovací aplikace pro praktické nasazení celého systému. Zaměřte se především na rychlé zpracování obrazových dat, jejich fúzní transformaci a přehledné uživatelské rozhraní pro operátora.
6. Implementujte do prototypové aplikace vhodný síťový protokol (video server) pro sdílení fúzního obrazu po síti TCP/IP. Zaměřte se na protokoly MPEG-4, H.264, RTSP a M-JPEG. Vyzkoušejte naimplementovaný video server zobrazit na více klientských stanicích.

Seznam odborné literatury:

[1] Stan Z. Li: Handbook of Face Recognition, Springer 2011
[2] Hlaváč V., Sedláček M.: Zpracování signálů a obrazů, skriptum, Vydavatelství ČVUT 2009.
[3] Bress, T. J.: Effective LabVIEW Programming, Tom Robbins, 2013, ISBN: 978-1-934891-08-7
[4] Vollmer M.: Infrared Thermal Imaging: Fundamentals, Research and Applications, 2010

Vedoucí diplomové práce: Ing. Jan Kovář

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 3. 2. 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Jan J e ř á b e k

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Diploma Thesis: Online Face Recognition in Radiometric Sequence from Thermal Camera

Guidelines:

1. Study contactless temperature measurement theory and get familiar with Thermal Vision PRO multispectral bicamera system made by Workswell Corporation.
2. Design and implement a Labview algorithm for real-time temperature face measurement. Determine face temperatures and discuss measurement uncertainty. Find a way to reduce this measurement uncertainty and choose the best location for the temperature measurement accordingly to the results. Discuss a presence of the black body in the measured scene.
3. Design and implement fusion algorithm for thermal and visible images (combination of two scenes into one). Discuss the best fusion algorithm according to efficient image interpretation and high temperature alarm recognition. Highlight suspicious persons in real-time and store their photos with temperature values.
4. Discuss the algorithm improvement using the additional feature, i.e. facial temperature and optimize the resulting homography. Focus primarily on the eyes region.
5. Design a special Labview application for practical deployment of the face detection temperature system. Focus mainly on the fast image processing, fusion image transformation and user friendly graphical interface.
6. Implement an appropriate network protocol (video server) for sharing the resulting images via TCP/IP network. Focus on MPEG-4, H.264, RTSP and M-JPEG. Try to test this video server on multiple software clients.

Bibliography/Sources:

[1] Stan Z. Li: Handbook of Face Recognition, Springer 2011
[2] Hlaváč V., Sedláček M.: Zpracování signálů a obrazů, skriptum, Vydavatelství ČVUT 2009.
[3] Bress, T. J.: Effective LabVIEW Programming, Tom Robbins, 2013, ISBN: 978-1-934891-08-7
[4] Vollmer M.: Infrared Thermal Imaging: Fundamentals, Research and Applications, 2010

Diploma Thesis Supervisor: Ing. Jan Kovář

Valid until: the end of the summer semester of academic year 2015/2016

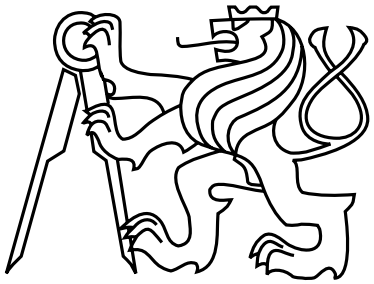
L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 3, 2015

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics



Master's Thesis

Online face recognition in radiometric sequence from thermal camera

Jeřábek Jan

Supervisor: Ing. Jan Kovář

Study Program: Cybernetics and Robotics, Master

Field of Study: Robotics

May 5, 2015

Poděkování

Rád bych zde poděkoval vedoucímu diplomové práce Ing. Janu Kovářovi za jeho rady, věcné připomínky a vstřícný přístup během zpracování této práce. Dále bych rád poděkoval kolegům a společnosti Workswell za poskytnutí potřebného zázemí a vybavení.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 5. 5. 2015 .....

Abstrakt

Tato práce se zabývá detekcí osob se zvýšenou teplotou pomocí bezkontaktního měření teploty s využitím termokamery. Přináší algoritmus pro vyhledávání obličejů v radiometrickém obraze, které následně zobrazuje v obraze kamery s viditelným spektrem. Algoritmus pro detekci obličejů je založený na korelaci vzoru pro detekci s částí radiometrického obrazu.

Osoby jsou v radiometrickém obraze detekovány s úspěšností větší než 95% a následně s vysokou přesností tansformovány do viditelného spektra. Tato práce tak přináší ucelené a přehledné řešení, které je možné nasadit v nejrůznějších prostředích, a poskytuje tak rychlou a efektivní metodu pro detekování nemocných osob.

Abstract

This paper brings a method for detecting persons with elevated temperature by application of remote temperature measurement with use of thermal camera. It employs facial recognition algorithm operating in radiometric image, which is then translated into the visible spectrum. The algorithm is based on correlation of the pattern for detection with parts of the radiometric image.

Individuals are detected in the radiometric image with success rate greater than 95% and translated with high accuracy into the visible spectrum image. This work establishes a complete, user-friendly solution, which can be deployed to various environments and provides quick and efficient means of detection of potentially sick people.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Today, we live in the most spectacular age. The ever accelerating pace of technological development has made mankind both more resilient and more fragile at the same time on many different levels. While we have made some truly astounding breakthroughs in the fields of medicine and disease prevention, the technological leaps in other fields – namely our capability to sustain dense urban populations and our ever more widely accessible ability to quickly travel between those - have also increased the probability of a potential pandemic outbreak scenario.

It is no longer inconceivable, that certain resilient bacterial or viral diseases, such as Ebola Virus Disease, Bolivian hemorrhagic fever, or mutations thereof, could under right conditions spread quickly and potentially overwhelm entire regions of the world, especially in less developed countries.

The recent Ebola outbreak in West Africa should have been a wake up call to everyone. For various economical and technological reasons, there is, as of today, no effective and accessible cure. Sadly, the only way to shield the healthy is to identify the sick and isolate them, ideally before they cross whichever border that sets them apart from the "unaffected" areas. Airports and other dense transportation hubs are critical points of interest for anyone, who would like stop or slow down the pending contagion. One of the symptoms, which are common for many of the potentially pandemic-able diseases is elevated temperature.

This work is an attempt to use image recognition techniques in combination with thermal cameras, in order to automatically measure temperatures of passengers in traffic hubs, or in any other dense population areas. In theory, all that is required for the unobtrusive measurement is that the person appears in the camera's field of view, facing it in a certain angle range. With use of multiple cameras, it should be possible to cover the vast majority of the passengers. This solution could be one of uncounted others, which, if used properly together, might just tip the scales and save some lives.

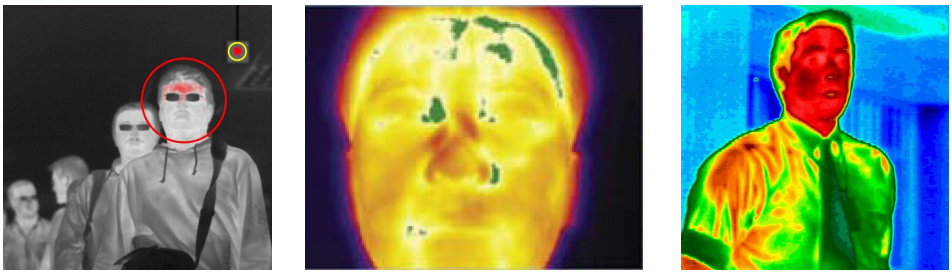
1.2 Objectives

The goal of this work is to develop an application, which is capable of detecting people with elevated temperature, even in crowded, busy environments. For this purpose, an algorithm capable of human recognition and temperature measurement in a radiometric image must be developed. It is quite easy to establish the temperature based on thermal camera images, but the data are not easily readable for humans. Thus, the detected persons must be translated into a generic visible spectrum camera image. The video output must then be accessible on client machines via TCP protocol.

For archiving purposes, it would be good to automatically store detected persons on disk along with any meta-data, such as date, time or measured temperature. The final solution should provide an user-friendly interface, which would enable the operator to easily identify the detected person and share the information with others over the network.

1.3 Current solutions

Nowadays, there are several solutions which serve for humans temperature measurement and indication whether this temperature is elevated or not. The main three of them are FevIR Scan Fever Screening System, FLIR Systems IR cameras and Human Body Temperature Monitoring System.



(a) FevIR [2] (b) FLIR [3] (c) LAND [4]

Figure 1.1: Current solutions for detecting the humans with elevated temperature, i.e. FevIR Scan Fever Screening System, FLIR Systems IR cameras and Human Body Temperature Monitoring System.

- **FevIR Scan Fever Screening System** [2]
  System from the Thermoteknix Systems Ltd company used for monitoring and detecting humans with elevated body temperature using thermocamera. When detecting a temperature higher than previously set threshold, the system sounds the alarm and marks the spot with elevated temperature. An example of the detection can be seen in the Figure 1.1a.

- **FLIR Systems IR cameras** [3]
  System from the FLIR company used for detecting elevated humans body temperature.

Each person has to stand before the thermocamera which captures an image. The system then appropriately recalculates this image and visualize the spots with elevated temperature. Measured temperature is considered as elevated, whether it is higher than previously set threshold, which is corrected by the average temperature of the last ten measured persons. An example of the detection can be seen in the Figure 1.1b.

- **Human Body Temperature Monitoring System** [4]
  System from the LAND Instruments International company used for monitoring and detecting humans with elevated body temperature using thermocamera. Similarly as with the FLIR system, the elevated temperature is visualized to the operator. An example of the detection can be seen in the Figure 1.1c.

Each of these systems identifies a person with the elevated temperature by finding a maximal value within the captured thermal image. This is relatively fast and accurate solution when set properly, but it demands a scene which does not contain any non-human objects with temperature similar to the 38°C. This paper brings a solution which searches for persons present in the captured scene first, and then determines, whether these persons have elevated temperature or not. Therefore this solution should be more robust and could be deployed in various places with minimal limitations.

Chapter 2

Contactless temperature measurement

2.1 Thermographic equation

Every substance with its temperature higher than absolute zero emits an infra-red radiation which corresponds to this temperature. The cause of this effect is the inner mechanical molecular movement whose intensity depends on the object's temperature. The molecular movement represents reallocating the charge which emits the electromagnetic radiation - photon particles. Spectrum of this radiation covers the wavelengths from $0.7\,\mu$m to $1000\,\mu$m which is mostly not visible by the naked eye. As we can see from the Stefan–Boltzmann law for the black body, the energy emitted by the black body is proportional to the fourth power of its thermodynamic temperature T

$$M = \sigma\, T^4, \tag{2.1}$$

where M is the intensity of the radiated energy and σ is the Stefan–Boltzmann constant.

The infrared radiation can be measured for example by the microbolometer. The detector's material heats up when stroked by the infrared radiation and changes its electrical resistance. This change is measured and serves for the radiometric image (thermogram) creation. Contactless temperature measurement is very fast and enables to measure a temperature of moving objects in the real time. Also, it enables the ability of measuring objects which are badly accessible or dangerous. The measurement itself does not affect the measured object or its temperature.

Equation (2.1) represents an intensity of the emitted energy by the Black body. In the real world, any object does not emit or absorb all of the energy so the parameter of emissivity ε has to be introduced [5]. Emissivity represents an ability of the current object to emit infrared energy carrying information about its temperature. Emissivity can take values from 0, which represents completely non-emitting object, to 1, which represents completely emitting object, the Black body. For non-black body objects, the equation (2.1) is in the form

$$M = \varepsilon \, \sigma \, T^4, \tag{2.2}$$

where ε is the object's emissivity.

However, the detected infrared radiation does not depend only on the object's temperature but also on the another sources of infrared radiation and on the atmospheric temperature and transmission. Therefore the overall intensity of the radiated energy M is in the form of the thermographic equation

$$M = M_z + M_a + M_{atm}, \tag{2.3}$$

with

$$\begin{aligned} M_z &= \varepsilon \, \sigma \, T_0^4 \\ M_a &= (1 - \varepsilon) \, \sigma \, T_a^4 \\ M_{atm} &= (1 - \tau_{atm}) \, \sigma \, T_{atm}^4, \end{aligned}$$

where T_0 is the object's temperature, T_a is the temperature of another sources of radiation, τ_a is the atmospheric transmission and T_{atm} is the atmospheric temperature. Then we can rewrite the thermographic equation into the form

$$M = \varepsilon \, \sigma \, T_0^4 + (1 - \varepsilon) \, \sigma \, T_a^4 + (1 - \tau_{atm}) \, \sigma \, T_{atm}^4, \tag{2.4}$$

from which we can calculate the object's temperature.

2.2 Thermocamera principle

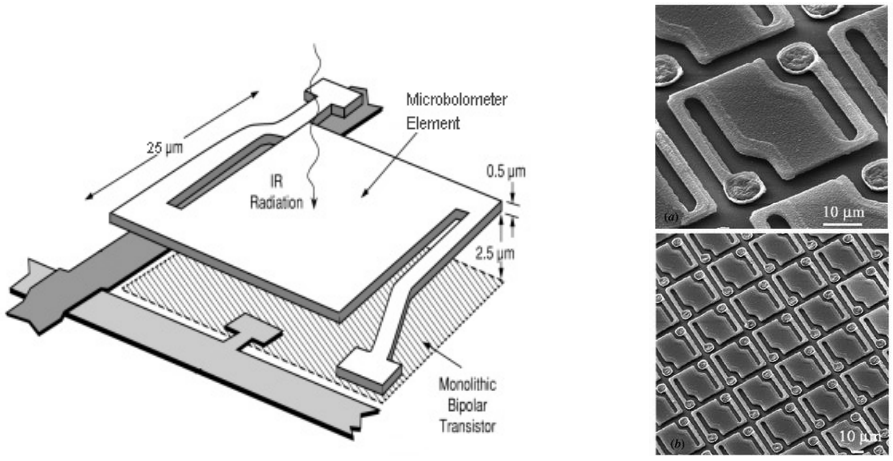
Any thermocamera consists of an objective, detecting element and evaluation electronics. An objective projects the incident thermal radiation onto the detecting element which measures its intensity. This intensity is transformed into voltage or current, digitalized and recalculated into resulting image called thermogram. Thermogram is an image consisting of pixels whose values correspond to the object's surface temperature.

- **Optics**

  An optics of most of the thermocameras consists mainly of the convex lens made from germanium without an option of the optical zoom. The lens surface is covered with an antireflexion layer which prevents reflection of the infra-red radiation. This layer also works as a filter so that the incident electromagnetic radiation of all wavelengths is filtered only to the desired ones.

- **Detector**

  An example of the infra-red radiation detector is a microbolometer. A microbolometer changes its resistance accordingly to the amount of absorbed incident infra-red radiation [6]. This amount is proportional to the temperature so the temperature can be computed from the changes of the microbolometer's resistance. Scheme of a microbolometer can be seen in the Figure 2.1a.



(a) Scheme of a microbolometer [7] (b) Photo of a microbolometer array [8]

Figure 2.1: Scheme and photos of a microbolometer and an array of microbolometers captured by an electron microscope.

The top side of a microbolometer consists of an absorption layer which absorbs the incident infra-red radiation. This layer has to be isolated from the rest of the microbolometer so that the absorption layer heats itself only from the incident radiation, not from the microbolometer itself. An infra-red detector consists of an 2-D array of microbolometers, as in the Figure 2.1b, whose amount determines the resulting thermocamera resolution. The resolution of mid-range thermocameras is 320×240 pixels, the high-end cameras can have resolution 1054×768 pixels and higher [9].

Chapter 3

Hardware and software configuration

This section describes the software used in this application, i.e. the LabVIEW environment along with the hardware, which consists of the Thermal Vision PRO [10] bi-camera housing from the Workswell company with a thermal imaging camera and visible spectrum CMOS camera.

3.1 LabVIEW

LabVIEW is a short-cut for Laboratory Virtual Instrumentation Engineering Workbench and presents a platform for developing programs in a graphical language called G, introduced by the National Instruments company [11]. Graphical programming is relatively new and unusual programming technique patented by the National Instruments company in 1990. Developers of the LabVIEW claim that program created in G language can run as fast as program written in C language [12]. LabVIEW is mostly used for interfacing, logging and data acquisition from various peripheral devices, especially from measurement tools.

The G language is a dataflow programming language so that a code execution is determined by a sequence of wire-connected nodes with different purposes and functions. Wires between nodes can propagate variables and any node can execute only when all of its input's data are available.

The biggest advantages of graphical programming are relatively short time to develop an application and its capabilities of multithreading. Multithreading can be easily programmed by wiring the appropriate nodes in parallel. Another advantage is an easy and transparent way of a graphical user interface creation.

3.2 Cameras

There are two types of cameras used in this application. The first one is a visible spectrum camera, the second one is a thermocamera.

3.2.1 Visible spectrum camera

Visible spectrum camera used in this application is a fast 1.3 Megapixel Color CMOS Camera with the USB 3.0 Interface.

This camera is compatible with the USB3 Vision standard and compliant with the USB 3.0 SuperSpeed specification. Its advantages are fast process capturing which makes it suitable for industrial automation, motion capture, machine vision, facial recognition and many other applications. Another advantage are its very small dimensions which make it possible to fit the camera into very narrow places. Some key parameters of this camera can be seen in the Table 3.1.

Parameter	**Value**
Resolution	1280 × 1024
Frame Rate	60 Hz
Image Data and Control Interface	USB 3.0 standard Micro B
Sensor Technology	CMOS, Global shutter
Input Power	0.9 W
Operating Temperature Range	0 °C to +50 °C
Dimensions W×H×D	26 × 26 × 26 mm

Table 3.1: Parameters of the visible spectrum camera.

3.2.2 Thermocamera

Thermocamera used in this application is a fast uncooled thermal imaging camera with the USB 3.0 Interface.

Figure 3.1: Photo of the thermocamera.

Advantages of this thermocamera are its 60 Hz frame rate, active contrast enhancement, silent shutterless NUC for continuous image improvement and many others. Some key parameters of this thermocamera can be seen in the Table 3.2.

Parameter	Value
Display Format	640 × 512
Frame Rate	30 Hz
Sensitivity	<50 mK at f/1.0
Scene Range (High Gain)	-25 °C to +135 °C
Input Power	4.0 - 6.0 VDC
Operating Temperature Range	-40 °C to +80 °C

Table 3.2: Parameters of the thermocamera.

3.2.3 Bi-camera housing

Both of the cameras are placed in the bi-camera housing called Thermal Vision PRO from the Workswell company, as can be seen in the Figure 3.2.

Figure 3.2: Photo of the bi-camera housing.

This housing ensures that the cameras are well protected from the environmental influences and especially that their position relative to each other is the same all the time.

3.3 Connection

The process of any camera to computer connection in the LabVIEW environment consists mainly of the four tasks [13], which are: *Opening*, *Configuration*, *Acquisition* and *Closing*.

- **Opening**
  In this part of the camera connection process, the most important thing is to define a camera session. Camera session is a handle to the camera and identifies the selected camera with its settings. If there are more than one cameras connected, the appropriate session name has to be chosen for each of the cameras.

- **Configuration**
  In this part, we can configure the properties of the images acquisition. Every image coming out of the camera is stored in the internal buffer in a way of Fist-In Fist-Out (FIFO). Any number of buffers can be set by the user. It is recommended to use three or more buffers for continuous acquisition, especially if the camera is connected via the Ethernet interface. The more internal buffers, the more reliability of the data transfer. The size of each buffer is determined by the size of the raw data coming out of the camera. If we do not want continuous acquisition but only a single-shot acquisition, we have to set the number of internal buffers accordingly to the desired number of images to receive.

- **Acquisition**
  In this part, the images stored in the internal buffers are copied to the user buffer for further processing. When creating the user buffer, it is needed to allocate memory for the image according to its desired type. Also the user buffer has to be destroyed when the acquisition is done, unlike the internal buffers which are destroyed automatically.

- **Closing**
  The last part closes the camera session when the acquisition is done.

Figure 3.3 shows a block diagram with the camera acquisition flow. Its sections are the same as described before. First, the session name is chosen and the camera can be opened and configured along with setting the user buffer. Then, the block *Grab* is called in a *while-loop* until the *stop* condition is met, which serves for the images acquisition. Finally the camera session is closed and user buffers are destroyed.

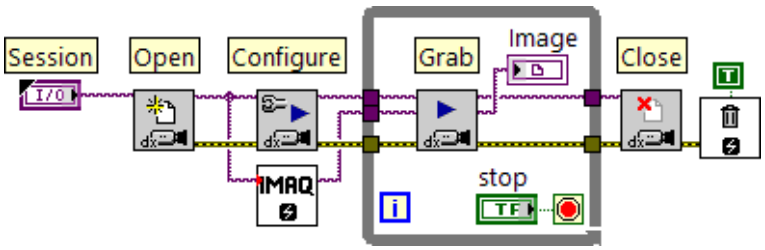


Figure 3.3: Block diagram for connecting a camera, acquiring images and closing the camera.

3.3.1 Visible spectrum camera

To successfully acquire images with the visible spectrum camera in LabVIEW, there is a need for the special library. This library is a wrapper of the *.NET* based *xiAPI.NET* used in LabVIEW and contains all necessary LabVIEW functions for acquiring images, instead of *C++* or *.NET*. The connection of the visible spectrum camera to the computer in LabVIEW is similar to the general camera connection but with some changes.

1. **Create**
   This function serves for searching the computer interface for connected cameras, and creates a unique reference to all of the connected cameras.

2. **Open Device**
   This function serves for opening the connected camera by the *Device ID* and creates a unique reference to the camera. Number 0 indicates that the first camera of the connected ones will be used.

3. **Load from INI**
   If we do not want to set the cameras parameters manually, we can load a configuration *INI* file, which contains predefined parameters for the desired camera's behavior.

4. **Start Acquisition**
   This function serves for preparing and starting the images acquisition from the camera.

5. **Get Image**
   This function acquires images data and returns them in a form of byte array.

6. **Convert Image**
   This function serves for converting the images data from the byte array to the LabVIEW image.

7. **Stop Acquisition**
   This function terminates the images acquisition from the camera.

8. **Close Device**
   The last function releases allocated resources and releases the connected camera references.

Figure 3.4 shows a block diagram with the visible spectrum camera acquisition flow. Its sections are the same as described before.

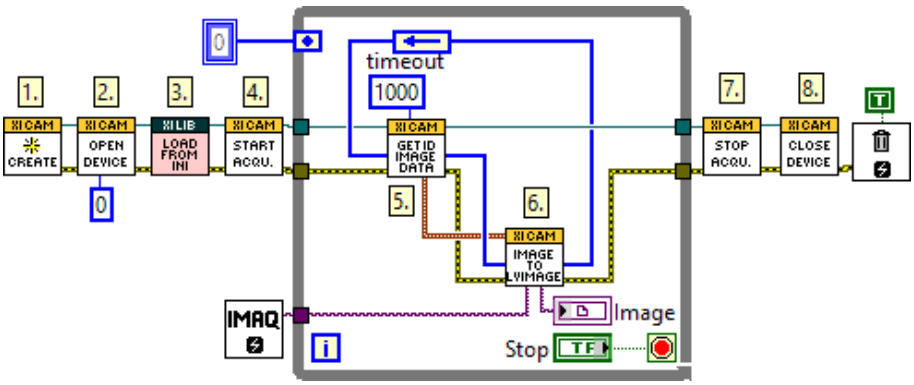


Figure 3.4: Block diagram for connecting the visible spectrum camera, acquiring images and closing the camera.

3.3.2 Thermocamera

Connection the thermocamera and acquiring its images is similar to the visible spectrum camera. The difference is in the library which is necessary for the thermocamera's connection and images acquisition. In case of the FLIR thermocameras, it is the eBus library [14]. All steps needed for camera connection, configuration, images acquisition and camera disconnection are the same as in the visible spectrum camera case.

3.4 Parameters

The user should be able to set up several parameters which are important for the correct camera's behavior. This behavior is dependent on the environmental conditions and their incorrect setting can cause misinterpretation of the measured temperatures.

1. **Emissivity** - As mentioned before, setting the correct value of emissivity is crucial for the correct temperature measurement. In an extreme case, the incorrect value of the emissivity can cause that the human temperature appears to be nearly 100 °C [15].
2. **IR Format** - temperature resolution of each pixel - 10 mK, 100 mK or radiometric. Its misinterpretation can cause an error in the temperature evaluation.
3. **Camera focus** - any image should be as sharp as possible to ensure the correct temperature readings [15].
4. **Manual range** - settings of the maximal and minimal temperature range can, in certain situations, make the image more readable.

Chapter 4

Main application

This chapter describes the main application's structure and some of its key features. It also shows the steps needed for the thermal images preprocessing so that the images are suitable for the face detection.

4.1 Application structure

The main application consists of several independently running parallel loops and its block diagram is shown in the Figure 4.1.

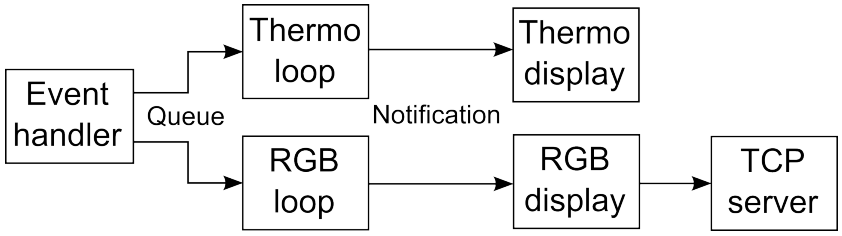


Figure 4.1: Block diagram showing the main application structure.

The first loop is the event handler which handles interaction between the user and the application. When an event occurs, e.g. a mouse click, the queue element with appropriate information about the event is enqueued and sent to the *Thermo* or *RGB* loop for further processing. *Thermo* and *RGB* loops are independently grabbing images from the connected cameras and these images are processed as required by the user. These processed images are sent to the displaying loops via the notifications and to the *TCP server* loop which sends them over TCP protocol to the network.

4.2 Thermal images preprocessing

For better results of the face detecting algorithm and for better readability of the thermo-camera video output, every image has to be preprocessed and its pixel values have to be recalculated appropriately.

4.2.1 Palettes

Sometimes, the data from the thermocamera are not clear for the human user and it can be useful to recalculate them into different color palettes. These palettes can point out some regions of interest in the image or display parts of the image which were not visible before. Recalculate color palette means that for each pixel value in the grayscale image, i.e. from 0 to 255, there are predefined values for the red, green and blue color components. These values can be chosen by the user or there exist several well known palettes which are used most often.

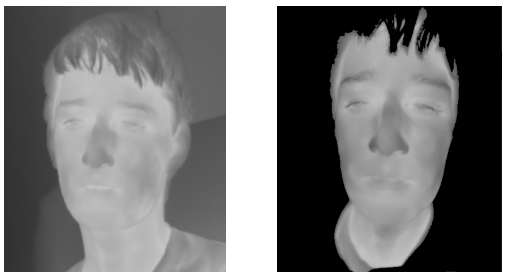
Figure 4.2 shows the same scene recalculated into three different palettes - Figure 4.2a shows the Temperature palette, 4.2b the Natural palette and 4.2c the WBRGB palette.

(a) Temperature (b) Natural (c) WBRGB

Figure 4.2: The same image recalculated with three different palettes - Temperature, Natural and WBRGB.

4.2.2 Background removal

For better accuracy of the face detection, the image's background has to be removed. The input is a thermogram with the pixel values in degrees Celsius. When the pixel value is lower than 25 °C or higher than 45 °C, it is set to the zero, i.e the black color. Example of an image with the background removed can be seen in the Figure 4.3b.



(a) Greyscale (b) No background

Figure 4.3: The same thermogram in the grayscale and with the background removed.

The background removal is very important and ensures that there are no other objects in the scene, unless their temperature is about the same as the humans. When the background is removed, the image is represented by the array of doubles so it has to be recalculated to the appropriate image format again by the equation

$$\mathrm{img}(x, y) = \frac{(\mathrm{px}(x, y) - \min) \cdot 255}{\max - \min}, \tag{4.1}$$

where max and min is maximal and minimal pixel value respectively. In our case min = 25 °C and max = 45 °C. $px(x, y)$ is the pixel value at coordinates x and y, $img(x, y)$ is the recalculated pixel value at coordinates x and y.

4.2.3 Black body temperature correction

Each thermocamera can have a certain temperature measurement error which is same for all of the pixels, say camera offset. This error can be reduced using the Black body. Black body is a body which absorbs all of the radiation of all of the wavelengths falling onto its surface. Industrially made Black bodies are made for certain temperature, for example 45 °C. By measuring the temperature of this Black body with the thermocamera, we can determine the temperature offset. Then with measuring the human's temperature, which is around 37 °C, we can modify it by the calculated offset. This means that the Black body has to be present in the detected scene, as shown in the Figure 4.4.

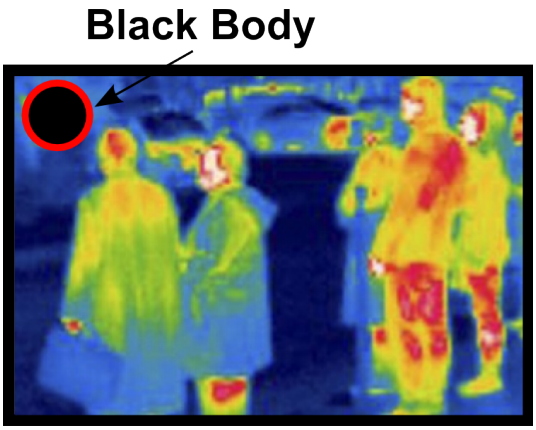


Figure 4.4: An example of a scene with the Black body in the left top corner.

An user can click the point where is the Black body's location. In the ±2 px surroundings of this point, the average value is calculated and then the temperature error can be reduced.

4.3 Application features

4.3.1 Cameras calibration

As mentioned before, there are two cameras used in this application. The first one is a visible spectrum camera and the second one is a thermocamera, which are placed in the bi-camera housing. We want to display positions of certain pixels in the therogram onto the visible spectrum image, which can be done by using homography, as described in the next chapter. For calculating the homography, we need to have at least four corresponding pairs of points in both of the images. When an user wants to calibrate the cameras, he can click four times in each image to the corresponding pairs of points and the correct homography for pixel's position is calculated.

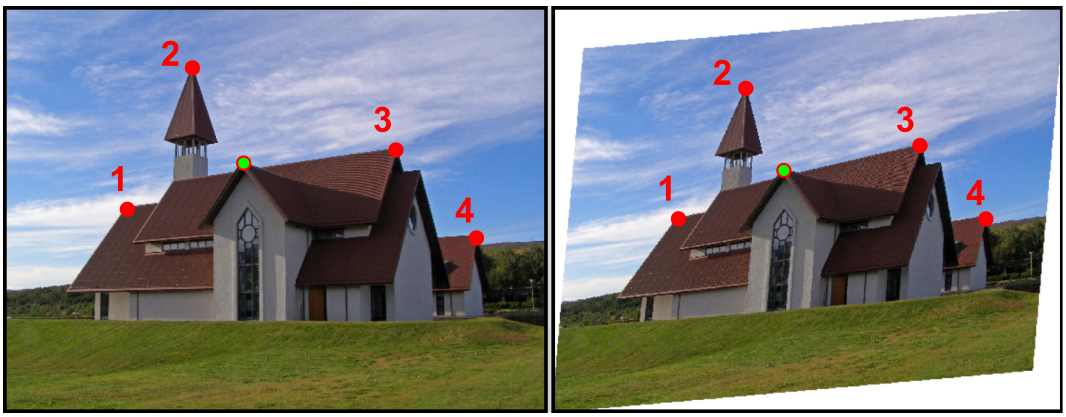


Figure 4.5: An example of the matrix of homography creation.

Figure 4.5 shows an example of the matrix of homography creation. The right image is the same as the left one but with the skew applied. It is needed to click four corresponding points in each image, as marked with the red dots in the Figure 4.5. Then, the correct coordinates of the point clicked in the left image can be computed and marked in the right image, as illustrated with the green dots.

4.3.2 Saving photos

All of the detected objects are sorted in descending order by their temperature. The photos of the two objects with the highest temperature are displayed separately in the warning box, assuming that their temperature is higher than the set threshold. Position of these two objects is tracked in the scene so the displayed photos can be those with the best match. In another words, the photo of the detected object is changed only if the newly detected object has a better matching score than the previous one. The photo is a cropped visible spectrum image by the bounding box of the detected object in the thermal image.

The user can easily click the Save button to save the photos of the two detected ill persons with highest temperatures or click the Cancel button to discard the current displayed photo. This photo should be the best one for the period during the object moved in the scene. The

Figure 4.6 shows an example of two face detections with their temperature higher than the threshold set to 36°C. When the Save or Cancel button is clicked, the corresponding photo is saved to the hard drive or discarded respectively.

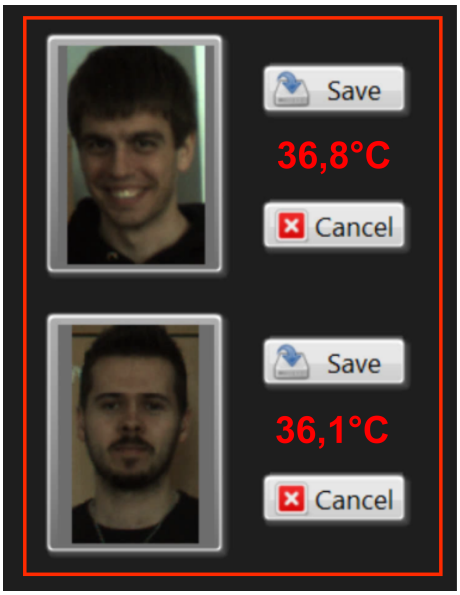


Figure 4.6: An example of two face detections with their temperature higher than the threshold set to 36°C.

4.3.3 Video server

Each processed visible spectrum image is sent via notification to the TCP video server loop. TCP video server loop is a loop running independently in parallel sending the images via the TCP protocol. TCP video server is in a form of state machine with four states which are: *Initialize*, *Status check*, *Send data* and *Stop*.

- **Initialize** - This state creates a listener reference that is used to check for new client TCP network connections. Its inputs are a port number on which we want to listen for connection, a net address which specifies on which network address to listen and a timeout which sets a time that the function waits until finishes and returns an error. In another words, the timeout is a time in which a client has to connect to the server, or it returns an error.
- **Status check** - This state checks whether or not there are any new clients connected that need data. If there are any client connections, the state machine goes to the next state, i.e. *Send data*. Otherwise it stays in the *Status check* state. Any new client connections are stored in an array of connections for further processing.
- **Send data** - First, this state reads a byte from all clients in the array of connections. A non error answer indicates, that the client has closed the connection and so it is

removed from the clients array. If the TCP read function times out, the current data can be send to the clients. TCP write function writes number of data bytes along with the actual data. The data itself consists of the visible spectrum image with overlaid objects detections and their temperature.

- **Stop** - This state closes all open client connections stored in the array of connections as well as the listener connection and stops the entire *TCP Server* loop.

4.3.4 Video client

This separate application serves for reading and imaging data sent by the *TCP server*. It opens a TCP connection with given port and IP address. Then the TCP *Read* function acquires the size of the data and the second TCP *Read* function reads the data itself. When there is an error or the user presses the stop button, the TCP *Write* function sends a character to the *TCP Server* to indicate that the client has stopped. After this, TCP *Close* function closes the connection and the client application is stopped.

Sending a large number of images to multiple clients can congest the network. Therefore these images have to be compressed to the appropriate format. Most commonly used formats for the video compression are MPEG-4 AVC/H.264 or for the separate images M-JPEG [16].

- **MPEG family**
  The main principle of the MPEG family codecs is that the video is divided into separate frames [17]. These frames are split into groups of approximately 12 frames in which the first frame is called Intra Frame. This Intra Frame is compressed as an ordinary image in the JPEG format and is divided into several macroblocks. Brightness, blue and red components are extracted from each of the macroblocks for further processing which significantly reduces the size of the image.

  The frames after the Intra Frame in each group are compressed in another way using the method of P or B frames. P frame carries only the information about changes in macroblocks between the previous Intra Frame or P frame and itself. B frame carries the information about changes in macroblocks between the previous Intra Frame or P frame or between the following Intra Frame or P frame. Therefore the codec can pick the frame from which it can create better and sharper image preserving the used space.

- **M-JPEG**
  The main principle of the M-JPEG or Motion JPEG is that each of the frames is compressed separately using the JPEG compression format. Quality of the compression with M-JPEG is worse compared to MPEG but an advantage might be, that each of the frames is compressed separately so the missing or lost frame in the video does not cause significant problems.

Since LabVIEW does not support H.264 compression and the output of the detecting algorithm is a set of individual images, the M-JPEG compression format is used. Figure 4.7 shows the dependency of the network traffic on the amount of the JPEG compression with one client connect to the video server. The original image has resolution 1280 $\times$ 720 pixels

and the resulting image's quality can be set within the range from 0 to 1000, where 1000 means no compression and 0 means an absolute compression. Images from the video server are sent via Wi-Fi to the video client.

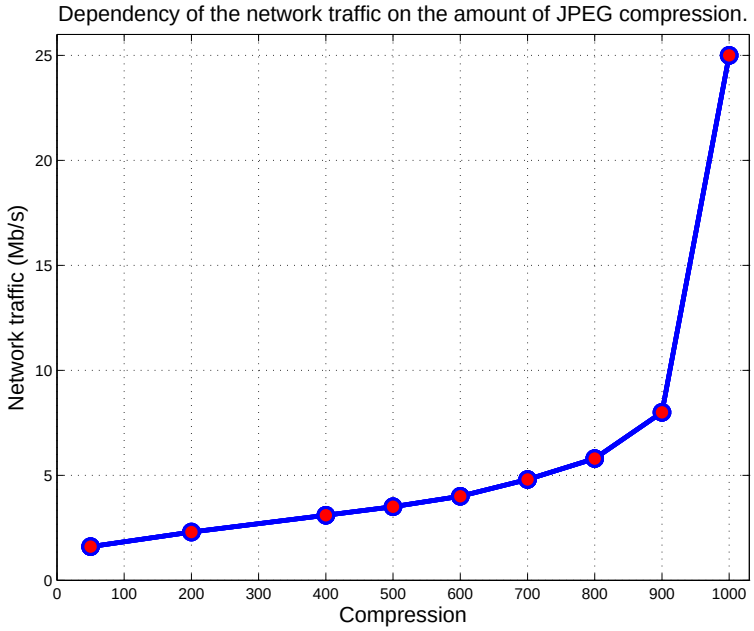


Figure 4.7: Dependency of the network traffic on the amount of the JPEG compression with one client connected to the video server.

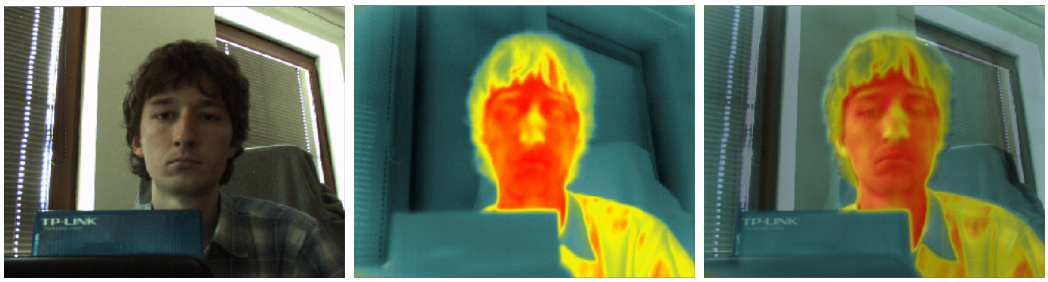
As can be seen, the amount of the network traffic is highly dependent on the JPEG compression. The reasonable amount of compression seems to be between 400 and 800. With this settings, there is no measurable delay or FPS drop in the received video stream as well as the resulting quality of the received video. The artifacts caused by the JPEG compression start to appear with the amount of compression set below the 300.

Advantage of the separate client application is that it can be customized to the user's demands and can have more functions than only showing a video stream. However, it could be useful if the client application was a kind of a freeware application so that the user would not have to be bothered with the LabVIEW license keys which are needed for applications running on the LabVIEW core. For this purpose, the VLC player can be used as a RTSP server or client [18]. Video input to the VLC player has to be in MPEG of H.264 format so the processed images have to be compressed appropriately. This could be done by creating a *.dll* library called in the LabVIEW application, which will be implemented in the future work.

4.3.5 Images fusion

The images from the visible spectrum camera can be displayed on the images captured by the thermocamera so the both scenes can be combined into the one. For lesser demands on computing power, the images are resized to the 320×254 px resolution. Since the images

have not the same resolution, the second one has to be resized in a way, that the object in the captured scene has the same size. Then, the offset of the second image has to be set, so that scenes in both of the images fit to each other.



(a) RGB image (b) Thermal image (c) Mixed image

Figure 4.8: The same scene captured by the RGB camera, thermo camera and fusion of both of the images. The thermal image is recalculated with the sepia palette.

For each pixel in the captured image, its color value is divided into its three components, i.e. Red R, Green G and Blue B. Then the fusion algorithm takes values of the same pixel from both of the images and linearly interpolates their R, G and B components. Linear interpolation is weighted by the *transparency* parameter, which can be set from 0 to 100. Value 0 means that only the value of the first image's pixel is taken and value 100 means that only the value of the second image's pixel is taken. Any value set in between these limits determines the weight of the first or second image's pixel value. The values of the R, G and B components are combined back together to the resulting mixed image.

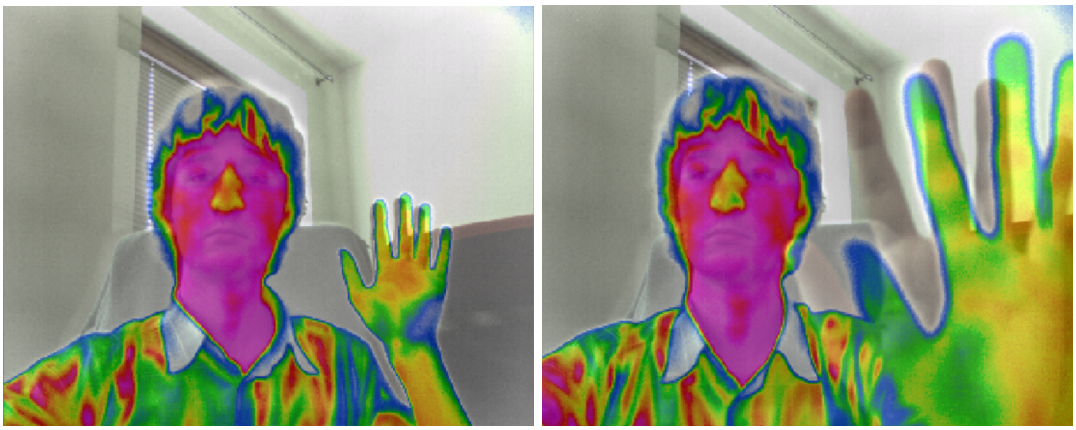
Figure 4.8 shows the same scene captured by the RGB camera and thermocamera with the *sepia* palette. Figure 4.8a shows the image from RGB camera, 4.8b from the thermocamera and 4.8c shows the mixed image with *transparency* parameter set to 60, i.e. 60% from the thermal image and 40% from the visible spectrum image. Figure 4.9 shows the same scene captured by the RGB camera and thermocamera with the *WBRGB* palette. Figure 4.9a shows the image from RGB camera, 4.9b from the thermocamera and 4.9c shows the mixed image with *transparency* parameter set to 60.

(a) RGB image (b) Thermo image (c) Mixed image

Figure 4.9: The same scene captured by the RGB camera, thermo camera and fusion of both of the images. The thermal image is recalculated with the WBRGB palette.

The advantage of the images fusion is that the resulting image contains information from both of the visible and thermal images. For example, as can be seen in the Figures 4.8 and 4.9, the writings or signs are visible only on the visible spectrum image, not in the thermal image. But with the correctly set the *transparency* parameter, the mixed image shows both the signs and the temperature information from the thermocamera.

However, the fusion is highly dependent on the distance between the cameras and the object. The resolution and offset of the one image has to be set accordingly to the other image and is dependent on the distance between the the captured object and the cameras. As can be seen in the Figure 4.10, the fusion can be successfully used only for one distance between the object and the cameras.



(a) Hand within the plane (b) Hand in front of the plane

Figure 4.10: Influence of the distance between the object and focused plane.

The resolution and offset of the image shown in the Figure 4.10a is set in a way that the images from the RGB camera and from the thermocamera fit to each other in the distance of the head from the cameras. It can be seen that the resulting fusion image is relatively clear and there are no visible spots, where the two images do not fit to each other. However, if the object changes its position towards to the cameras with the same resolution and offset, the images do not fit to each other at all, as can be seen in the Figure 4.10b. The fusion image of the head is still clear but there can be seen relatively big offset between the images of the hand placed in front of the cameras.

The fusion of the images could be used for scenes which have all of the objects in the same plane and distance between these objects and the cameras is higher than several meters. Unfortunately in this application, the scene is set in a way that the objects are moving towards to the cameras and their distance is varying from several meters to approximately a half a meter, therefore there is no reference point for which the correct resolution and offset can be set.

To fit the images to each other correctly, the x-offset and resolution have to be recalculated according to the distance of an object from the cameras. The relation between the distance x_d and the x-offset x_o can be computed by the equation

$$x_o = b \cdot \ln(a \cdot x_d),$$

where $a = 5,923$ and $b = 7,976$.

Another problem is that there is a certain delay between the RGB camera and the thermocamera so when the objects are moving fast, the visible spectrum image and the thermal image do not fit to each other. Figure 4.11 shows an example of the LabVIEW code used for the images fusion.

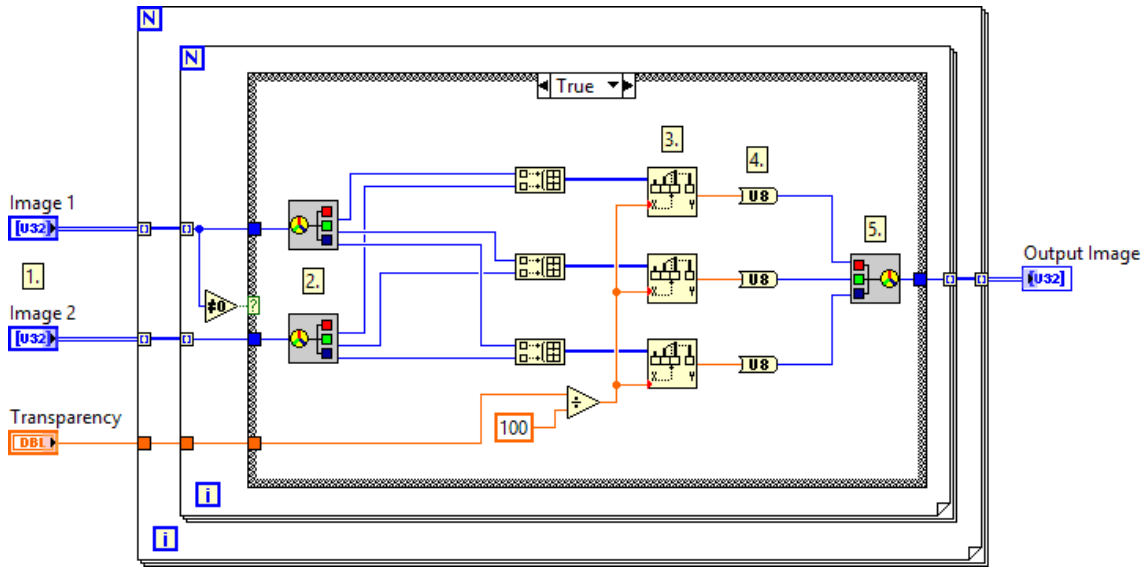


Figure 4.11: An example of the LabVIEW code used for the images fusion.

1. As inputs, there are two 2-Dimensional arrays of the visible spectrum image and the thermal image.

2. In two *for-loops*, the individual pixel values are divided into its three color components, i.e. Red, Green and Blue.

3. For each color component, the linear interpolation weighted by the transparency parameter is calculated.

4. Values representing each new colors are cast into the $U8$ format.

5. All of the interpolated color components are combined back together to the resulting 2-Dimensional array representing the combined image.

Chapter 5

Face detection algorithms

This chapter describes two algorithms for face detection which were used and implemented. The first one detects faces in visible spectrum images using OpenCV. The second one detects faces in thermogram and is based on the pattern detection.

5.1 OpenCV in RGB camera

OpenCV [19] (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. Among other things, it provides functions for face and object detection. Face detection algorithm, which is used in this application, is based on haar-like features presented by Jones and Viola in [20]. First we need to train the classifier with large number of positive and negative images. Positive images are those which contain only the object for detection. Negative images are images without this object. Then the haar-like features are extracted from these pictures. An example of two haar-like features can be seen in the Figure 5.1.

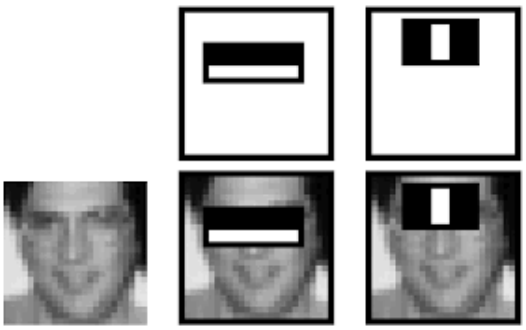


Figure 5.1: An example of haar-like features marking the eyes regions [20].

Each haar-like feature is a number represented by subtraction of pixel values in the black region from the white region. Finding a feature in an image can be represented by a weak classifier. Weak classifier is a classifier whose decision is just slightly better than random guessing [21]. When using large number of these weak classifiers, we can build a strong one,

for example with the Adaboost algorithm [22]. Viola and Jones in [20] have used cascade classifiers and significantly decreased the detection time. Features found before are divided into several stages and the detection runs as can be seen in the Figure 5.2.

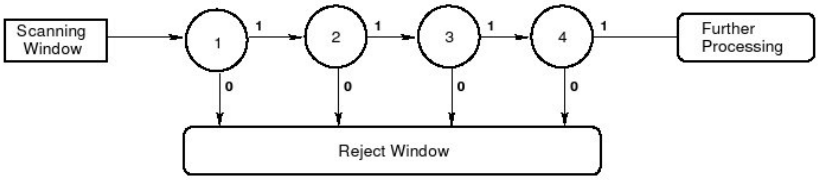


Figure 5.2: Diagram illustrating a cascade of classifiers [23].

Each circle represents one stage. If the algorithm decides that the window is not the detected object, the window is rejected and not further processed. This can save a lot of time because usually the biggest part of the image is filled with the background and only a small part is the detected object. If the window runs through all of the stages, we can say that it is the detected object. This algorithm has several parameters which are very important for its correct behavior.

- **Minimum neighbors** - minimum number of overlapping rectangles to mark a rectangle as face.
- **Window size** - minimal size of the object for detection.
- **Scale factor** - ratio between the old and new detector's window size.

First, the algorithm takes the image as the first window for detection and finds features in it. Then this window is reduced by the *scale* factor and this whole process continues until the window's size is the same as the minimal size of the object for detection. The current window shifts pixel by pixel through the whole image, as shown in the Figure 5.3.

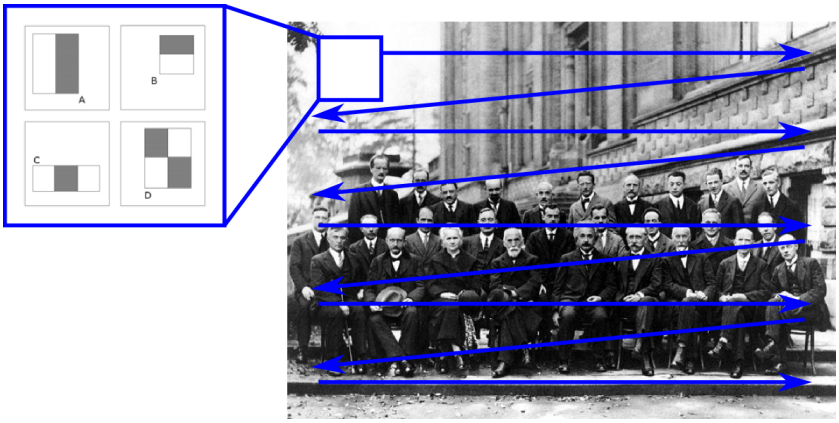


Figure 5.3: An example of the sliding window in the OpenCV face detection algorithm [24].

Correct settings of the parameters is crucial not only for the quality of detection but also for the time needed to detect the objects. The most important parameter to set with respect to the detection time is the *scale* factor. The scale factor determines the resulting number of windows which are detected in each of the images so it has to be set carefully or the detection time can raise up very high. Figure 5.4 shows detection time in the visible spectrum image with 1280 × 1024 px resolution when varying *scale* factor parameter and *window size* parameter set to 50.

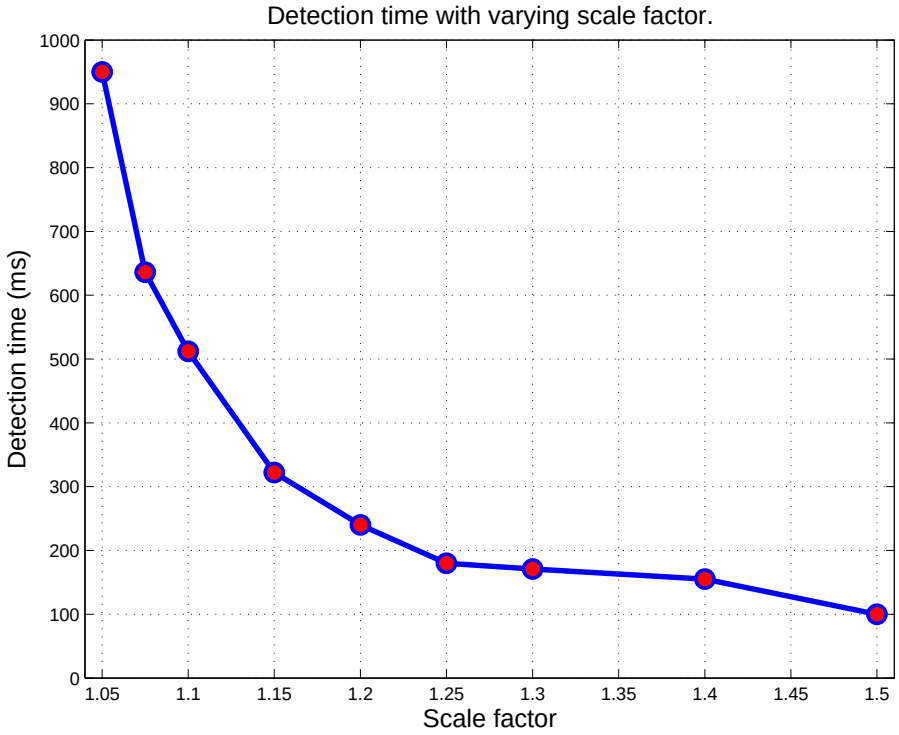


Figure 5.4: Detection time in the visible spectrum image with 1280 × 1024 px resolution varying *scale* factor and *window size* parameter set to 50.

As can be seen, the detection time is highly dependent on the *scale* factor. If the *scale* factor is set near 1, the detection time raises almost exponentially. Table 5.1 shows the concrete measured values of the detection time in the visible spectrum image with 1280 × 1024 px resolution when varying *scale* factor parameter and *window size* parameter set to 50.

Reasonable settings of the *scale* parameter with respect to the detection time is around 1,25. Setting the parameter higher does not significantly reduce the detection time but only causes that objects of certain sizes can not be detected because the detecting window is either too small or to big for detecting the object correctly. This is not crucial when all of the detected objects have about the same size and are in the correct distance from the camera. But if the objects are moving, there can appear spots or places in the scene where the objects can not be detected. Another parameter which affects the detection time is the *window size* parameter, i.e. the smallest size of the object we want to detect.

Scale	**Time** (ms)
1,05	950
1,08	636
1,10	512
1,15	322
1,20	240
1,25	180
1,30	171
1,40	155
1,50	100

Table 5.1: Detection time in the RGB image with 1280 × 1024 px resolution varying *scale* factor and *window size* parameter set to 50.

Figure 5.5 shows detection time in the visible spectrum image with 1280 × 1024 px resolution when varying *window size* parameter from 30 to 100 and *scale* factor set to 1,1, 1,3 and 1,5 respectively.

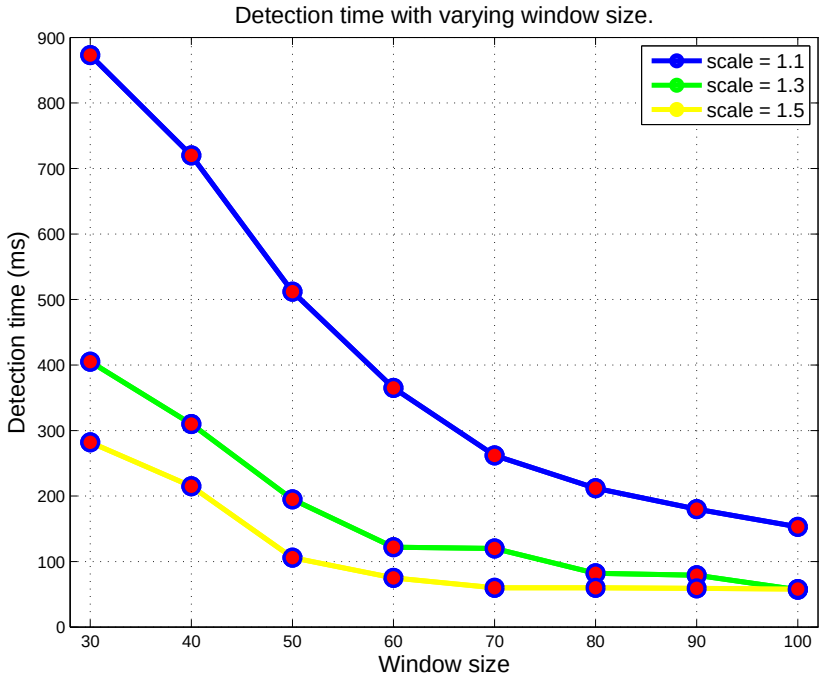


Figure 5.5: Detection time in the visible spectrum image with 1280 × 1024 px resolution varying *window size* parameter and *scale* factor set to 1,1, 1,3 and 1,5 respectively.

Table 5.2 shows the concrete measured values of the detection time in the visible spectrum image with 1280 × 1024 px resolution when varying *window size* parameter from 30 to 100 and *scale* factor set to 1,1, 1,3 and 1,5 respectively.

	Scale = 1,1	*Scale* = 1,3	*Scale* = 1,5
Window size	**Time** (ms)	**Time** (ms)	**Time** (ms)
30	873	405	282
40	720	310	215
50	512	195	106
60	365	122	75
70	262	120	60
80	212	82	60
90	180	79	59
100	153	57	58

Table 5.2: Detection time in the visible spectrum image with 1280 × 1024 px resolution varying *window size* parameter and *scale* factor set to 1,1, 1,3 or 1,5 respectively.

As can be seen, the detection time is highly dependent on the *window size* factor as well. It is hard to say, which settings of the *window size* parameter is the best because it depends on the objects we want to detect. If we are not interested in detecting small objects, the algorithm can be sped up by setting the *window size* parameter as high as possible, i.e. the smallest size of the object we want to detect.

We want to determine the temperature of the object so the detected rectangle in visible spectrum image has to be recalculated using homography to correspond to the image captured by the thermocamera.

5.2 Pattern detection

This algorithm is based on the pattern detection in the thermogram using Grayscale Value Pyramid method. Pattern for detection is an example of the object we want to detect. It is an image which should contain all of the key information about the object we want to detect needed for the correct recognition. This means that the objects we want to detect have to have some common key features that are contained in each of them.

The Grayscale Value Pyramid algorithm consists of two main phases, i.e. learning and matching the pattern for detection. This algorithm is based on the greyscale value method which treats an image as a greyscale picture consisting of 256 tones. Greyscale processing uses all of the information kept in an image so it is more accurate than extracting only a binary (black/white) information as in another similar detecting algorithms.

The learning phase extracts pixel intensities in a grey scale of the pattern for detection, as can be seen in the figure 5.6, and saves them as a part of the template.

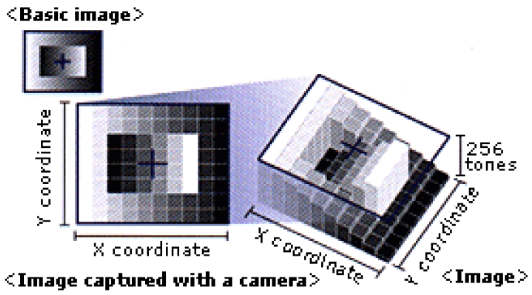


Figure 5.6: An example of greyscale values extraction used in pattern learning [25].

In a matching phase, the algorithm searches for places with the biggest cross-correlation between the image and the learned template. Normalized cross-correlation is commonly used method for pattern searching in an image. Generally, it measures a similarity between two signals when they are shifted against each other. For discrete functions, the cross-correlation [26] is in the form

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^{\star}[m]\, g[m+n], \tag{5.1}$$

where $f^{\star}$ is the complex conjugate and n represents the lag between the two signals.

When processing an image, it should be normalized first, due to the varying lighting and exposure conditions. We can compute the normalized cross-correlation γ of the template t and part of the image f in each possible point (u, v) in the whole image [27] using the equation

$$\gamma = \frac{1}{n} \sum_{x,y} \frac{(f(x,y) - \bar{f})(t(x,y) - \bar{t})}{\sigma_f \sigma_t}, \tag{5.2}$$

where n is the number of pixels in both of the image and template, $\bar{f}$ or $\bar{t}$ is the average of f or t respectively and σ_f or σ_t is the standard deviation of f or t respectively. The desired position of the template in the image is equivalent to the position with the maximal value γ_{max} of $\gamma(u, v)$.

Since this method is based on large series of multiplication operations, it can have high demands on computational time. To fasten up the necessary calculations, we can use the Pyramidal matching method [28].

The main idea of the Pyramidal matching method is to reduce both of the image and the template in the same ratio several times, always to the one-fourth of the previous size and first, search the most reduced pattern in the most reduced image. An example of the pattern and its reduction can be seen in the Figure 5.7.

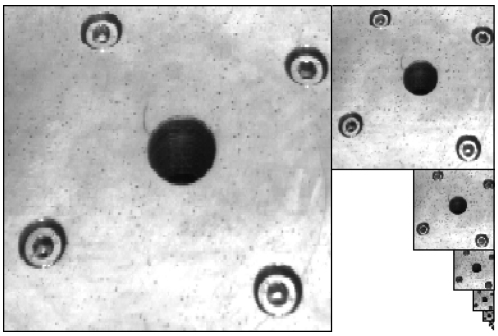


Figure 5.7: An example of a pattern and its reduction to lower resolutions [29].

The reduced images and templates are stored in a form of the Gaussian pyramid. Each layer of the pyramid represents one size of the image and template. The lower the layer, the larger the image and template resolution. At first, the pattern contained in the template is searched in the highest level of the pyramid, i.e. the one with the lowest resolution. An example of the Gaussian pyramid can be seen in the Figure 5.8.

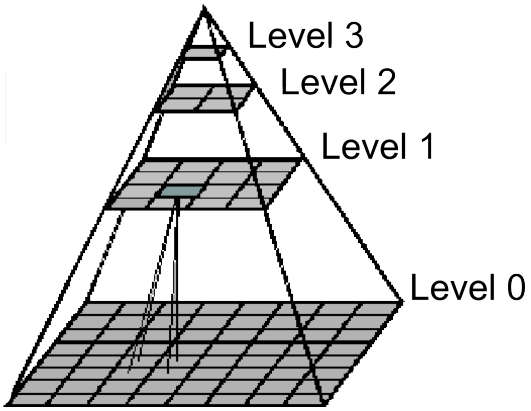


Figure 5.8: An example of the Gaussian Pyramid [30].

Pattern searching in the low resolution is not so computationally demanding as in the high resolution, therefore we can easily search the reduced image for the possible match areas. These areas are then moved to the next lower layer in the pyramid with the higher resolution. Finally, only the small part of the image in the original resolution has to be searched, so the computational time can be significantly reduced, if the number of pyramidal layers is set properly.

The advantage of the Greyscale value method is that no information is lost during the

matching. This method is suitable when the template contains intricate textures without well structured information. Disadvantage might be that successful matching is dependent on the changes in the surrounding lighting, due to the extraction of the greyscale pixel values. However, in the case of thermal images, the surrounding lighting does not have any influence on the objects in the captured image.

5.2.1 Face detection

The testing template used for learning phase of the Grayscale Value Pyramid algorithm is slightly modified head, with the background removed, as can be seen in the Figure 5.9. The goal is to find a template, which has all of the key features as most of the human faces do. Another possibility is that the user can mark and crop any part of the scene, which can be set as a template for detection.

Figure 5.9: An example of the default testing pattern for face detection.

Grayscale Value Pyramid method used for matching the template can be applied only to the one size of the desired object for detection. If the faces in the scene vary in their sizes, it is necessary to create multiple scaled templates for detection and detect them using the algorithm separately. The default template is sent to the learning phase and stored in the array of templates. After that, its size is reduced until the following condition is met

$$ws \geq \frac{imgSize}{scale^{\,i}}, \tag{5.3}$$

where ws represents the desired minimal size of the detected face, $imgSize$ is the current image's smaller dimension, $scale$ is the factor by which the template is reduced and i is the current loop iteration. Each of these templates is sent to the learning phase of the detecting algorithm and stored in the array of templates.

The speed of the algorithm is dependent on the *scale* factor and on the desired minimal size of the detected face or, in another words, on the number of created templates for detection, similarly as with the OpenCV face detection algorithm. For speeding up the algorithm, the array of templates is divided into four equally large parts. Each of these arrays is sent to the matching part of the algorithm, which runs separately in the four independent parallel loops or threads. The output of the matching phase is an array of detected objects, which consists of the coordinates of the overlaying rectangle, coordinates of the center point and the matching score. The matching score is a number in a scale

from 0 to 1000 and its value depends on the quality of the match, where 1000 represents an absolute match.

Since some of the templates are very similar in size, it can happen that the template is detected several times, only in different sizes. These multiple detections have to be removed otherwise the output image is not clear. If the two matches have their center points less than 40 pixels from each other, the one with the lower score is deleted. Finally, the output of the matching phase is an array of unique detections in the current image.

Similarly as with the OpenCV face detection algorithm, the correct settings of the *scale* and *window size* parameters is very important for the quality of the detection and the detection time. In case of the Grayscale Value Pyramid algorithm, the *scale* factor determines the resulting number of windows which are detected in each of the images as well. Figure 5.10 shows detection time in the thermal image with 640 × 512 px resolution when varying *scale* factor and *window size* parameter set to 60, 80 and 100 respectively.

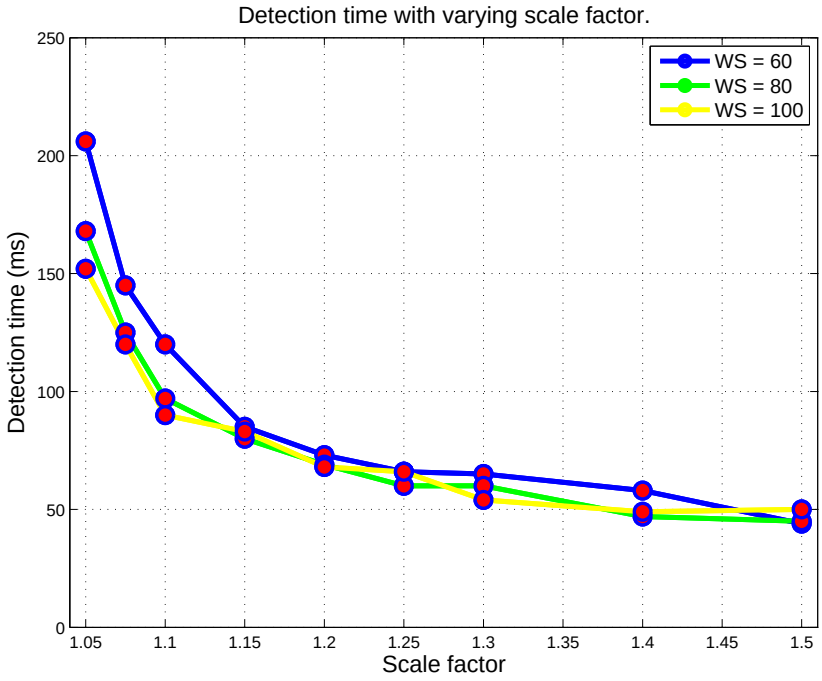


Figure 5.10: Detection time in the thermal image with 640 × 512 resolution varying *scale* factor and *window size* parameter set to 60, 80 or 100 respectively.

As can be seen, the detection time is highly dependent on the *scale* factor. If the *scale* factor is set near 1, the detection time raises almost exponentially. Table 5.3 shows the concrete values of the detection time in the thermal image with 640 × 512 px resolution when varying *scale* factor and *window size* parameter set to 60, 80 and 100 respectively. When the *scale* factor is set below the 1,15 value, the detection time for different values of *window size* parameter differs. It can be seen that the detection time is highest with the *window size* parameter set to 60 and lowest with the *window size* parameter set to 100. This is not surprising because the smaller the window size, the greater the amount of data to process. When the *scale* factor is set higher than the 1,15 value, the detection time remains

about the same for all of the values of the *window size* parameter.

	$WS = 60$	$WS = 80$	$WS = 100$
Scale	**Time** (ms)	**Time** (ms)	**Time** (ms)
1,05	206	168	152
1,08	145	125	120
1,10	120	97	90
1,15	85	80	83
1,20	73	69	68
1,25	66	60	66
1,30	65	60	54
1,40	58	47	49
1,50	44	45	50

Table 5.3: Detection time in the thermal image with 640 × 512 resolution varying *scale* factor and *window size* parameter *WS* set to 60, 80 or 100 respectively.

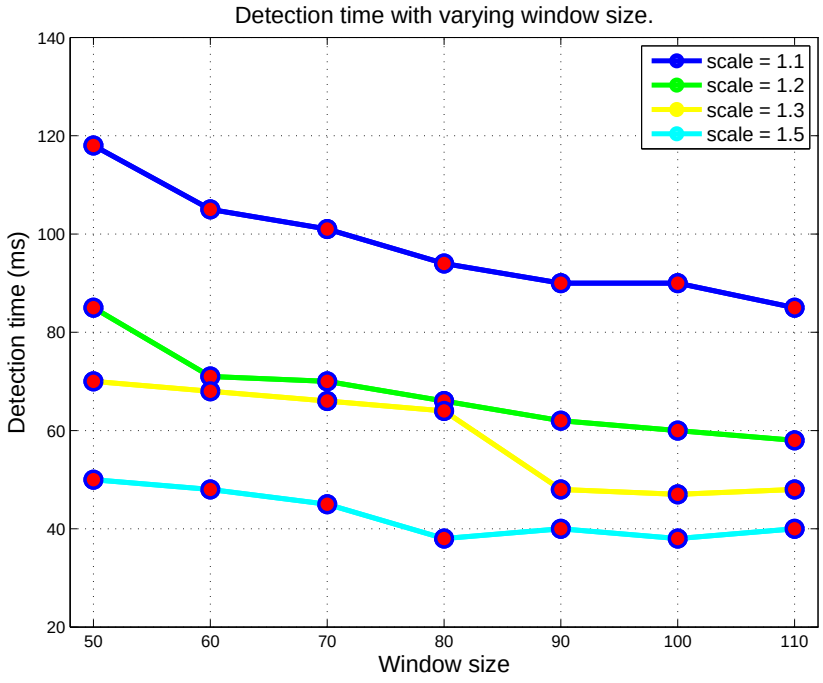


Figure 5.11: Detection time in the thermal image with 640 × 512 resolution varying *window size* parameter and *scale* factor set to 1,1, 1,2, 1,3, or 1,5 respectively

The functionality of the *window size* parameter is the same as in OpenCV face detection algorithm, i.e. settings of the smallest size of the object we want to detect. Figure 5.11 shows detection time in the thermal image with 640 × 512 resolution varying *window size* parameter and *scale* factor set to 1,1, 1,2, 1,3, or 1,5 respectively.

As can be seen, the detection time is dependent on the *window size* parameter as well. Table 5.4 shows the concrete values of the detection time in the thermal image with 640 × 512 resolution varying *window size* parameter parameter and *scale* factor set to 1,1, 1,2, 1,3, or 1,5 respectively. It can not be said which settings of the *window size* parameter is the best because it depends on the objects we want to detect. If the *scale* factor is set below the 1,2 value, the detection time rises when lowering the *window size* parameter. If the *scale* factor is set higher than the 1,2 value, the detection time remains almost the same for all of the values of the *window size* parameter. If we are not interested in detecting small objects, the algorithm can be sped up by setting the *window size* parameter as high as possible, i.e. the smallest size of the object we want to detect. But in case of the Grayscale Value Pyramid algorithm, the speed-up is not so significant as in the case of the OpenCV algorithm.

	Scale = 1,1	*Scale* = 1,2	*Scale* = 1,3	*Scale* = 1,5
Window Size	**Time** (ms)	**Time** (ms)	**Time** (ms)	**Time** (ms)
50	118	85	70	50
60	105	71	68	48
70	101	70	66	45
80	94	66	64	38
90	90	62	48	40
100	90	60	47	38
110	85	58	48	40

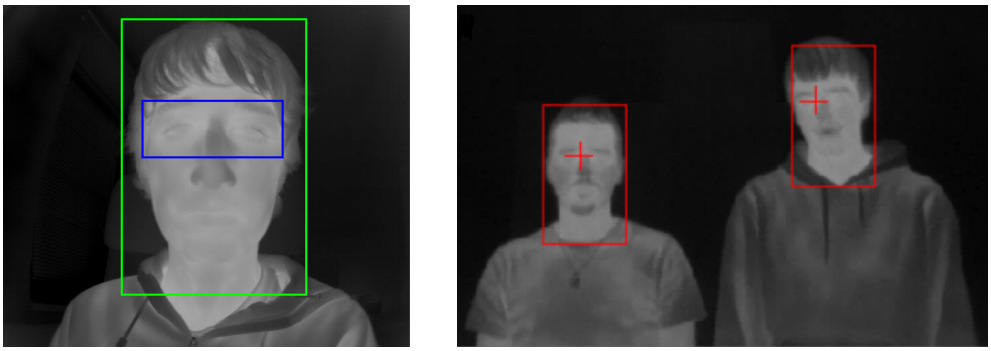
Table 5.4: Detection time in the thermal image with 640 × 512 resolution varying *window size* parameter and *scale* factor set to 1,1, 1,2, 1,3, or 1,5 respectively.

5.2.2 Eyes detection

The motivation for detecting eyes within the detected faces is that the most stable temperature in the humans face is in the corner of the eyes [3][31]. Corners of the eyes are spots where the face temperature is closest to the temperature of the body core, so measuring the temperature in these spots is the most accurate and reliable way how to determine the humans temperature from its face. Eyes are detected in each of the previously detected faces using the Grayscale Value Pyramid algorithm, only with different pattern for detection.

However, the eyes detection highly depends on the face position and rotation and is not reliable. Example of the eyes detection can be seen in the Figure 5.12a. Another disadvantage of the eyes detection are its computational demands which are rising with the amount

of detected faces. For these reasons, the eyes detection was excluded from the detection process and therefore is not used. On the other hand, the maximal temperature in the humans face is in the corner of the eyes anyway, as can be seen in the Figure 5.12b, so there is no particular need for the eyes detection.



(a) Detected eyes (b) Maximal temperature within the faces

Figure 5.12: An example of detected eyes along with the red cross marked maximal temperature within the faces.

5.3 Homography

When there are two cameras looking at the same scene and we want to know coordinates of a point in the first image x in the second one x', we can use the homography [32]. The goal is to find the matrix H which complies with the equation

$$\lambda x' = Hx, \tag{5.4}$$

where $\lambda \in \mathbf{R}$.

Since H is 3×3 matrix, we need to introduce homogeneous coordinates. The 2D points x or x' are in the form

$$x = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad x' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}. \tag{5.5}$$

Now from equations (5.4) and (5.5) we can write

$$\lambda u' = h_1^T x \tag{5.6}$$
$$\lambda v' = h_2^T x \tag{5.7}$$
$$\lambda = h_3^T x. \tag{5.8}$$

If we substitute (5.8) into (5.6) and (5.7), after some modifications we get

$$\begin{bmatrix} u & v & 1 & 0 & 0 & 0 & -u'u & -u'v & -u' \\ 0 & 0 & 0 & u & v & 1 & -v'u & -v'v & -v' \end{bmatrix} h = 0 \tag{5.9}$$

$$Mh = 0. \tag{5.10}$$

Each pair of corresponding two points in both of the images brings two new rows into the matrix M. We are looking for one-dimensional subspaces of 3×3 matrices of rank 3, so the matrix M has to have rank at least 8. This means that we need four corresponding points, in which no three of the four lie on the same line. Since

$$h = \begin{bmatrix} h_1^T & h_2^T & h_3^T \end{bmatrix}^T,$$

the matrix of homography H can be computed as the null space of the matrix M.

5.4 Imaging detections

All of the objects detected by the Grayscale value pyramid algorithm can be displayed to the user in two ways, namely in the thermal image and in the visible image. The array of detections, which is an output of the matching phase of the detection algorithm, includes coordinates of the overlaying boxes around each of the detections. These boxes are displayed around the detected faces in the thermal image. The color of the bounding box is determined by the maximal temperature of the detected object. If it is higher than previously set temperature threshold, the color is red, otherwise is it green, as can be seen in the Figure 5.13.

Figure 5.13: An example of detection of the two persons in thermal image. The left one has temperature lower than the limit, the temperature of the right one's is elevated.

When displaying the detections in the visible spectrum image, its coordinates have to be recalculated. Each bounding box around the detected object in the thermal image has four coordinates. Each of this coordinates has to be multiplied by the matrix of homography H, as described in the previous section. However, the coordinates are represented by x and y positions but the matrix of homography H has dimensions 3×3. For this purpose, we have to use homogeneous coordinates [33] as follows

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Coordinates in the visible spectrum image x' and y' can be computed by

$$H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \cdot \frac{1}{z'}$$

With these recalculated coordinates of the bounding boxes, we can visualize them in the visible spectrum image, as can be seen in the Figure 5.14.

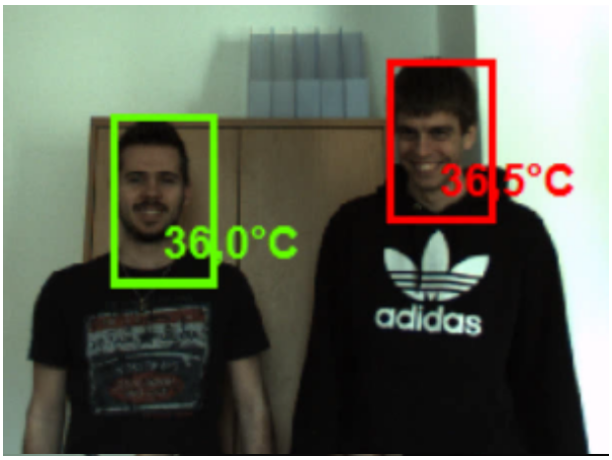


Figure 5.14: An example of detection of the two persons visualized in the visible spectrum image. The left one has temperature lower than the limit, which is set to the 36 °C, the temperature of the right one's is elevated.

Similarly, as with the thermal image, the color of the bounding box is dependent on the object's temperature. Along with the bounding box, the concrete value of the object's temperature is displayed in the visible spectrum image. To easily see all of the detected objects, the visible spectrum images are cropped accordingly to all of the recalculated bounding boxes, as can be seen in the Figure A.2 in Appendix A. These cropped detections are, for better clarity, sorted by the object's temperature in descending order.

Chapter 6

Detection evaluation

For evaluating the quality of face detection, we can use the Receiver Operating Characteristic (ROC) curve [34]. ROC curve shows a relationship between the specificity and sensitivity of the detector for all of the values of the chosen parameter. For correct understanding the ROC curves, four parameters have to be introduced. TP or true positives is a number of all correctly identified objects, FP or false positives is a number of incorrectly identified objects, TN or true negatives is a number of correctly rejected objects and FN or false negatives is a number of incorrectly rejected objects. With these parameters, the sensitivity or true positive rate TPR can be defined as

$$TPR = \frac{TP}{TP + FN}. \tag{6.1}$$

Specificity, or true negative rate SPC can be defined as

$$SPC = \frac{TN}{TN + FP}. \tag{6.2}$$

True positives, False positives and False negatives can be measured but True negatives is a number which is unknown. For this purpose, we have to use modified ROC curve with modified False positive rate instead of the specificity. Modified False positive rate FPR_m can be defined as

$$FPR_m = \frac{FP}{TP + FN}. \tag{6.3}$$

Limits of the y or True positive rate axis are from 0 to 1, where 0 means that there are no correctly identified objects and 1 means that all of the objects are identified correctly. Limits of the x or Modified False positive rate axis are from 0 to infinity, where 0 means that there are no incorrectly identified objects and there can be any number of incorrectly identified objects. The best settings of the tested parameter is the one which is closest to the left upper corner of the ROC curve, i.e. the one with the highest True positive rate and simultaneously with the lowest False positive rate as possible.

6.1 Testing application

For evaluation of the face detection, the testing application has to be implemented. This application serves for two main purposes which are marking the correct detections and detecting objects from video.

- **Marking detections**
  This part of the application loads the video file on which we want to test the detecting algorithm. In this video file, we have to manually mark all of the objects which should be detected by the detecting algorithm, frame by frame. Coordinates of these marked overlaying rectangles are saved for further processing.

- **Detecting from video**
  This part of the application detects objects from the loaded video file. Similarly, as in the Main application, each image of the video stream is recalculated with the desired palette. Then, the desired pattern is detected in the image using the Grayscale Value Pyramid algorithm, exactly as in the Main application. All of the detections are visualized in a form of overlaying rectangles as well as the previously manually marked correct objects.

  The detected rectangle is considered as the True positive one in case that the euclidean distance between the centers of the detected and correctly marked overlaying rectangles is lower than 30 px and the ratio of the areas of these rectangles is lower than 1,5. If there is a marked rectangle which does not fit to any of the detected ones, it is considered as the False negative one. Finally, if there is a detected rectangle which does not fit to any of the previously marked ones, it is considered as the False positive one.

6.2 Detection influence

6.2.1 Parameters influence

The correct settings of the detection algorithm's parameters is very important not only for the detection time but mainly for the quality of the detection. The two parameters which affect the quality of the detection the most are the *scale* factor and the *score* parameter.

The *scale* factor determines how big can be the differences between different sizes of the same object for detection. Setting the *scale* factor higher causes that in the detected scene, there will be certain areas where the objects could not be detected. Setting the *scale* factor lower raises the computational time and can cause an increase of the False positive detections. The ROC curve introduced before can be helpful for finding the best settings of the *scale* factor. The other parameters are set to constant values while varying the *scale* factor. Concrete values of the parameters used for tuning-up the *scale* factor are $score = 700$ and $window\ size = 80$. We are finding the point on the ROC curve which has the highest value of the True positive rate and the lowest value of the False positive rate as possible, i.e. ideally the point in the left upper corner. For this point, we can find the value of the tested parameter which can be then considered as the best settings.

Figure 6.1 shows the ROC curve for the *score* parameter set to 700 and *window size* parameter set to 80 varying *scale* factor from 1,05 to 1,6. The scene on the source video for this ROC curve is set in way that it simulates persons passing through a hallway towards to the cameras. In this video, there is only one person passing by.

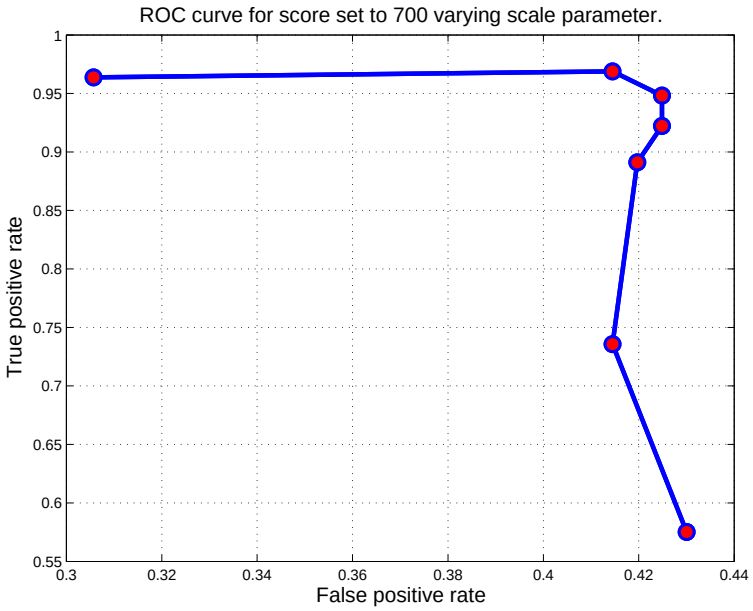


Figure 6.1: ROC curve for the *score* parameter set to 700 and *window size* parameter set to 80 varying *scale* factor from 1,05 to 1,6.

Table 6.1 shows the concrete values of detection counts for the *score* parameter set to 700 and *window size* parameter set to 80 varying *scale* factor from 1,05 to 1,6.

Scale	**TP**	**FP**	**FN**
1,05	186	59	7
1,10	187	80	6
1,20	183	82	10
1,30	178	82	15
1,40	172	73	21
1,50	142	80	51
1,60	111	83	82

Table 6.1: Detection counts for the *score* parameter set to 700 and *window size* parameter set to 80 varying *scale* factor from 1,05 to 1,6.

As can be seen, the left and topmost point on the ROC curve is the one with the *scale* factor set to *scale* = 1,05. With this settings, the algorithm detected most of the objects

correctly with the smallest number of false positive detections. When the *scale* factor is set to *scale* = 1,1, the amount of correctly detected objects is almost the same but the amount of false positive detections is higher. Setting the *scale* factor higher than *scale* = 1,1 does not bring any new false positive detections but also decreases the amount of correctly detected objects. The best settings for the *scale* factor seems to be from 1,05 to 1,2.

Another video for tuning-up the *scale* factor is set in way that it also simulates persons passing through a hallway towards to the cameras but with multiple persons passing by. Figure 6.2 shows the ROC curve for the *score* parameter set to 700 and *window size* parameter set to 60 varying *scale* factor from 1,05 to 1,5 with multiple persons in the scene.

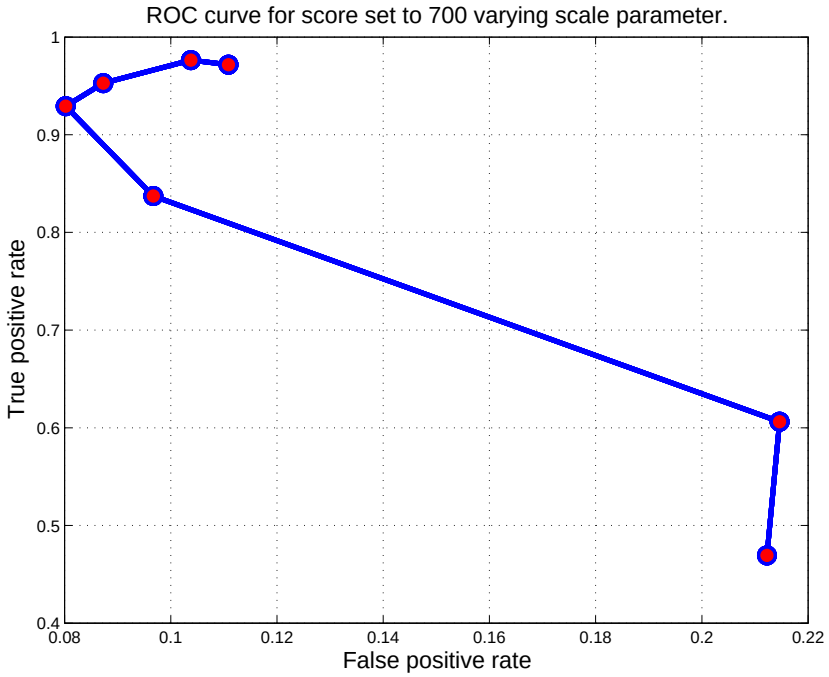


Figure 6.2: ROC curve for the *score* set to 700 and *window size* set to 60 varying *scale* factor from 1,05 to 1,5 with multiple persons in the scene.

Table 6.2 shows the concrete values of detection counts for the *score* set to 700 and *window size* set to 60 varying *scale* factor from 1,05 to 1,5 with multiple persons in the scene.

As can be seen, there is not one point which is both leftmost and topmost in this scene with multiple persons passing by. The topmost point on the ROC curve is the one with the *scale* factor set to *scale* = 1,1. If we are aiming for the highest amount of correctly detected objects, we should set the scale factor to *scale* = 1,1. The leftmost point on the ROC curve is the one with the *scale* factor set to *scale* = 1,25. If we are aiming for the lowest amount of false positive detections, we should set the *scale* factor to *scale* = 1,25. Setting the *scale* factor higher than *scale* = 1,25 significantly reduces the amount of correctly detected objects with increasing number of false positive detections. The best settings for the *scale* factor seems to be from 1,1 to 1,25.

Another parameter which can be tuned-up with the ROC curves is the *score* parameter which indicates the quality of the current detection, i.e. how much is the detected object

Scale	TP	FP	FN
1,05	412	47	12
1,10	414	44	10
1,20	404	37	20
1,25	394	34	30
1,30	355	41	69
1,40	257	91	167
1,50	199	90	225

Table 6.2: Detection counts for the *score* set to 700 and *window size* set to 60 varying *scale* factor from 1,05 to 1,5 with multiple persons in the scene.

similar to the pattern set for detection. The minimal value of the *score* parameter is $score = 0$ and the maximal one is $score = 1000$, which indicates an absolute match. Raising the score parameter's value decreases the number of false positive detections but also, from certain value, decreases the amount of correctly detected objects. Setting the score parameter to the low values can assure that all of the objects will be correctly detected but also significantly increases the number of false positive detections.

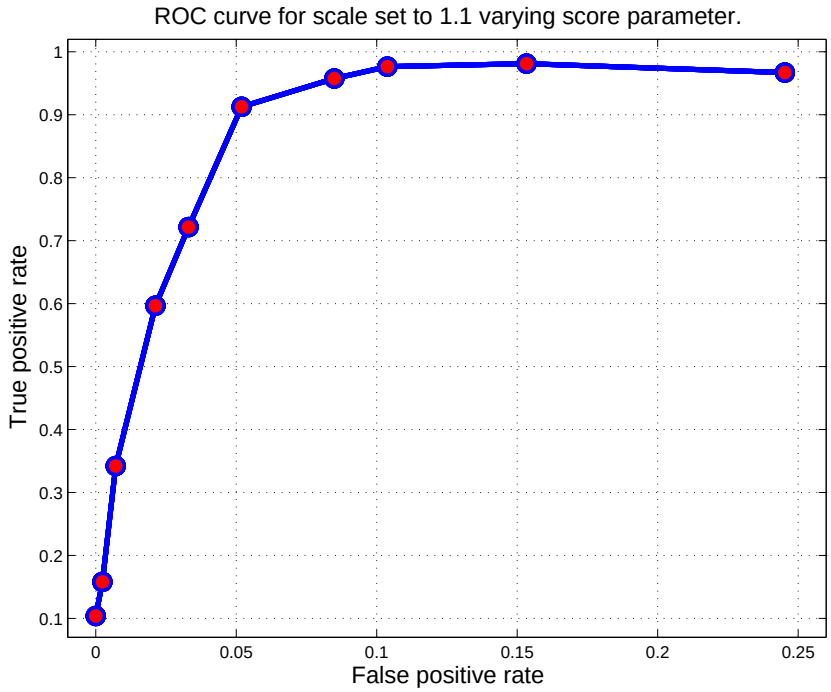


Figure 6.3: ROC curve for the *scale* factor set to 1,1 and *window size* set to 60 varying *score* parameter from 800 to 650 with multiple persons in the scene.

Figure 6.3 shows the ROC curve for the *scale* factor set to 1,1 and *window size* parameter set to 80 varying *score* parameter from 800 to 600. The video and scene are the same as with tuning-up the *scale* factor with multiple persons moving in the scene.

Table 6.3 shows concrete values of detection counts for the *scale* factor set to 1,1 and *window size* set to 80 varying *score* parameter from 800 to 600 with multiple persons moving in the scene.

Score	TP	FP	FN
800	44	0	380
780	67	1	357
760	145	3	279
740	253	9	171
730	306	14	118
720	387	22	37
710	406	36	18
700	414	44	10
680	416	65	8
650	410	104	14

Table 6.3: Detection counts for the *scale* factor set to 1,1 and *window size* set to 60 varying *score* parameter from 800 to 650 with multiple persons in the scene.

As can be seen, this ROC curve is similar to the ones used in medicine or machine learning. The leftmost point on the ROC curve is the one with the lowest amount of false positive detections. However, the amount of correctly detected objects is only around 10%. This is for the *score* parameter set to $score = 800$. If we do not want to detect any false positive detections, we could use this settings of the *score* parameter but with the cost of the very low True positive rate.

The rightmost points on the ROC curve are the ones with the highest amount of correctly detected objects. However, the amount of false positive detections is significantly higher. This is for the *score* parameter set to $score = 650$. The amount of correctly detected objects rises with lowering the *score* parameter as well as the amount of false positive detections. The highest True positive rate with relatively low amount of false positive detections seems to be for the *score* parameter set from 700 to 720. With these settings, the True positive rate is higher than 90% and modified False positive rate is from 5% to 10%. Again, the settings of the *score* parameter depends on whether we want to detect as much objects correctly as possible or reduce the amount of false positive detections.

Figure 6.4 shows the ROC curve for the *scale* factor set to 1,1 and *window size* parameter set to 60 varying *score* parameter from 800 to 650. The video and scene are the same as

with tuning-up the *scale* factor with one person moving in the scene.

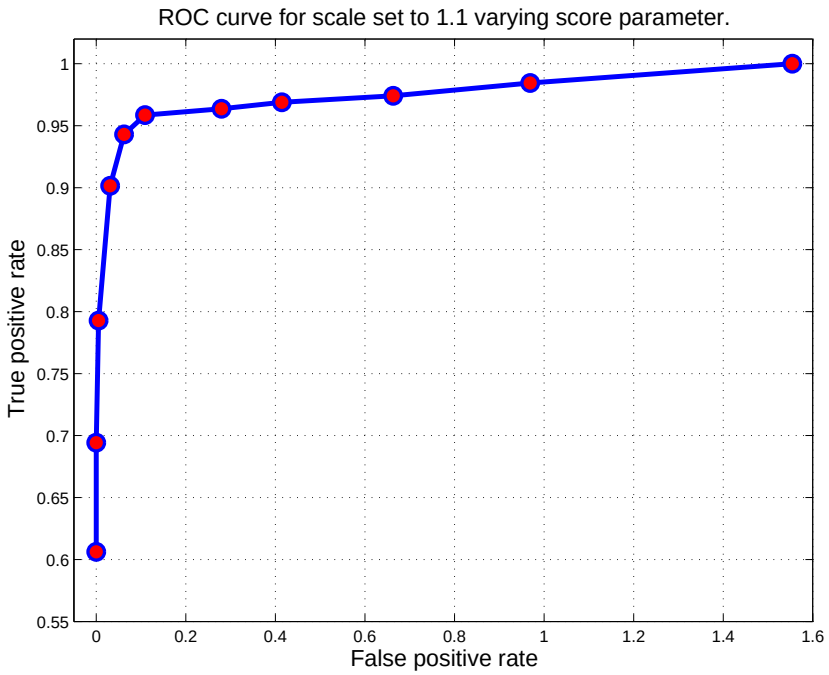


Figure 6.4: ROC curve for *scale* factor set to 1,1 and *window size* parameter set to 80 varying *score* parameter from 800 to 600 with one person in the scene.

Table 6.4 shows the concrete values of detection counts for *scale* factor set to 1,1 and *window size* parameter set to 60 varying *score* parameter from 800 to 650 with one person moving in the scene.

As can be seen, this ROC curve is very similar to the previous one. The leftmost point on the ROC curve is the one with the lowest amount of false positive detections. The amount of correctly detected objects is only around 60%. This is for the *score* parameter set to $score = 800$. The rightmost point on the ROC curve is the one with the highest amount of correctly detected objects. In this case, the True positive rate is 100%, i.e. all of the objects were correctly detected. However, the amount of false positive detections is more than 1,5 times higher than the true positive ones. This is for the *score* parameter set to $score = 600$.

The highest True positive rate with relatively low amount of false positive detections seems to be for the *score* parameter set from 700 to 730. With these settings, the True positive rate is higher than 95% and modified False positive rate moves from 10% to 30%.

6.2.2 Scene influence

Another thing which needs to be taken under consideration is the influence of the background and the scene itself. Background of the scene set up before for tuning-up the parameters was almost all over the place covered with a black color. Achieving the black color in the background can significantly improve quality of the face detection. Black background can be easily set by aiming the cameras towards the objects with uniformly distributed temperature

Score	TP	FP	FN
800	117	0	76
780	134	0	59
760	153	1	40
740	174	6	19
730	182	12	11
720	185	21	8
710	186	54	7
700	187	80	6
680	188	128	5
650	190	187	3
600	193	300	0

Table 6.4: Detection counts for *scale* set to 1,1 and *window size* parameter set to 80 varying *score* parameter from 800 to 600 with one person in the scene.

lower than 25 °C or higher than 45 °C, in our case to the wall. If the scene is not set up properly, there can appear constant False positive detection, as can be seen in the Figure 6.5.

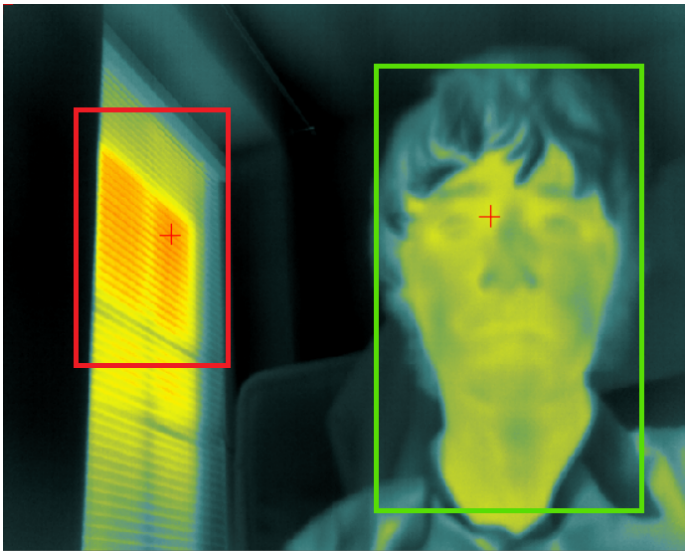


Figure 6.5: An example of a scene with the constant False positive detection.

In this scene, there is a constant False positive detection of the window which is heated by the sun to the temperature similar to the humans one. These constant False positive

detections can make the resulting image very unclear and it is better to set up the scene carefully to minimize their detection.

6.2.3 Pattern influence

Very important thing for the pattern detection is selection of the pattern itself. Since the Greyscale Value Pyramid algorithm extracts grey values from the image and correlates them with the pattern, the patterns should have some key significant features. The main of the features are its oval grey shape, darker nose in the middle, lighter eyes, darker eyebrows and a neck area under the oval. The best pattern complying with these conditions seems to be a slightly modified head. An example of several patterns tested for face detection can be seen in the Figure 6.6.

(a) Pattern 1 (b) Pattern 2 (c) Pattern 3 (d) Pattern 4 (e) Pattern 5

Figure 6.6: The set of testing patterns.

All of these patterns were tested in the same videos as with testing the influence of the parameters. The scene was a hallway with several persons passing by, towards the cameras. The parameters were to the values *scale* factor = 1,1, *window size* parameter = 60 and *score* parameter = 700. Figure 6.8 shows the graph of the True positive and False positive rate in the same video scene for different patterns. In this scene, there were both one and multiple persons passing by.

Table 6.5 shows the concrete values of the True positive and False positive rate in the same video scene for different patterns.

Pattern	**TP**	**FP**	**FN**
1	603	172	14
2	599	201	18
3	572	56	45
4	426	159	191
5	150	152	467

Table 6.5: Detection counts for different patterns with *scale* factor = 1,1, *window size* parameter = 60 and *score* parameter = 700.

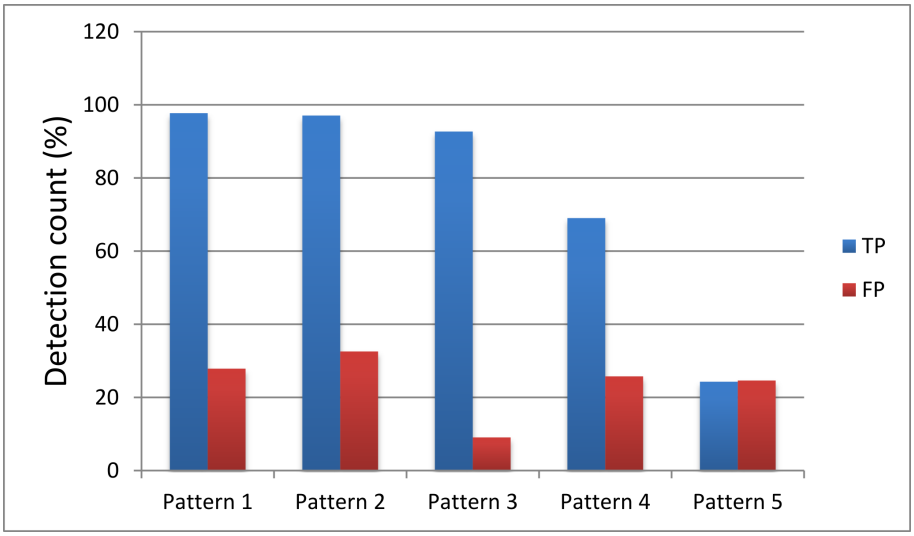


Figure 6.7: Graph of the True positive and False positive rate in the same video scene for different patterns with *scale* factor = 1,1, *window size* parameter = 60 and *score* parameter = 700.

As can be seen, the highest amount of correctly detected objects appeared when using the pattern 6.6a for face detection. Similar amount of correctly detected objects but with higher amount of False positive detections was achieved with the pattern 6.6b. From these first two patterns, there can be seen the importance of the key features contained in the pattern. The pattern 6.6b is the same as the pattern 6.6a but with the blur applied. Therefore some of the key features are lost and the algorithm detects some other objects with similar shape but different structure as False positive detections.

The poor results of the pattern 6.6d are caused by its shape. It has a structure similar to the best pattern 6.6a but it does not contain the neck area. The neck area distinguishes many similar oval shaped objects from the real faces. The last pattern 6.6e was introduced only to illustrate that even a grey oval with no structure within can detect several faces. However, its accuracy is only a little above 20% with about the same amount of False positive rate.

6.2.4 Palettes influence

Another thing which can affect the face detection is selection of the color palettes. Each of the palettes can point out different areas in the image so it is worth testing their influence on the face detection. Figure 6.8 shows the graph of the True positive and False positive rate in the same video scene for different palettes with parameters set to *scale* factor = 1,1, *window size* parameter = 60 and *score* parameter = 700.

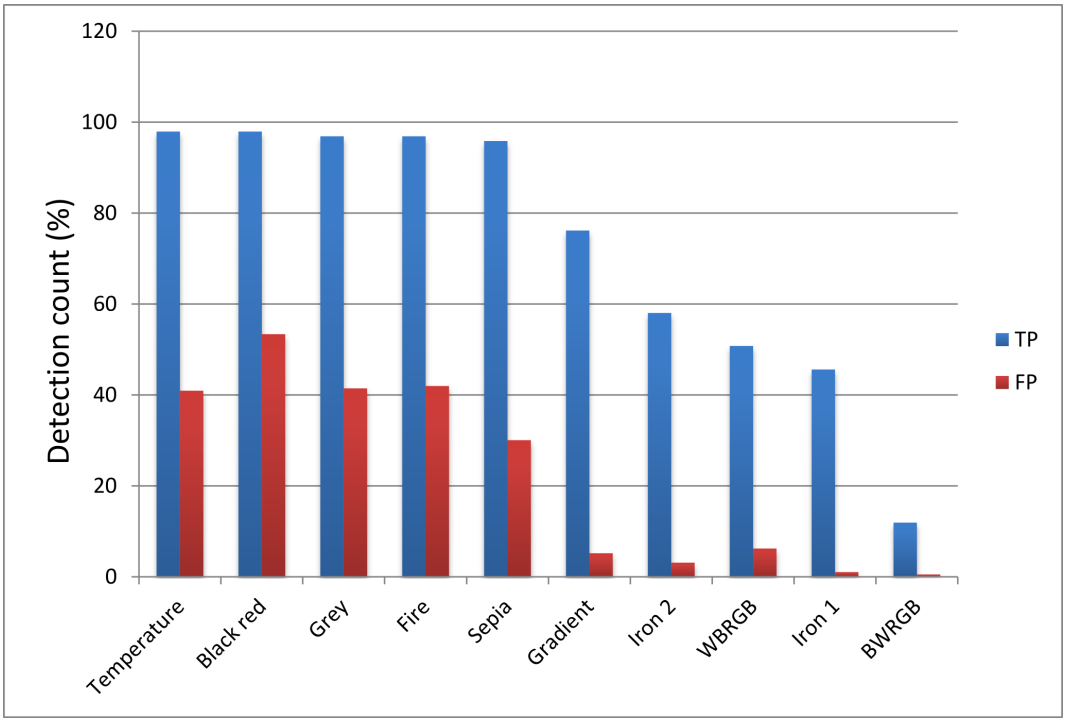


Figure 6.8: Graph of the True positive and False positive rate in the same video scene for different palettes with *scale* factor = 1,1, *window size* parameter = 60 and *score* parameter = 700.

Table 6.6 shows the concrete values of the True positive and False positive rate in the same video scene for different palettes.

(a) Temperature (b) Black Red (c) Grey (d) Fire (e) Sepia

Figure 6.9: The set of five best testing palettes.

The first five palettes with the best results can be seen in the Figure 6.9, the rest of them in the Figure A.1 in the Appendix A. For testing the palettes influence, each of the frames in the video was recalculated with the appropriate palette as well as the pattern for detection. Both of these images, the frame and the pattern, were then recalculated back to the greyscale image and further processed as in the previous cases.

Regarding the amount of correctly detected objects, the first five palettes can be considered as equal with the True positive rate higher than 96%. The next four palettes have lower amount of False positive detections but also their True positive rate is only from 50% to 70%.

Palette	**TP**	**FP**	**FN**
Temperature	189	79	4
Black red	189	103	4
Grey	187	80	6
Fire	187	81	6
Sepia	185	58	8
Gradient	147	10	46
Iron 2	112	6	81
WBRGB	98	12	95
Iron 1	88	2	105
BWRGB	23	1	170

Table 6.6: Detection counts for different palettes with *scale* factor = 1,1, *window size* parameter = 60 and *score* parameter = 700.

The reason why the first five palettes came out of the testing as the best ones is that they are all very similar to each other and they do not have large changes in their contrast. Also, their background has a black color which is important for the correct detection. We can see, that setting the palette incorrectly can cause very low True positives rate. On the other hand, we can not say that there is only one palette which would be the best for the face detection, so the basic Grey palette can be used.

6.3 Results

It is important to say, that all of the measured and computed values of True positives, False positives and False negatives must be treated with caution. Since this evaluation is based on comparing the detected objects to the manually marked ones, the marking process can have significant impact to the results. When marking the correct objects in the scene, it is sometimes difficult to determine, whether an object is still valid for detection or not. For example if the object's position is on the edge of the detected scene, or even partially out of

the scene, it is very hard to say, whether the detecting algorithm should detect this object or not.

Another error in the detection evaluation can appear when the positions of the detected and marked objects are about the same but they vary in their sizes more than the previously set limit. This evaluation error can cause that some of the detected objects are incorrectly considered as False positives instead of the True positive ones. All of the manually marked objects as well as the limits used for evaluating the detections were set to the disadvantage of the algorithm. In another words, the evaluation process rejected some of the correctly detected objects and caused that the amount of False positives was higher than in reality. This settings should ensure, that in the reality, the algorithm will behave in the same way or in better way than while the testing, but not worse.

From the parameters influence testing can be seen, that the correct settings of the *scale* and *score* parameters has big impact on the quality of the detection. The *scale* factor settings is important mainly for the amount of correctly detected objects rather than for the amount of the False positive detections. The settings of the *scale* factor which ensures the highest amount of True positive detections with the lowest False positives rate is between 1,1 to 1,2. The *score* parameter determines the amount of correctly detected objects but mainly, its correct settings can reduce the amount of the False positive detections. The settings of the *score* parameter which ensures the highest amount of True positive detections with the lowest False positives rate is between 700 to 720.

The False positives rate can be also reduced by setting the scene correctly. The background should contain as small amount of objects with the temperature similar to the humans one as possible. Otherwise, there can occur constant False positive detections.

The quality of the detection depends also on the selection of the pattern for detection. The pattern should have key significant features which are contained in each of the detected objects. There were five patterns for detection which were tested and each of them had a type of a slightly modified head. There were two patterns which came out of the testing as the best ones, i.e. the patterns 6.6a and 6.6c. The pattern 6.6a had the highest True positive rate, the pattern 6.6c had lower True positive rate but also detected lower amount of False positive detections.

The last thing tested was influence of the different color palettes on the quality of the detection. Each of the palettes can point out different areas in the image and therefore some key features might be more distinct. The testing showed that there are several palettes suitable for face detection as well as there are palettes which are not suitable at all. The five best testing palettes in the Figure 6.9 can be considered as equal for the face detection. Their True positive rate is higher than 96% and there is not one which could be considered as the best one.

Chapter 7

Conclusion

This paper presents an application, designed for detecting people with elevated temperature using thermocamera. For this purpose, an algorithm capable of human recognition and temperature measurement in a radiometric image was developed. The detected persons in the radiometric image are transformed into a generic visible spectrum camera image. The video output is accessible on client machines via TCP/IP protocol. Detected persons with elevated temperature can be stored on a hard-drive along with any meta-data, such as date, time or measured temperature. This application provides an user-friendly interface, which enables the operator to easily identify the detected person and share the information with others over the network.

There are two algorithms used for face detection, i.e. the OpenCV for visible spectrum and Greyscale Value Pyramid for detecting faces in radiometric images. The main emphasis is put on detecting faces in radiometric image with the Greyscale Value Pyramid algorithm while the OpenCV algorithm is introduced only as an alternative and can be combined with the Greyscale Value Pyramid algorithm in the future work. Motivation for detecting faces both in the radiometric and visible spectrum images can be higher robustness of the algorithm and resistance to the environmental influences.

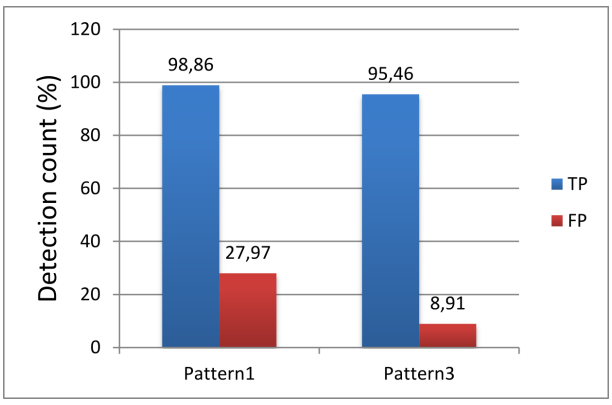


Figure 7.1: Graph of the True positive and False positive rate with the best settings for two of the patterns.

Figure 7.1 shows the graph of the True positive and False positive rate with the optimal settings of the algorithm's key parameters for two of the patterns for detection. The scene was set in way that it simulated persons passing through a hallway towards to the cameras with one and multiple persons passing by.

Total number of True positive detections, i.e. the amount of faces in all of the video frames was 717. As can be seen, the pattern 1 (6.6a) had the highest True positive rate which was almost 99%, the pattern 3 (6.6c) had lower True positive rate, a little bit over 95%, but also detected lower amount of False positive detections. As described in the previous chapter, the True positive rate and False positive rate in the Figure 7.1 is the worst case scenario. In reality, the amount of False positive detections should be lower with the same amount of correctly detected persons.

The quality of the face detection with the Greyscale Value Pyramid algorithm seems to be good enough so there is no particular need for using the OpenCV algorithm for detecting faces in the visible spectrum images. Excluding the OpenCV algorithm from the Main application can save a lot of the computational time and can speed up the application significantly.

The scene with detected persons can be displayed in several ways. Persons with overlayed rectangles over their heads are shown either in the radiometric image with the choice of different color palettes, in the visible spectrum image or in the mixed image. This mixed image is a result of the radiometric and visible spectrum images fusion.

Resulting visible spectrum images are sent via the TCP/IP protocol to the network and can be displayed with multiple external client LabVIEW applications. An improvement can be done by compressing these images to the MPEG or H.264 format so they can be displayed by a freeware application, e.g. the VLC player.

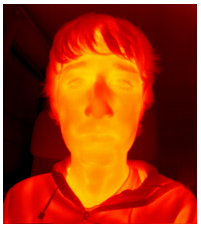
This work establishes a complete, user-friendly solution, which can be deployed to various environments. The possibility of imaging face detections in both radiometric and visible spectrum images provides quick and efficient means of detection of potentially sick people.

Appendix A

(a) Temperature

(b) Black Red

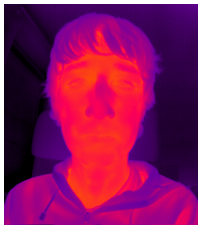
(c) Grey



(d) Fire

(e) Sepia

(f) Grad



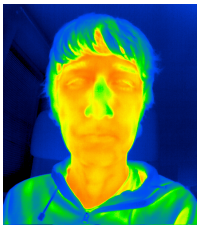
(g) Iron 2

(h) WBRGB

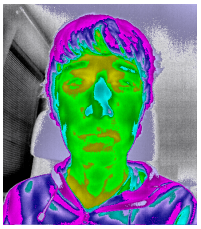
(i) Iron 1

(j) BWRGB

(k) Rainbow 2



(l) Natural



(m) Steps

Figure A.1: The set of all testing palettes.

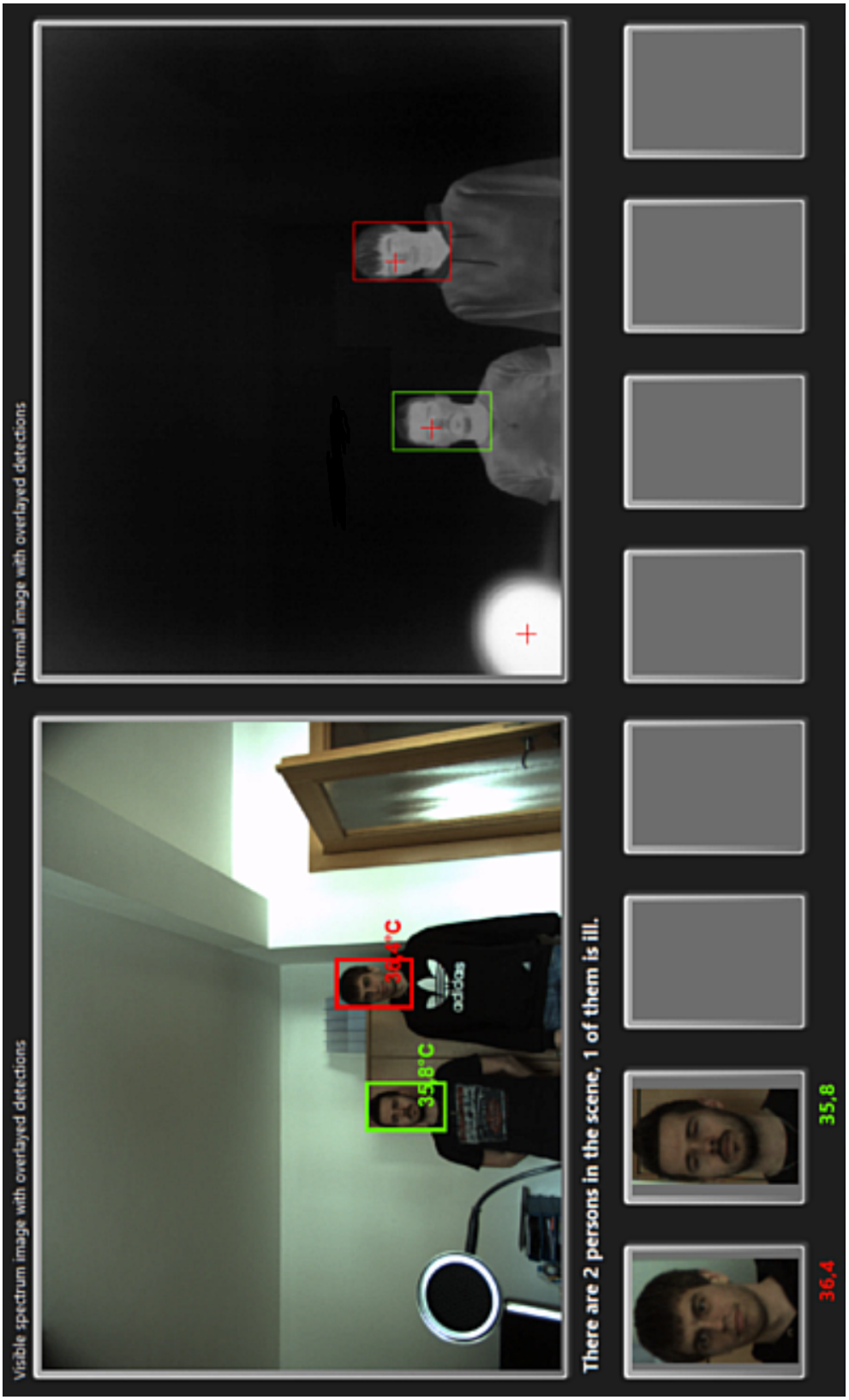


Figure A.2: An example of detection of the two persons visualized in the visible spectrum image. The left one has temperature lower than the limit, which is set to the 36 °C, the temperature of the right one's is elevated. The detected faces are ordered and displayed in descending order by their temperature.

Figure A.3: A printscreen of the application with three persons in the scene, one of them is ill.

Bibliography

[1] Workswell, 2015. URL <http://www.workswell.cz>.

[2] Thermoteknix Systems Ltd. FevIR Scan Skin Temperature Measurement System. [online], 2014. URL <http://www.thermoteknix.com/wp-content/uploads/2014/08/fevir-scan-fever-screening.pdf>. Last visited: 2015-04-15.

[3] FLIR. Use of Thermal Imaging to Detect Elevated Body Temperatures. [online]. URL <http://www.flir.com/thermography/americas/us/view/?id=60114>. Last visited: 2015-04-14.

[4] LAND Instruments International. Human Body Temperature Monitoring System. [online]. URL <http://data.etherm.cz/land/aplikacni_pyrometry/hbtms/hbtms_dat_en.pdf>. Last visited: 2015-04-15.

[5] X. MALDAGUE. *Theory and practice of infrared technology for nondestructive testing*. Wiley series in microwave and optical engineering. Wiley, 2001. ISBN 9780471181903. URL <http://books.google.cz/books?id=ts9RAAAAMAAJ>.

[6] R. BHAN, et al. Uncooled infrared microbolometer arrays and their characterisation techniques (review paper). *Defence Science Journal*, 59(6):33–41, 2009.

[7] Optotherm. Microbolometers. [online]. URL <http://www.optotherm.com/microbolometers.htm>. Last visited: 2015-04-08.

[8] S. Ya ANDRUSHIN, et al. Formation of porous silicon on a non-conductive substrate and its use as a sacrificial layer. *Semiconductor Science and Technology*, 20(12):1217, 2005. URL <http://stacks.iop.org/0268-1242/20/i=12/a=013>.

[9] Infrared Cameras Inc. [online]. URL <http://www.infraredcamerasinc.com/Thermography-FAQ/Infrared-icons/Thermal-Camera-Resolution.html>. Last visited: 2015-04-08.

[10] Workswell. Thermal Vision PRO, 2015. URL <http://www.workswell.eu/thermal-vision-for-drones/>.

[11] *Introduction to LabVIEW*. Telemark University College, 2014. URL <http://home.hit.no/~hansha/documents/labview/training/Introduction20to20LabVIEW/Introduction20to20LabVIEW.pdf>.

[12] Martin ČERMÁK. Moderní měřicí systémy. diploma thesis, Masarykova Univerzita v Brně, 2003.

[13] *NI-IMAQdx User Manual*. National Instruments, 2015. URL <http://www.ni.com/pdf/manuals/371970a.pdf>.

[14] Pleora Technologies. ebus sdk - data sheet. [online]. URL <http://www.pleora.com/support-center/documentation-and-downloads/ebus-sdk-data-sheet>. Last visited: 2015-04-08.

[15] *Thermal imaging guidebook for industrial applications*. FLIR Systems AB, 2015. URL <http://www.flir.com/uploadedFiles/Thermography/MMC/Brochures/T820264/T820264_APAC.pdf>.

[16] AXIS Communications. An explanation of video compression techniques. [online]. URL <http://classic.www.axis.com/files/whitepaper/wp_videocompression_33085_en_0809_lo.pdf>. Last visited: 2015-03-29.

[17] S. VETRIVEL, et al. An overview of mpeg family and its applications. *Indian Journal of Computer Science and Engineering*, 1(4):240–250, 2010. URL <http://www.ijcse.com/docs/IJCSE10-01-04-01.pdf>.

[18] A. LATTRE, et al. Videolan streaming howto, 2002. URL <http://images.videolan.org/doc/streaming-howto/en/streaming-howto-en.pdf>.

[19] OpenCV. URL <http://www.opencv.org/>.

[20] P. JONES, Paul VIOLA, and Michael JONES. Rapid object detection using a boosted cascade of simple features. In *University of Rochester. Charles Rich*, pages 905–910, 2001.

[21] Michael KEARNS. Thoughts on hypothesis boosting. Unpublished manuscript 45 (1988): 105.

[22] Yoav FREUND and Robert E. SCHAPIRE. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, 1995.

[23] Cascade of Classifiers. [online]. URL <http://photos1.blogger.com/blogger/634/1895/1600/cascade.jpg>. Last visited: 2015-03-29.

[24] Viola-Jones Face Detection. [online], 2012. URL <https://sites.google.com/site/5kk73gpu2012/assignment/viola-jones-face-detection>. Last visited: 2015-04-12.

[25] Keyence. Machine Vision Technology. [online]. URL <http://www.visionsystem.com/technology/position_detection.php>. Last visited: 2015-03-29.

[26] Mike BROOKES. Correlation. [online]. URL <http://www.ee.ic.ac.uk/hp/staff/dmb/courses/E1Fourier/00800_Correlation.pdf>. Last visited: 2015-03-29.

[27] K. BRIECHLE and U. D. HANEBECK. Template matching using fast normalized cross correlation. In *Proceedings of SPIE: Optical Pattern Recognition XII*, volume 4387, pages 95–102, 2001.

[28] E. H. ADELSON, et al. Pyramid method in image processing. *RCA Eng.*, 29(6):33–41, 1984.

[29] Vienna University of Technology. Gray Level Pyramids. [online]. URL <http://oldwww.prip.tuwien.ac.at/research/research-areas/image-pyramids/gray-level-pyramids>. Last visited: 2015-03-29.

[30] Nikos DRAKOS. Gaussian and Laplacian Pyramids. [online]. URL <http://www.cs.utah.edu/~arul/report/node12.html>. Last visited: 2015-03-29.

[31] Veterinary Thermal Imaging Ltd. How A Thermal Imaging Camera Works. [online], 2015. URL <http://www.veterinary-thermal-imaging.com/thermography/how-thermal-imaging-caemras-work>. Last visited: 2015-04-14.

[32] Tomáš PAJDLA. Elements of geometry for computer vision. Technical report, CTU, 2013. URL <https://cw.felk.cvut.cz/courses/GVG/2013/Lecture/GVG-2013-Lecture.pdf>.

[33] Jules BLOOMENTHAL and Jon ROKNE. Homogeneous coordinates. Technical report, The University of Calgary, 2013. URL <http://www.mat.ucsb.edu/594cm/2010/Week1/homog-coords.pdf>.

[34] Tom FAWCETT. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8): 861–874, 2006. URL <http://portal.acm.org/citation.cfm?id=1159475>.