BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

DETECTION, EXTRACTION AND MEASUREMENT OF THE LENGTH AND WIDTH OF THE METACARPAL BONES IN IMAGES
DETEKCE, EXTRAKCE A MĚŘENÍ DÉLKY ZÁPRSTNÍCH KOSTÍ V RENTGENOVÝCH SNÍMCÍCH LIDSKÉ RUKY

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR   **DANIEL PAUL**
AUTOR PRÁCE

SUPERVISOR   **Prof. Ing., Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2020

Department of Intelligent Systems (DITS)   Academic year 2020/2021

Bachelor's Thesis Specification

Student: **Paul Daniel**

Programme: Information Technology

Title: **Detection, Extraction and Measurement of the Length and Width of the Metacarpal Bones in Images**

Category: Image Processing

Assignment:

1. Study the literature on bone imaging and processing.
2. Propose an algorithm for detecting, extracting and measuring the length and width of the metacarpal bones in images.
3. Implement the proposed algorithm and test it on the available database of the Department of Anthropology, Faculty of Science, Masaryk University.
4. Evaluate the achieved results and discuss possible extensions.

Recommended literature:

- LIU, Zhi-Qiang; AUSTIN, Timothy J.; MOORE, Daniel. Image processing techniques for bone image analysis. In: *Proceedings., International Conference on Image Processing*. IEEE, 1995. p. 458-461.
- CHALECHALE, A.; BAHRI, A.; VATANCHIAN, M. Vision-based bone image recognition using geometric properties. *Iranian Journal of Science and Technology*, 2010.
- DROZDOVÁ, Eva. Základy osteometrie. 1. vyd. Brno: *Nadace Universitas Masarykiana*, 2005. 196 s. Panoráma biol. a sociokulturní antropologie 18. ISBN 80-7204-291-2.

Requirements for the first semester:

- Items 1 and 2.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.**

Head of Department: Hanáček Petr, doc. Dr. Ing.

Beginning of work: November 1, 2020

Submission deadline: May 12, 2021

Approval date: November 11, 2020

Abstract

A human body has been a point of the many studies for centuries. We want to discover everything there is to know about us, how we function and why we function that way. Scientists provide us with research that clarifies these questions. We should be a part for this research and enrich our knowledge with contribution to science. That's why this work will be focused on detecting, extracting and measuring metacarpal bones in images. The work will provide the reader with knowledge about certain image processing methods and metacarpal bones. It will introduce process of extracting their contour and show a way to measure their width and height.

Abstrakt

Ľudské telo bolo po celé storočia bodom mnohých štúdií. Chceme zistiť všetko, čo je možné sa o nás dozvedieť, o tom, ako fungujeme a prečo tak fungujeme. Vedci nám poskytujú výskumy, ktoré objasňujú tieto otázky. Mali by sme byť súčasťou tohto výskumu a obohatiť naše vedomosti prispievaním do vedy. Preto bude táto práca zameraná na detekciu, extrakciu a meranie metakarpálnych kostí na obrázkoch. Táto práca poskytne čitateľovi vedomosti o určitých metódach spracovania obrazu a záprstných kostiach. Predstaví proces extrakcie ich kontúr a ukáže spôsob, ako merať ich šírku a výšku.

Keywords

metacarpal bones, image processing, detection, extraction, length and width measurement, filter, HSV model, morphological transformation, boundary, pixel, dpi, euclidean distance

Klíčová slova

metakarpálne kosti, spracovanie obrazu, detekcia, extrakcia, meranie dĺžky a šírky, filter, model HSV, morfologická transformácia, hranica, pixel, dpi, euklidovská vzdialenosť

Reference

PAUL, Daniel. *Detection, Extraction and Measurement of the Length and Width of the Metacarpal Bones in Images.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D.

Rozšířený abstrakt

Ľudia skúmali svoje telo po celé storočia. Je tvorené štruktúrou rôznych buniek, tkanív a orgánových systémov, avšak stále nie je celé preskúmané, a preto sa snažíme o ňom dozvedieť sa čo najviac. Táto ľudská nevedomosť je dôvodom, prečo vedci uskutočňujú výskumy, ktoré objavujú o našom tele nové skutočnosti. Bioinformatika je jedna z vedných disciplín, ktoré prispievajú ku skúmaniu ľudského tela pomocou aplikovanej matematiky a počítačových technológií. Spracovávajú rôzne typy dát, simulujú prostredia alebo vykonávajú merania častí tela. Výsledok ich snaženia prispieva k celkovému pochopeniu nášho tela, aj sveta okolo nás.

Cieľom tejto práce bolo vytvoriť program, ktorý v naskenovanom snímku detekuje záprstnú kostičku, extrahuje jej kontúru a následne odmerá jej šírku a dĺžku. Ako testovacie dáta bola použitá aténska zbierka záprstných kostí. Tieto kosti boli naskenované so šedým odtieňom pozadia, z dorzálnej a radiálnej strany. Ich štruktúra sa skladá z hlavy, tela, podstavy a krku. Niektoré kosti boli na snímkoch prichytené plastelínou, aby držali na mieste pri skenovacom procese.

Prvým krokom bola detekcia kosti na snímku. Aby sa ľahšie identifikovala kosť na snímku, bolo treba zdôrazniť jej charakteristické črty. Tento krok vyžadoval použitie rozmazávajúceho filtru na pôvodný snímok. Táto práca preskúmala použitie troch typov takýchto filtrov - Gaussov, mediánový a bilaterálny filter. Ďalším krokom bolo vytvorenie masky záprstnej kosti. Na vytvorenie masky boli použité dve metódy - prahové hodnoty farebného modelu HSV reprezentujúci odtieň, sýtosť a hodnotu jasu farby a Cannyho hranový detektor, ktorý detekuje hrany kosti. Tieto metódy boli zvolené, pretože metóda prahových hodnôt HSV farebného modelu dokázala detekovať aj plastelínu kosti, ale nebola veľmi adaptívna na farebnú rozmanitosť kostí a Cannyho hranový detektor nedokázal spoľahlivo detekovať hrany plastelíny kvôli jej hladkosti povrchu. Následne sa maska upravila pomocou morfologických transformácií. Transformácia zatvorenia zacelila masku kosti vytvorenú predchádzajúcimi metódami, následne sa vyplnilo vnútro kontúry kosti a transformácia otvorenia vyčistila väčšinu šumu, ktorý zostal v snímku. Týmto procesom sa vytvorila finálna maska záprstnej kosti. Nakoniec sa použila bitová operácia A (and) medzi maskou a pôvodným snímkom. Tým sa vytvoril snímok, na ktorom bola reprezentovaná samotná záprstná kosť.

Nasledujúcim krokom bola extrakcia kontúry. Vykonaná bola pomocou vytvorenia prahu medzi maskou a pozadím snímku. Týmto spôsobom sa vytvoril čiernobiely binárny snímok, z ktorého sa kontúra dala detekovať. Následne sa vybrala kontúra, ktorá mala najväčší povrch reprezentujúca metakarpálnu kosť, zvyšok bol šum v snímku, ktorý tam zostal po kroku detekcie. Výsledkom bolo pole pixelov, s ktorými bolo možné robiť výpočty.

Posledným krokom bolo vypočítať samotnú šírku a výšku záprstnej kosti. Tento výpočet bol vykonaný pomocou Euklidovskej vzdialenosti medzi dvoma pixelmi, ktoré reprezentujú danú výšku, alebo šírku. Výškou kosti bola definovaná vzdialenosť úsečky medzi najvrchnejším a najspodnejším pixelom kontúry. Šírka bola definovaná ako najkratšia vzdialenosť úsečky medzi pravou a ľavou stranou kosti. Preto bolo potrebné si kosť rozdeliť na polovicu a porovnávať vzdialenosť pixelov z dvoch strán. Následne boli výsledné hodnoty prevedené zo vzdialenosti v pixeloch do vzdialenosti v ľudských jednotkách - milimetrov. Táto konverzia vyžadovala vedieť akým DPI (počet bodov na palec) bola kosť naskenovaná. Výsledné hodnoty výšky, dĺžky, pixelov reprezentujúcich výšku a dĺžku a pixelov kontúry boli spracované do formátu CSV (hodnoty oddelené čiarkami), pre prípadné ďalšie použitie.

Detection, Extraction and Measurement of the Length and Width of the Metacarpal Bones in Images

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D. The supplementary information was provided by Assoc. Prof. RNDr. Miroslav Králík. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

<div align="right">

....
Daniel Paul
May 11, 2021

</div>

Contents

Chapter 1

Introduction

1.1 Motivation

Humans have been studying their body for centuries. It is what defines us, what we are made of - everything in our body has its purpose. It is only natural that we want to know as much as possible about ourselves, how things function, what is their purpose, what are we made of. That's why scientists conduct research on various topics about our body - they simply want to understand as much as possible, because its human nature to gain knowledge. Bioinformatics is one of the disciplines helping scientists to ease analyzing their research. It uses the combination of IT technologies, biochemistry and applied mathematics to process data. I want to contribute to the research by helping Department of Anthropology, Faculty of Science, Masaryk University with measuring width and height of metacarpal bones as it will contribute to the further understanding of our body.

1.2 What will this thesis discuss

This thesis will discuss several topics. It is organised as follows. Chapter 2 will examine required knowledge concerning the matter of metacarpal bones - what they are, their structure in order to determine the best method of processing them. Subsequently it will look at various image processing methods and how to use them with the purpose of accomplishing measurements of the bone. This chapter will furthermore cover filters used to blur the image, how to create a mask with HSV color threshold, what is required to detect edges with canny edge detection method and it will discuss how to correct minor errors in the mask with morphological transformations. Afterwards it will also analyze how to extract the metacarpal bone from the image with finding its boundary and how to use that boundary to measure its height and width. Chapter 3 will cover already existing solutions (partial or complete) with intention of broadening the horizon of our possibilities and describe my own approach to solving this assignment. Consequently it will clarify how I designed and implemented mentioned theory into practise. This chapter will propose step by step analysis on how was the solution created. Focus of chapter 4 will be testing during implementation and my trial and errors that led to the final solution. Ultimately it will present other possibilities that I have considered to use during the development and problems that occurred and how I managed to solve them. Lastly in chapter 5 I will briefly summarize the whole process, what I learned during this time and what I managed to achieve and cover possible extensions that could be made in the future.

Chapter 2

Metacarpal bones and methods of their Detection, Extraction and Measurement of the Length and Width in scanned images

This chapter serves as clarification of needed theory for solving this assignment. It will look at what are metacarpal bones and describe their structure, the process of detection, extraction and measurement of height and width of metacarpal bones and various methods that are being used for achieving desired result. Additionally this chapter will cover the output choice for our implementation.

2.1   Definition and structure of a metacarpal bone

Metacarpal bones or **metacarpus** consist of five cylindrical bones that create a part of the skeletal hand located between the phalanges of the fingers (finger joints) and the carpal bones of the wrist. [13]

Human metacarpal bone consists of:

- **Head** represents oblong surface markedly convex from before backward, less so transversely, and flattened from side to side.

- **Body** prismoid in form, and curved, so as to be convex in the longitudinal direction behind, concave in front. The medial and lateral surfaces are concave. The dorsal surface presents a smooth, triangular, flattened area. This surface is bounded by two lines, which converge and meet some distance above the center of the bone and form a ridge which runs along the rest of the dorsal surface.

- **Base** is of a cuboidal form and broader behind than in front: it articulates with the carpal bones and with the adjoining metacarpal bones

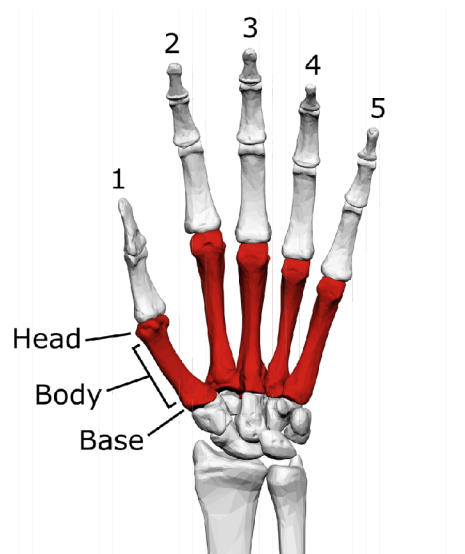- **Neck** represents the transition zone between the body and the head.

Figure 2.1: Metacarpal bones on a human hand (highlighted in red) [4].

2.2 Detection

There are many ways how to approach detecting an object in the image. My initial idea was to create an AI, that would be trained with machine learning methods to detect the metacarpal bone for me, but after some research I chose to go with the option of filtering image and excluding background.
Based on studied material I decided to use following approach:

- Import the image

- Blur the image

- Begin creating mask

- Convert image to HSV color model / Use canny edge detection

- Process image with morphological transformations

- Put mask over top of the original image

2.2.1 Blurring filters

Image convolution and kernel

Kernel is a small matrix of numbers that is used in image convolutions. [11]. It is used for blurring, edge detection, enhancing and more. This is achieved with convolution between an image and kernel.

Table below represents kernel size relative to smoothness/roughness change in the image:

Smoothness/Roughness	Kernel size [10]
Very Smooth	9·9
Smooth	
Semismooth	7·7
Smooth/rough	
Rough/smooth	5·5
Semirough	
Rough	3·3
Very rough	1·1

Convolution is the process of adding each element of the image to its local neighbours, weighted by the kernel - mathematical operation that is being applied is matrix multiplication. [11] In other words for each pixel, convolution operation multiplies the value of this pixel and values of the 8 surrounding pixels by the kernel corresponding value. Afterwards it adds the results, and the initial pixel is set to this final result value.

General form of matrix convolution is:

$$\begin{vmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{vmatrix} * \begin{vmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \cdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{vmatrix} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x_{(mi)(nj)} y_{(1+i)(1+j)} \qquad (2.1)$$

With focus on detecting the bone it is required to need to simplify the image. This is achieved by blurring and it allows us to easier pick apart colors and to emphasize features of the bone such as color and edges. Blurring is being achieved by using filters.
I was considering following:

Gaussian blur is achieved by convolving image with *Gaussian function*. It's main use is to reduce noise and reduce detail. Fourier transform of a Gaussian is another Gaussian therefore applying a Gaussian blur has the effect of reducing the image's high-frequency components - low pass filter. [9]
Gaussian function is represented by:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (2.2)$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution.

Median filter is a non-linear digital filter used to remove noise from image or signal. It is widely used in image processing because it can remove noise with preserving edges. It is based on processing signal entry by entry and replacing each entry with median of it's neighbours. Size of included neighbours is represented by so called „window". [8] In our

case (2D image) window is being represented with all entries in given radius.
Example with window size 3: $y = med(1, 3, 50) = 3$

Bilateral filter is a non-linear filter used to remove noise from signal or image. It is used to reduce noise, preserve edges and smooth the image. It's replacing the intensity of each pixel with a weighted average of intensity values from nearby pixels. Weight might be based on Gaussian distribution, but it is also dependent of range differences, such as color intensity, depth distance, etc. [12]
Bilateral filter is defined as:

$$I^{filtered}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\| I(x_i) - I(x) \|) g_s(\| x_i - x \|) \qquad (2.3)$$

with W_p defined as:

$$W_p = \sum_{x_i \in \Omega} f_r(\| I(x_i) - I(x) \|) g_s(\| x_i - x \|) \qquad (2.4)$$

where

$I^{filtered}$ is the filtered image
I is original image
x are coordinates of current pixel to be filtered
Ω is the window centered in x, so $x_i \in \Omega$ is another pixel to be filtered
f_r is range kernel for smoothing differences in intensities
g_s is the spatial kernel for smoothing differences in coordinates
W_p is normalization term

2.2.2 How to create mask of the bone

Converting image to HSV color model

After blurring the image the next step is to convert the picture to HSV color model in order to work with it further. HSV model is an alternative to RGB model and it is based on determining color with hue, saturation and value unlike RGB does with combining red, green and blue color.

Hue defines the color portion of the model expressed in angle from number 0 to 360 with colors: red, yellow, green, cyan, blue and magenta. Each of these colors take a portion of 60 degrees in the HSV model. [7]

Saturation determines the amount of gray in the image from 0 to 100%. The closer the value is to zero, the closer the saturation is to black color.[7]

Value describes brightness of the color in the image ranging from 0 to 100%. The closer the value is to zero the duller color in the image gets.[7]

So now that HSV model has been discussed, we will shift focus towards the method that is going to be used called color thresholding. It is a method that is being used for color filtering and specifies a range of colors that are required to be preserved and returns black and white image. Essential component of this method is color range of the background and the bones because from those it is possible to determine the best value for filtering out the background itself. After doing this, low range and high range of HSV values can be applied and finally filter the bone from the background. This will leave us with a mask of the metacarpal bone.

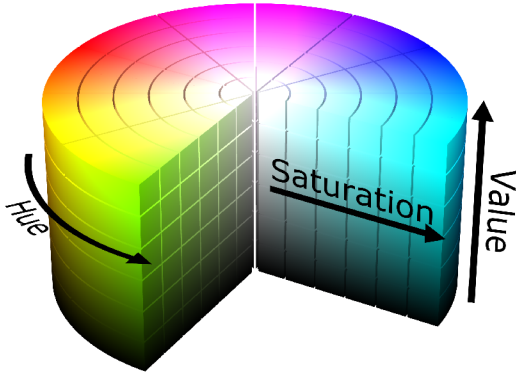Images below compare HSV and RGB models.
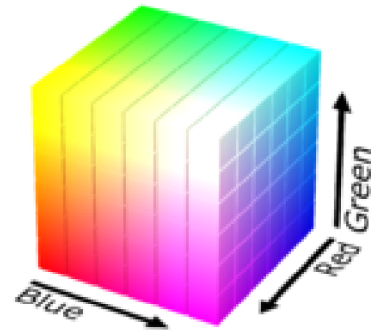

Figure 2.2: HSV color model [17].

Figure 2.3: RGB color model [17].

Canny edge detection

Another alternative how to detect bone in the image is to find its edge. [14] After blurring this is being done by filtering image with Sobel's kernel - an approximation to a derivative of an image. This is distinct in both y and x axis. If 3·3 matrix is being used, on x axis an edge is detected when there are negative numbers on the left side and positive on the right side and on y axis when there are negative numbers on the bottom and positive on the top.
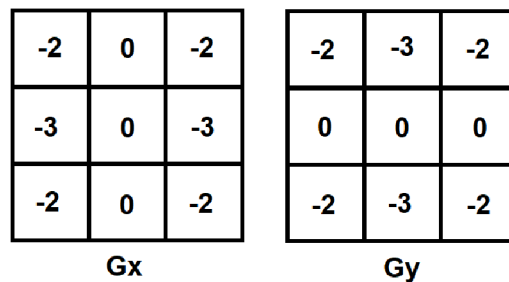

Figure 2.4: Sobel's filter matrices [14].

From images created by this kernel edge gradient and direction can be found for each pixel:

$$Edge\_Gradient(G) = \sqrt{G^2{}_x + G^2{}_y} \tag{2.5}$$

$$Angle(\Theta) = tan^{-1}(\frac{G_y}{G_x}) \tag{2.6}$$

8

After finding edge gradient the next step is to suppress all unwanted pixels. This is done by checking whether the given pixel is a local maximum in its neighbourhood for the direction of gradient.
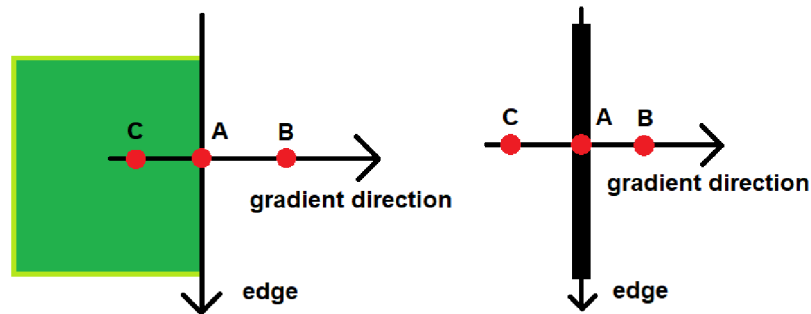

Figure 2.5: Non-maximum suppresion [14].

Last step is to do hysteresis thresholding. This step decides whether all edges are really edges and which are not. In order to do this two threshold values are needed, which will decide this - *minValue* and *maxValue*. Any edges above maxValue are considered as edges and those below minValue are being disposed. Edges that lie between these two values are conceived as edges based on their connectivity. If they are connected to the edge that is above maxValue they are also considered to be part of an edge, otherwise they are also discarded.
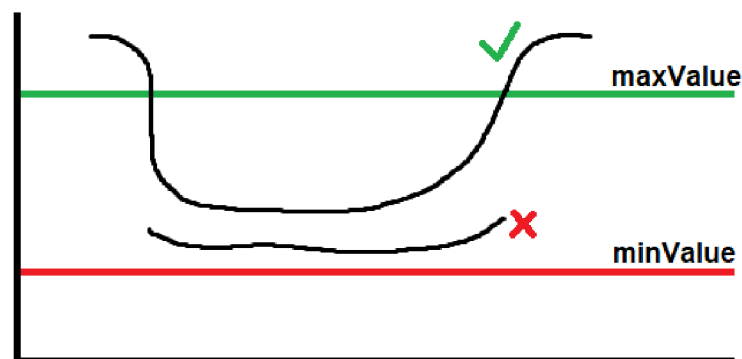
Figure 2.6: Hysteresis thresholding [14].

Morphological transformations

Morphological transformations are operations based on image shape. They are being applied in binary images mostly in grayscale. In order to make a morphological transformation you need an input image and *kernel*. Kernel decides the nature of the morphological operation (for example ellipse, rectangle,cross...) Morphological transformations are going to be used to complete the mask of the bone.

Two basic morphological transformations are:

Erosion erodes away boundaries of an object. Kernel slides along the image (2D convolution) and the pixel will be considered as 1 if all other pixels covered with kernel are 1.

Dilation opposite of erosion, kernel slides along the image and the pixel is considered as one if at least one pixel covered with kernel is 1. [5]
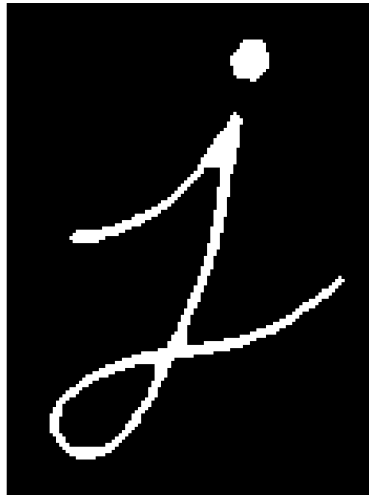

Figure 2.7: Original image [5].

Figure 2.8: Erosion [5].

Figure 2.9: Dilation [5].

In our case combination of the two is going to be used: **Closing** - dilation followed by erosion and **Opening** - erosion followed by dilation.

Figure 2.10: Opening [5].

Figure 2.11: Closing [5].

2.3 Extraction

After the metacarpal bone has been detected in the image it is possible to extract it using its boundaries. This can be done by finding metacarpal bone's contour.

Contour (boundary) can be defined as a set of points, which can be approached from outside and inside of the object. In our case, it is not necessary to approach these set of points from the inside, as the metacarpal bone has continues border, therefore all of the points can be found from the outside. Contour can be found by using a threshold on the mask. This threshold will differentiate between white and black colors, basically getting a binary image [19]. After getting the threshold the next step is to find the contours of the image by finding white pixels whose neighbour pixel is black. Using this process the metacarpal bone contour will be obtained which will be represented as a list of pixel sets with x,y coordinates. This contour will describe the shape of the bone.

2.4 Pixel to metric conversion, width and height calculation

In order to measure the metacarpal bone it is necessary to specify what is being measured and how it will be measured.

2.4.1 Computer metric compared to human metric

Computer images use different metric than humans - they use pixels as a measuring unit instead of human metres.

Pixel is a physical point on the raster image and the smallest element of a picture represented on the screen. [7]

Meter is the base unit of length in the International System of Units used by humans.

DPI or dot per inch is a measure of spatial printing and image scanner dot density. In other words it is a number of individual dots that can be placed in a line within the space of one inch.

With the aim of measuring a computer image in human metric when dpi is known and that one inch is 2.54 centimeters, pixel per centimeter ratio can be calculated as:

$$pixels\_per\_centimeter = \frac{2.54}{dpi} \qquad (2.7)$$

2.4.2 Measuring two points - Euclidian distance

Contour of the metacarpal bone that has been obtained in extraction process will be used for measuring purposes. As mentioned before the contour consists of set of pixels that create boundary around the metacarpal bone. These pixels can be treated as points and the distance between them can be measured using *Euclidian distance.*

Euclidian distance in 2D space is defined as:

$$(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2} \tag{2.8}$$

where p and q are two points whose distance is measured, q_1, p_1 represent x coordinate and q_2, p_2 represent y coordinate of these points.

Height of the metacarpal bone will be the distance between highest pixel in the head part of the bone and the lowest pixel in the base part of the bone.

Width of the metacarpal bone will be the smallest distance between the left and the right body part of the bone.

With this knowledge two pixels can be chosen, that will represent height, two pixels that will represent width, calculate their Euclidian distance and convert it to human metric with pixel per metric ratio.

2.5   CSV format

Resulting data was obtained from these calculations and it is necessary to display it in a format that is compact and easy to process. The file format that has been chosen for this purpose is CSV.

CSV format is a *delimited text file*, that uses separator to separate values. While there are various specifications and implementations for the CSV format, there is no real formal specification in existence, which allows for a wide variety of interpretations of CSV files. [18] However our implementation will apply following rules:

1. Each record will be located on a separate line, delimited by a line break (CRLF).
   aaa   bbb   ccc CRLF
   ddd   eee   fff CRLF

2. The last record in the file will have an ending line break.
   aaa   bbb   ccc CRLF
   ddd   eee   fff

3. There will appear a header on the first line of the file with the same format as normal record lines. It represents names corresponding to the fields in the file.
   field_name,field_name,field_name CRLF
   aaa   bbb   ccc CRLF
   ddd   eee   fff CRLF

4. Commonly employed delimiter *comma*, is going to be replaced with the whitespace character \t to separate values.
   aaa   bbb   ccc CRLF
   ddd   eee   fff

Chapter 3

Design and implementation

Focus of this chapter will be on existing solutions, outline design of the solution and take a look step by step on implementation of mentioned theory. Subsequently it will examine the result of each phase, data structure for containing output, output format and design of the user interface.

3.1 Existing solutions

In expectations of creating a good solution for this problem, it is important to know what programs are already available as of right now. They might provide inspiration, good insight and can widen the horizon of possible techniques that can be used in this project.

BoneXpert - In their words: „Bone age from a child's hand X-ray can be automatically measured by the BoneXpert software. The software delivers precise and standardised readings, thus overcoming the problem of considerable reader variability known from manual ratings." [1] They claim they have AI that can detect bone and using their program they can determine age of the bone. However they do not really describe the mechanism of their work. Judging by this, there exists a software which can detect a metacarpal bone, but they do not provide ability to measure it.

Figure 3.1: BoneXpert solution to detecting a bone [1].

Analyze Direct - this company has a tool called Bone Microarchitecture Analysis workflow [2] which they have described as:

Preprocessing - process where they isolate one specific bone from the image and correct it's orientation. This is irrelevant for me, because I have only one bone in the image and the bone is scanned in correct orientation from top to bottom.

Cortical Bone Segmentation - segmenting the bone to pieces in order to detect their cortical shell and their trabecular tissue regions.

Trabecular Bone Segmentation - completing segmentation and identifying the trabecular bone in the trabecular region. Two new regions, the trabecular bone and intra-trabecular space are derived in this step.

Calculate Bone Measurements - lastly measurements are automatically calculated for the trabecular and cortical bone and output in several comma-separated value (.csv) files.

This method is way to advanced for our needs, but inspiring nonetheless.
Following sections will look at how I approached the problem of metacarpal bone detection, extraction and measurement of width and height.

3.2   Structure of my approach

It is important to break down the problem into several parts:

- Detection - firstly it is important to detect metacarpal bone in the image.

- Extraction - secondly after detecting the metacarpal bone in the image, it is required to extract its contour (boundary).

- Measurement - lastly, obtained boundary pixels can be used to calculate metacarpal's bone width and height.

3.3   Resources and their use

I have been provided with the database of scanned metacarpal bones whose width and height needs to be measured from my supervisor. These bones are photographed from the front side (dorsalis) and the back side (radialis). Each bone has its specific shape (following the structure described in section 2.1) and they all have background that has similar color which we can use to our advantage and detect them this way. In some images plasticine is being used to make the bone stay in place during the scanning process, so we will also look at how I approached this complication. Structure of their name is also important as it will be employed for CSV output fields. For example 007_mc_d_1_a represents:

- First number - ID of the bone

- mc - the image contains metacarpus

- First letter - can either be d (dextra) meaning right hand or s (sinistra) meaning left hand

- Second number - signifies finger (1 - thumb, 2 - index, 3 - middle, 4 - ring, 5 - pinky)

- Last letter - can either be b (radial view of the bone, i.e. from the thumb side of the hand), a (view from the dorsal side), r (view from the dorsal side only of the bones of the left hand, but the image is mirror-inverted - we measure only the bones that look like the right, so that the comparison of the right and left side is not distorted by any visual laterality of the human eye) and t (view from the radial side of the bone of the left hand, image is mirror-inverted)

Some examples of bones that were provided in the database are illustrated on picture 3.2.
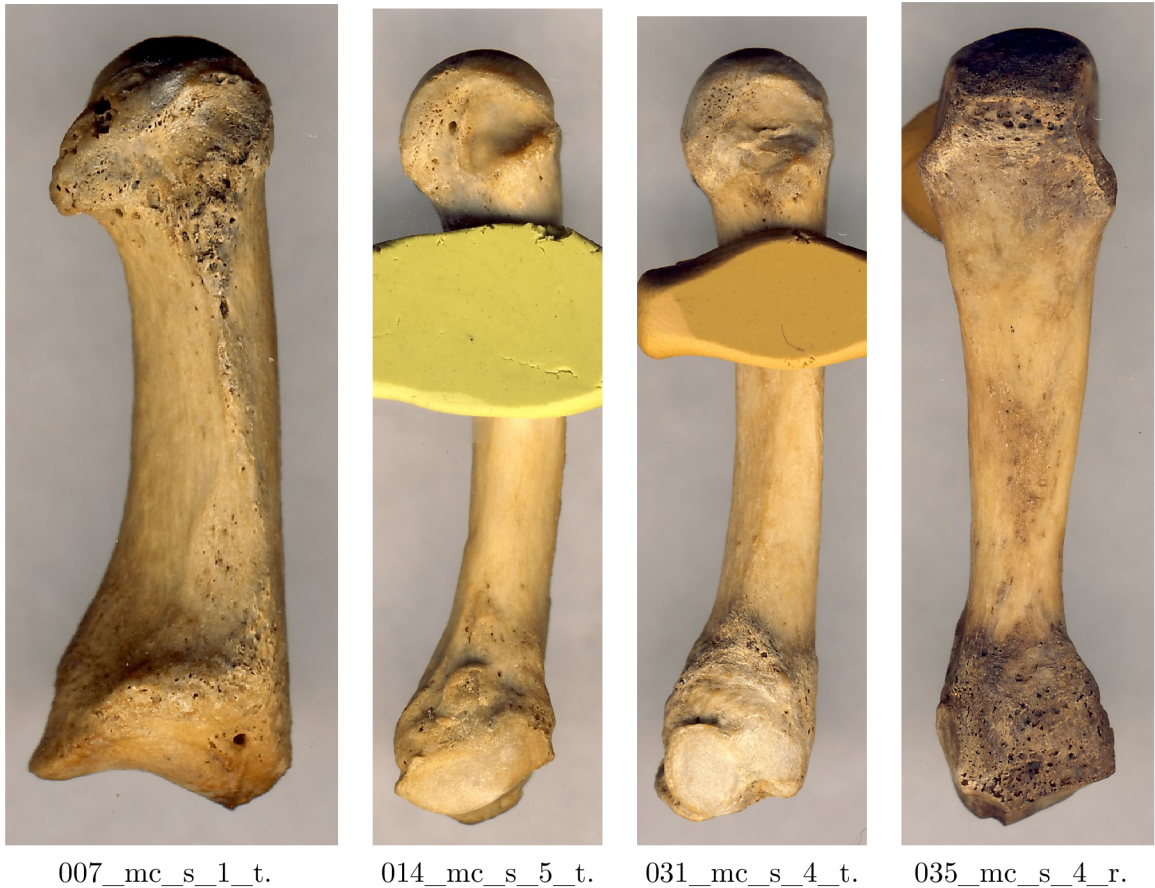

| 007_mc_s_1_t. | 014_mc_s_5_t. | 031_mc_s_4_t. | 035_mc_s_4_r. |

Figure 3.2: Bone examples in the database.

3.4 Software

For this project I decided to use programming language *Python* as it is simple language for implementation purposes with library *OpenCV* [16] which introduces a lot of useful image processing functions, specifically blurring filters, canny edge detection method, HSV thresholding, morphological transformations, contour detection and others. I will use *Numpy* [15] for data storing between calculations (numpy arrays) and mathematical calculations themselves, *Tkinter* [6] for creating user interface and it's functionality and lastly *Pyinstaller* [3] for the creation of the executable file.

3.5 Detecting the bone in the image

First of all I have imported the image on which I want to detect a metacarpal bone.

3.5.1 Application of blurring methods

For image filtering I've tried all methods mentioned in subsection 2.2.1.
In the case of *Gaussian blur* I've tried different sizes of kernel, but 5·5 seemed to work the best. With *median filter* after trying out different sizes, I've chosen kernel with size 5·5. Concerning the matter of *bilateral filter* I've chosen value of 15 for diameter of each pixel neighbourhood, and value of 75 for color space (the greater the value, the colors farther to each other will start to get mixed) and coordinate space (the greater its value, the more further pixels will mix together, given that their colors lie within the color space range.)
Results of comparison of these filters are illustrated in pictures 3.3 3.4 3.5. All three performed quite well. Colors were blended with blur and edges are perfectly preserved, which is needed for further processing.

Figure 3.3: Gaussian blur. Figure 3.4: Median filter. Figure 3.5: Bilateral filter.

3.5.2 Metacarpal bone mask creation

Converting to HSV color model

In order to convert our now blurred picture into HSV color model and create a mask it is essential to specify low and high range of HSV values. These values represent *color threshold* that will be filtered out of the image. Our goal is to detect bone in the image, so we have to choose such values, that will filter out the background. Results with different filters are illustrated on pictures A.1 A.2 A.3. The result of the conversion process from the blurred image to the HSV mask is illustrated on picture 3.6.

Figure 3.6: Mask created by HSV color threshold method.

Detecting bone with canny edge detection

Another method how to get mask is to detect edges of the metacarpal bone (discussed in 2.2.2). Fundamental things that need to be addressed are *minValue* and *maxValue* for hysteresis thresholding, which will decide whether to consider an edge or discard it completely. I picked values, that brought the best result - they are discussed in chapter 4. As you can see from the results of this method illustrated on pictures A.4 A.5 A.6, median filter was best in blurring the image as it preserved edges the best and the image did not contain any noise, which resulted in the best outcome. The result of the conversion process from the blurred image to the canny mask is illustrated on picture 3.7.

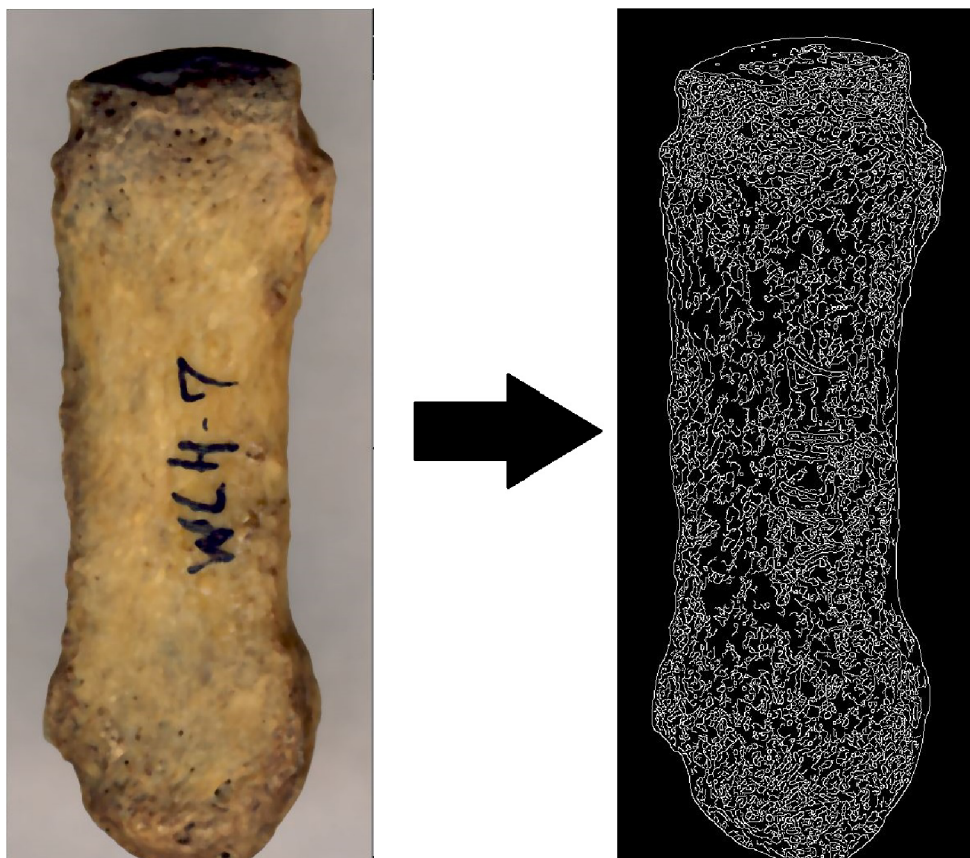Figure 3.7: Mask created by canny edge detection method.

Using morphological transformations

Now that I obtained the mask there is a need to consolidate the structure its structure. This is being done with morphological transformation - *closing* (filling up holes in the bone mask). Next step I implemented was filling up every contour that has been created to fill the bone without exposing too much background to the mask. Last part of this step was using morphological transformation - *opening* (filtering out excess parts in the background, if there are any left in the image). I have applied circular kernel for both morphological transformations, their sizes are discussed in chapter 4.

Ultimately as you can see from the pictures A.7 A.8 A.9 after applying these morphological transformations with HSV, the best result had gaussian and bilateral filter, because the remaining one has cut the top of the bone out of the mask. That's why I've decided to use gaussian filter from now on although I will show final results of all of these filters. When comparing images processed with canny edge detection A.10 A.11 A.12 you can see that all of filters have performed really well in the end, but I am believe overall results are better with median filter. The result of the conversion process from the HSV mask and canny mask to complete masks is illustrated on pictures 3.8 3.9.
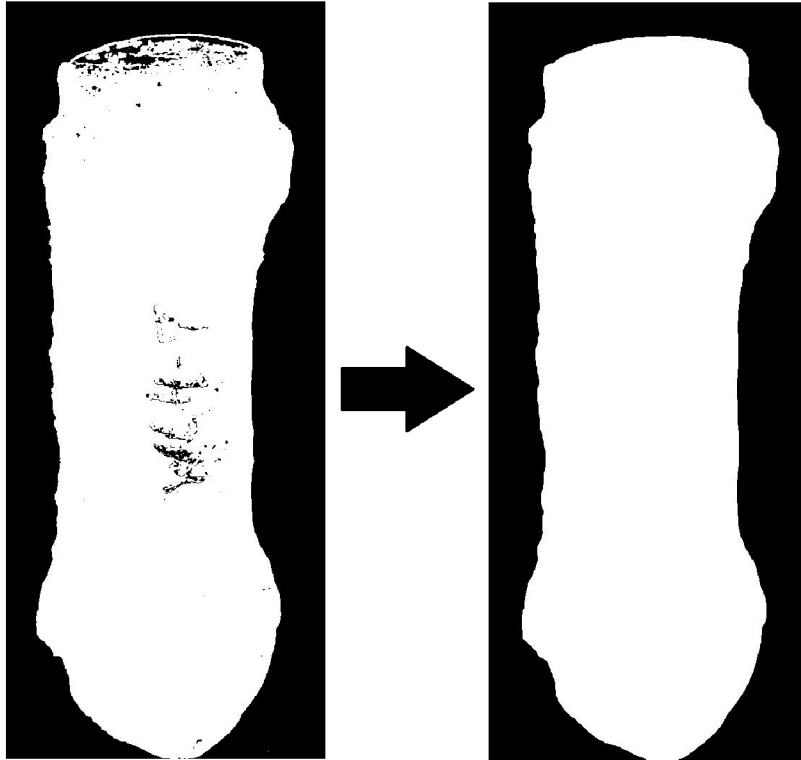
Figure 3.8: HSV mask completed with morphological transformations.
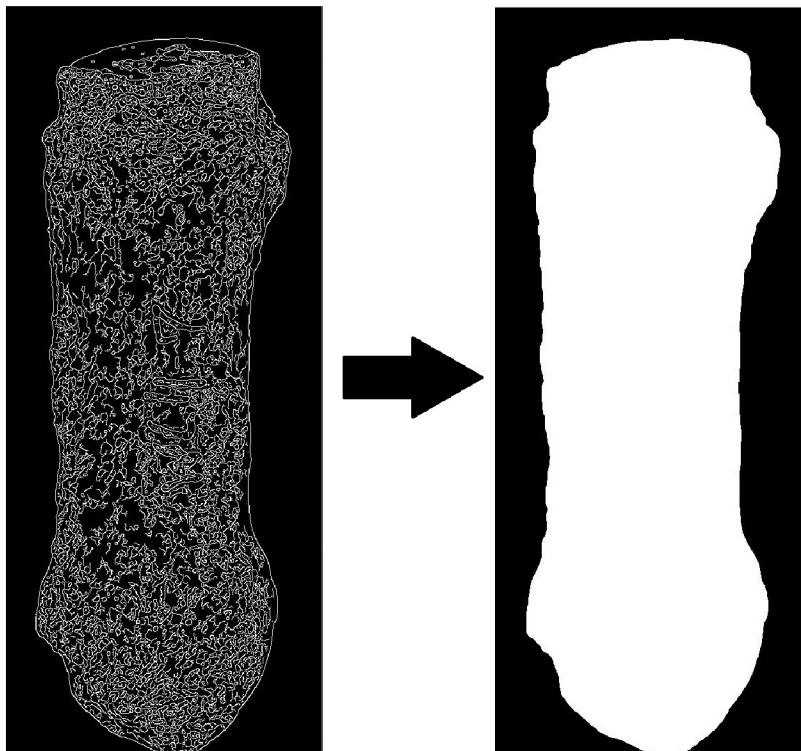

Figure 3.9: Canny mask completed with morphological transformations.

3.5.3 Conclusion of detection

Finally yet importantly after all this processing we are ready to use *bitwise and* operation on the image with created mask. This will leave us with image that contains only the metacarpal bone and the background will be excluded. Final results of HSV method and canny edge detection method can be seen on pictures A.13 and comparison between mask and final images is illustrated on pictures 3.10 3.11.

Figure 3.10: Result of HSV color threshold method with Gaussian filter.

Figure 3.11: Result of canny edge detection method with median filter.

Intending to confirm that our method works we need to test it on multiple metacarpal bones. As you can see from the results, our method for detecting metacarpal bones works.

| 007_mc_s_1_r.tif. | 020_mc_s_1_r.tif. | 035_mc_s_1_r.tif. |

Figure 3.12: Detection result of different metacarpal bones.

3.6   Extracting the contour

From detection process we are given a mask of the bone. In extraction phase it's crucial to select pixels of the bone contour correctly. As the image is now black and white, we will need to use threshold values that will detect contour based on these two colors. For this we can use *binary threshold type* and provide black and white color. Now that we have threshold we can finally get the outer contour of the metacarpal bone. Resulting contour is represented as a list of pixel sets [(x1,y1),(x2,y2),...]. In case more contours were found, because of some noise still being present in the image, we will pick the one with biggest area and discard the rest. In the next part of the implementation, I will use this list of pixel sets to calculate measurements of the bone.

To summarize, from pictures 3.13 you can see, that the contour has been constructed appropriately. It is being indicated by blue border around the bone.

Figure 3.13: Contour of bones 007_mc_d_1_a and 007_mc_s_1_r.

3.7 Finding pixels and measuring length and width

Now that we have contour we can finally measure the bone. I will focus on calculating height and width of the metacarpal bone.

We will find *height* by searching the list of pixel sets concentrating on the y coordinate. The bone height is defined as the length between the highest pixel and the lowest pixel. So if we search the list for those two pixels, we can calculate their euclidean distance to get the height measurement of the bone.

When calculating *width* of the bone we will need to concentrate in x coordinate. If we split the bone in half along the y axis, its width is defined as the smallest length between left part of and the right part of the bone. This will be a little more difficult process than calculating height as we will need to split the bone in the center to find the these pixels. For optimization purposes I have trimmed the list of pixel sets that are representing head and base of the bone (by removing the highest and lowest 20% of pixels) as it is in all cases wider than body which we will be using to calculate width. Using mean of x coordinates in the body of the bone results in finding sufficient center x coordinate of the bone. Consequently it is required to set apart pixels into two groups - for the left and for the right part of the bone. Now that we have left and right list of pixel sets we can find the smallest distance by measuring euclidean distance between each pixel from the left side with every pixel on the right side of the bone. This will result in finding the width.

22

Ultimately we will convert value of euclidean distance of found pixels to metric system with provided dpi (600) and knowing that dot per inch value is 2.54 it is simple to calculate bone's width and height as:

$$Height = \sqrt{(topPixel\_x - bottomPixel\_x)^2 + (topPixel\_y - bottomPixel\_y)^2} * (2.54/600)$$

$$Width = \sqrt{(leftPixel\_x - rightPixel\_x)^2 + (leftPixel\_y - rightPixel\_y)^2} * (2.54/600)$$

In conclusion as you can see from the pictures 3.14, there is a tiny difference in calculated values. This can be caused by error between conversion or difference between detected pixel. In this specific example, percentage difference in width is 0,031% and in height it is 0,0024%, which I consider rather small and I believe calculations were essentially rather precise. I also depicted left part of metacarpal bone body contour with green and right part with yellow color of the bone border.
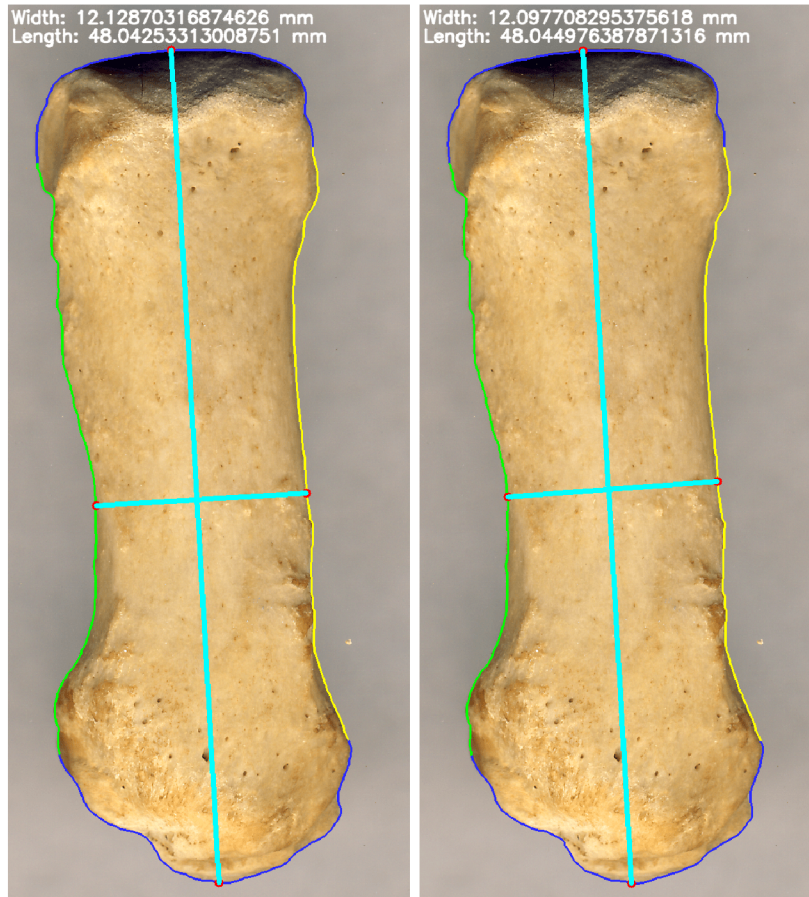

Figure 3.14: HSV (left) and Canny (Right) measurements of bone 007_mc_s_1_r.

3.8 Data structure, output format, User interface

3.8.1 Data class

In preparation for creating output for this assignment I needed to create a structure that would hold output data. This data is then further processed and exported into CSV format that was requested by Department of Anthropology, Faculty of Science, Masaryk University.

Data class is defined as:

```python
class Data:

    def __init__(self, width, length, width_pixels, length_pixels, contour):
        self.width = width # Width of the bone
        self.length = length # Length of the bone
        self.width_pixels = width_pixels # Set of pixels that defines a width distance
        self.length_pixels = length_pixels # Set of pixels that defines a length distance
        self.contour = contour # List of pixel sets that defines contour of the bone
```
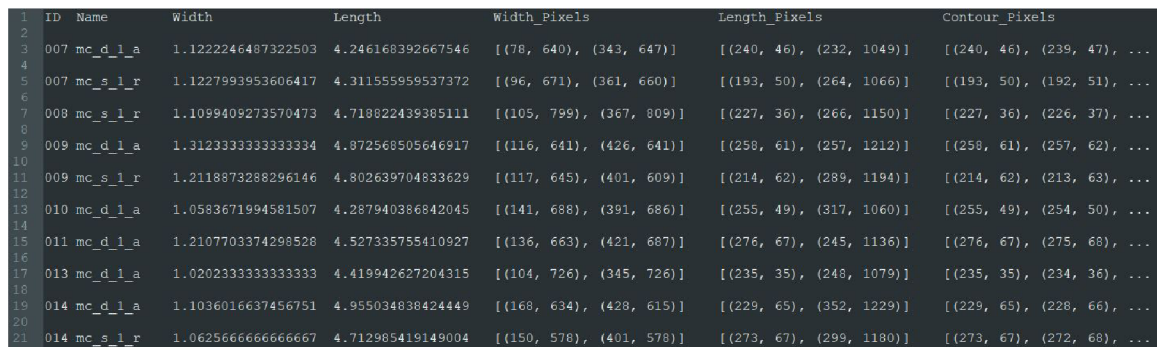
Listing 3.1: Class that encapsulates data.

3.8.2 CSV format

The next move is to create CSV file. For implementation I used standard CSV python library. I needed to define header to address individual fields, that will represent our data. It was agreed that the header will be specified as:
ID   Name   Width   Length   Width_Pixels   Length_Pixels   Contour_Pixels

Outcome of the CSV file can be seen on picture 3.15.

ID	Name	Width	Length	Width_Pixels	Length_Pixels	Contour_Pixels
007	mc_d_l_a	1.1222246487322503	4.246168392667546	[(78, 640), (343, 647)]	[(240, 46), (232, 1049)]	[(240, 46), (239, 47), ...
007	mc_s_l_r	1.1227993953606417	4.311555959537372	[(96, 671), (361, 660)]	[(193, 50), (264, 1066)]	[(193, 50), (192, 51), ...
008	mc_s_l_r	1.1099409273570473	4.718822439385111	[(105, 799), (367, 809)]	[(227, 36), (266, 1150)]	[(227, 36), (226, 37), ...
009	mc_d_l_a	1.3123333333333334	4.872568505646917	[(116, 641), (426, 641)]	[(258, 61), (257, 1212)]	[(258, 61), (257, 62), ...
009	mc_s_l_r	1.2118873288296146	4.802639704833629	[(117, 645), (401, 609)]	[(214, 62), (289, 1194)]	[(214, 62), (213, 63), ...
010	mc_d_l_a	1.0583671994581507	4.287940386842045	[(141, 688), (391, 686)]	[(255, 49), (317, 1060)]	[(255, 49), (254, 50), ...
011	mc_d_l_a	1.2107703374298528	4.527335755410927	[(136, 663), (421, 687)]	[(276, 67), (245, 1136)]	[(276, 67), (275, 68), ...
013	mc_d_l_a	1.0202333333333333	4.419942627204315	[(104, 726), (345, 726)]	[(235, 35), (248, 1079)]	[(235, 35), (234, 36), ...
014	mc_d_l_a	1.1036016637456751	4.955034838424449	[(168, 634), (428, 615)]	[(229, 65), (352, 1229)]	[(229, 65), (228, 66), ...
014	mc_s_l_r	1.0625666666666667	4.712985419149004	[(150, 578), (401, 578)]	[(273, 67), (299, 1180)]	[(273, 67), (272, 68), ...

Figure 3.15: Representation of CSV output.

3.8.3 User interface

Regardless of the fact that it is not the essence of this thesis, it is necessary to create a user interface for the usability of the program by users, so I will mention it. For implementation I used library called *tkinter*.

24

Mockup

I came to realization that I need a mockup to have a rough idea how things should look like. I followed the principles of a good user interface that should be easy to understand and intuitive to the user. My idea was to design the functionality so that the user would perform the steps from top to bottom. I came to a conclusion, that the user interface should consist of a button for entering path to the output folder, secondly radio buttons that will make selection between HSV thresholding method and canny detection method, thirdly *include images* checkmark which when checked will include images in the output folder and finally input button which will open up a file explorer for use to select a folder with metacarpal images that he wants to process. Constructed mockup can be seen on the picture 3.16
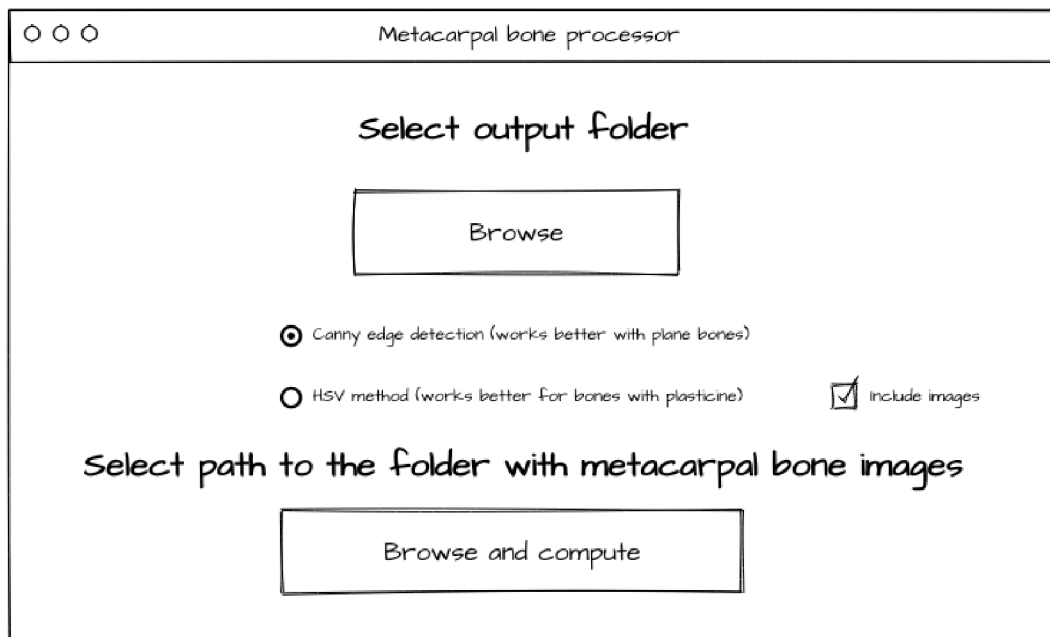

Figure 3.16: User interface mockup.

Functionality

When you select output path a a message *Path selected* will be displayed on the output path button. Afterwards you need to select a method which the bones will be processed by. After clicking on the input path button and selecting a folder, program iteratively checks all nested folders for images with *.tif* extension. Afterwards processing process will begin and text on the button will display how many bones have been processed. When the computing has been completed, the button will notify user with: „*All X bones processed*". When the user has not selected an output path after he tried to select input path a message *Select output path* is displayed on the output path button. Furthermore when user does not select input path a a message *Select input path* is displayed on the input path button.

Final user interface

Final version of the user interface pretty much follows a mockup as you can see on the picture 3.17. The only difference is styling of radio buttons. I decided to go with darker colors, as they are seemingly more pleasing to the eyes of the user.
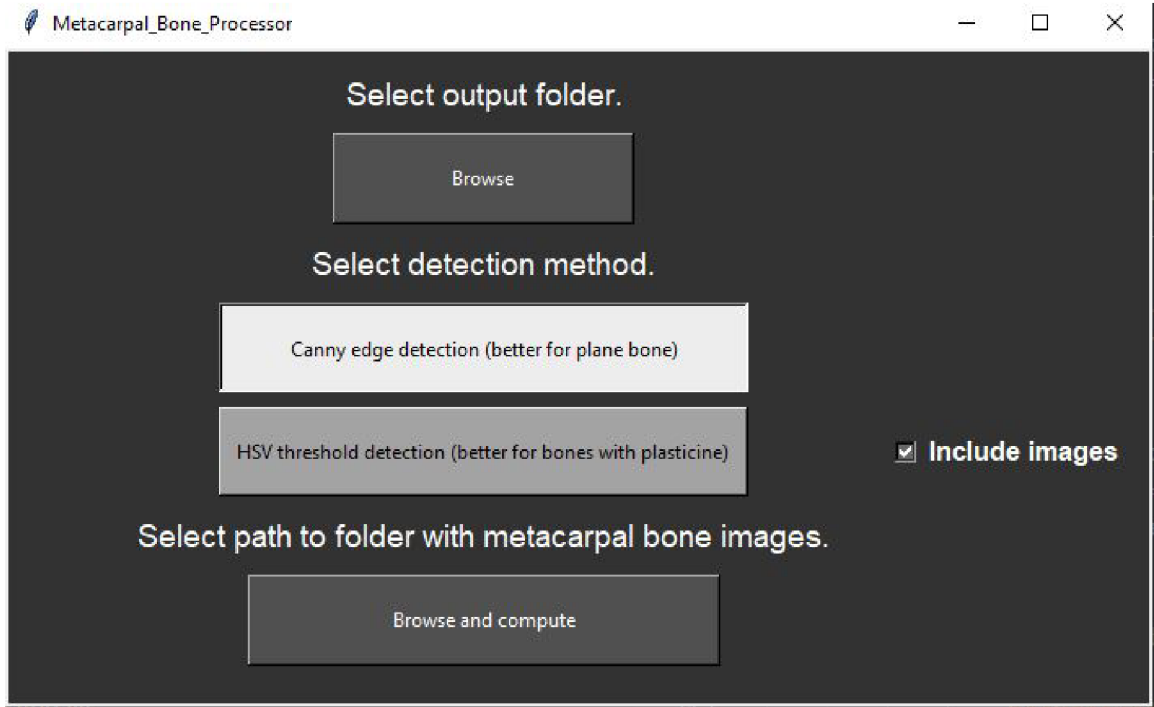

Figure 3.17: Final version of the user interface.

Chapter 4

Testing and evaluation

In this chapter, the testing process is described. It is necessary to get functional result, so I needed to make sure, I have tried all available methods and altered them, so the outcome is as accurate as possible. Finally it will describe how I corrected error by adjusting values and choosing the order of methods.

4.1 Testing

There have been quite a few things to consider throughout the development of the solution for this thesis. At each step of the process there were multiple approaches available and handful of alternatives that needed to be tried out in order to find the most optimal result. Sometimes it was estimated by knowledge gained from resources, other times it was principle of trial and error. Although I thought that I have found the best answer for the current step at the time, along the way I needed to backtrack to previous steps and modify them with intention of improvement that I wanted to accomplish in the following steps. There might exist a better approach, but with my research and testing I created the implementation that works on most images as intended.

4.1.1 Blur filters

During detection phase the first step was to blur the input image with a purpose of enhancing features of the metacarpal bone. As there are many blurring filters, it was quite a difficult task to choose the correct one. Research I have studied has provided me with some useful information, but in the end I had to try it out and see what results turned out best. On its own, the blurred image did not provide enough information for me to assess its usability, so I had to also implement start of the mask creation - HSV thresholding. Regardless of the fact that I was not completely sure whether I have chosen the correct color threshold that I want to detect, I finally had an image that was usable for testing.

Testing filters

From my research I decided to try three filters: Gaussian blur, median filter and bilateral filter. At first, I considered bilateral blur as the best option, because it has suggested excellent results. But as the time passed and I've implemented other parts of the program, I have switched to the Gaussian blur for HSV color thresholding method, because I have

figured that it enhances color features a little bit better than bilateral filter, even though both results were satisfactory for this purpose. Median filter here was out of the question, as it was cutting off darker parts of the bone. After some testing, I've chosen median filter with canny edge detection. I have also considered using Gaussian filter, but due to the fact that images filtered with this filter contained more noise than images filtered with the median filter, I decided to stick with the latter. Bilateral filter was not taken into account, as it did the worst out of all three. Comparison has been illustrates in previous chapter in images 3.3 3.4 3.5.

Filter kernel selection

Another thing that needed to be taken care of was size of kernel for filters. The problem was in choosing how smooth/rough we want our blur to be 2.2.1. I found out, that the same sizes of each kernel behaved the same for different filters, so it was only the matter of testing out which will blur the bone the best. I have tested sizes of 3·3, 5·5 and 7·7. The problem with the 3·3 kernel was that it insufficiently emphasized features and 7·7 has smoothened the image too much which transpired into worse results in mask creation stage. 5·5 kernel turned out to be the golden middle and smoothened the image just enough for HSV color thresholding and canny edge detection methods to work as intended.

4.1.2   Mask creation

Regarding mask creation, it was necessary to fine tune all values to detect features which are emphasized in the blurred image. Prior investigations have diverse approaches to, solving this problem. Essentially there are various methods on creating the mask of object in the image, which can lead to desirable outcome. After some research I decided to try two of them - HSV color thresholding for metacarpal bone images containing plasticine or canny edge detection for images without presence of plasticine, both combined with morphological transformations.

HSV color thresholding

Firstly I started just with HSV color thresholding and tried to find the correct threshold to filter out the background. I've switched to different values during implementation, and the problem was to find one constant that could be applied for all images. It wasn't easy, but in the end I figured out the numbers, that worked for the most images. Throughout testing this method on different samples though I found a very unpleasant issue, that was related to similarity between some portions of the bone and background. As you can see from the image 4.1 there is a clear interference in the part of the bone indicated with red, which is taken into consideration as section of the image that is being removed with the background. In this case, it was a serious issue, but I figured out that all I had to do was to decrease low saturation value just enough, so that morphological transformation opening could fix it in the next step.
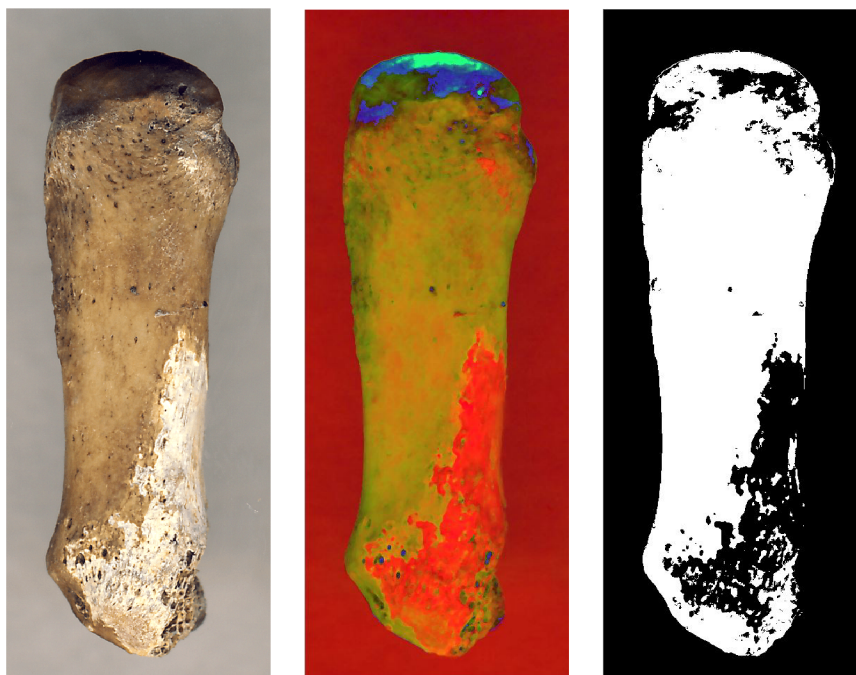
Figure 4.1: Comparison between plain image, HSV representation and mask.

BGR color thresholding

There was another method that I have tested - BRG color thresholding (CV library uses blue, red, green instead of red, green, blue). Although it brought some accurate results, it did not work as well as HSV variant, due to the fact that there were big color differences between colors of the bone and it was difficult to find constant values which would be a compromise and work for most of the images. To summarize it was easier to detect a bone in the image with hue, saturation and value, since saturation played a big role in the intensity of the color.

Canny edge detection

When it comes to canny edge detection the important parameters to set correctly are minimal and maximal value of hysteresis thresholding. Emphasis was placed on proper detection of outer edges, rather than on the inner part of the bone, because these edges are going to be enhanced with closing morphological transformation for the purpose of creating solid structure of the mask. After experimenting with some values, I figured out which were the most plausible, regardless of the fact that there has still appeared some noise which was considered as an edge. This noise was later removed with opening morphological transformation so it was not that serious issue to be worried about. The only issue that canny detection method brought up were images that contained plasticine. The problem that occurred was that the plasticine is too smooth to have edges detected properly and later I could not find a way to correct the mask with morphological transformations, so I decided to implement this method only for use on bones without plasticine. You can see this error on the picture 4.2.
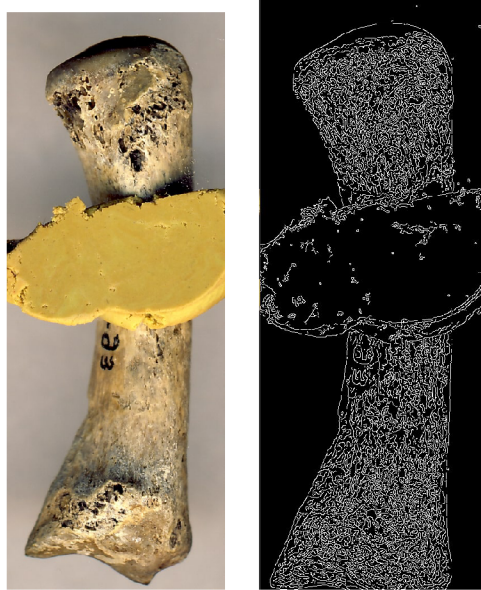
Figure 4.2: Plasticine being too smooth for canny edge detection.

Grayscale

Several studies agree that converting blurred image to grayscale before using canny edge detection is beneficial for better detection of edges. With expectations of improvement, I have examined and tested out this approach on some bone samples. In my case however, this concept did not perform as well as I hoped it would. In summary regardless of the fact that that the results were comparable, grayscaled image has proved to detect less edges inside the bone and I didn't want to risk that any edge on the perimeter of the bone is not going to be detected. From comparison of this testing in images 4.3 is apparent that the grayscaled image has detected less edges at the top and in the left middle part of the bone. Additionally some tiny noise is noticeable at the bottom right of the bone.
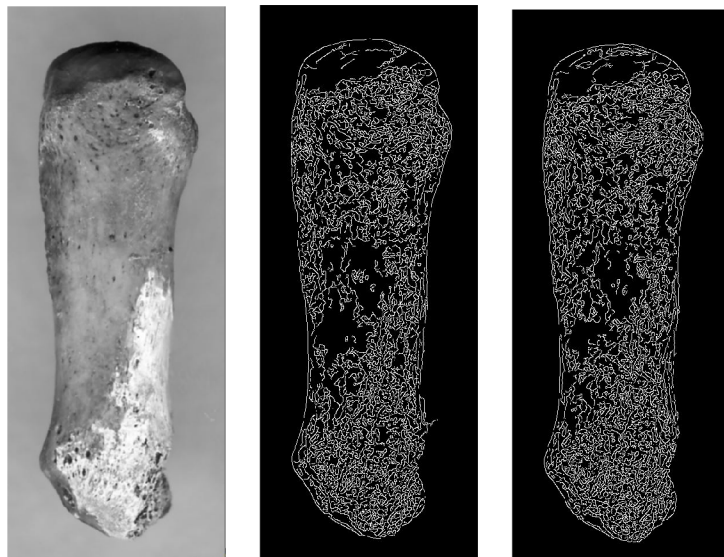
Figure 4.3: Comparison between grayscaled image (middle) and only blurred image (right).

Morphological transformations

Testing of this part was the crucial moment of the whole process in particular. In this stage I have faced the most problems during the development, considering that morphological transformations alter the obtained mask from the previous step. I had to choose the appropriate order of types of transformations themselves, their iteration count and kernel shapes and sizes for each of them.

For instance, when I decided to use kernel size that was too large for closing transformation, it resulted in contour of the bone being connected to the edge of the image 4.4 and when I used it for opening it in some cases removed the part of the bone that was detected with one of the previous methods. I came to conclusion that size of the kernel for morphological transformations should be 10·10 for canny edge detection processed images and 20·20 for HSV threshold method based on my testing. I also had to focus on which types of morphological transformation I will use. The logical solution for me was to use closing and opening on the bone with one iteration. At that point in time I have chosen small kernel and could not figure out a way to remove some bigger chunks of noise in the image with opening transformation so I used multiple iterations of dilation and erosion in a row. Later on I figured out that I could just increase kernel size and use just closing and opening.

Next step was to select the shape of kernel. There were three shapes available from CV library - cross, rectangle and circle. I have tested each of them, and the only one that served the purpose was circular kernel, so I decided to utilize it in all further development. Another issue I faced was that the opening morphological transformation destroyed part of the bone perimeter in the mask when using canny detection method, when closing transformation did not fill inside of the bone based on inner edges 4.5. I fixed this problem with filling up inside part of the bone based on compact perimeter that has been created by the closing morphological transformation and then used the opening transformation to clear noise from the image. In conclusion I believe I have tried most of methods in this field in vision of creating the best possible mask of the metacarpal bone.
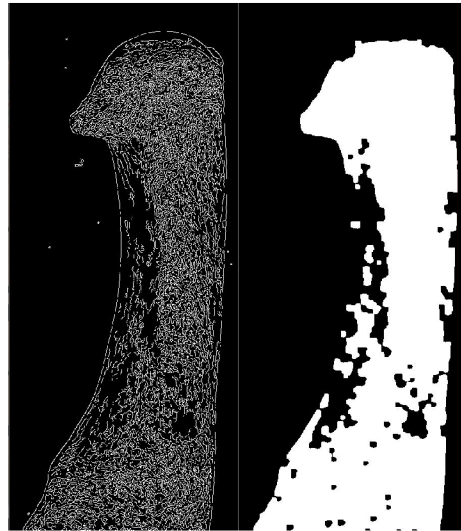

Figure 4.4: Mask of the bone after opening before filling contour.

Figure 4.5: Large kernel including background in the mask.

4.1.3 Conclusion of testing

Finally as it was not very necessary to test these phases, because the extraction of the contour was performed on the basis of the mask from the part of the detection and the calculation of the resulting width and length were only mathematical operations, it was time to test implemented functionality with database I have been provided from my supervisor.

After testing HSV method, the results were following:

- Number of files computed : 1884

- Number of errors in contour of the bone 149

- Number of errors in measurements based on imperfect contour: 24

Based on this result I can evaluate that using HSV method, the contour was faulty in 7.91% of cases and measurements were incorrect in 1.27% of cases. Most of the time the image was not correctly processed was when it contained plasticine. Faulty results are shown on picture 4.6.
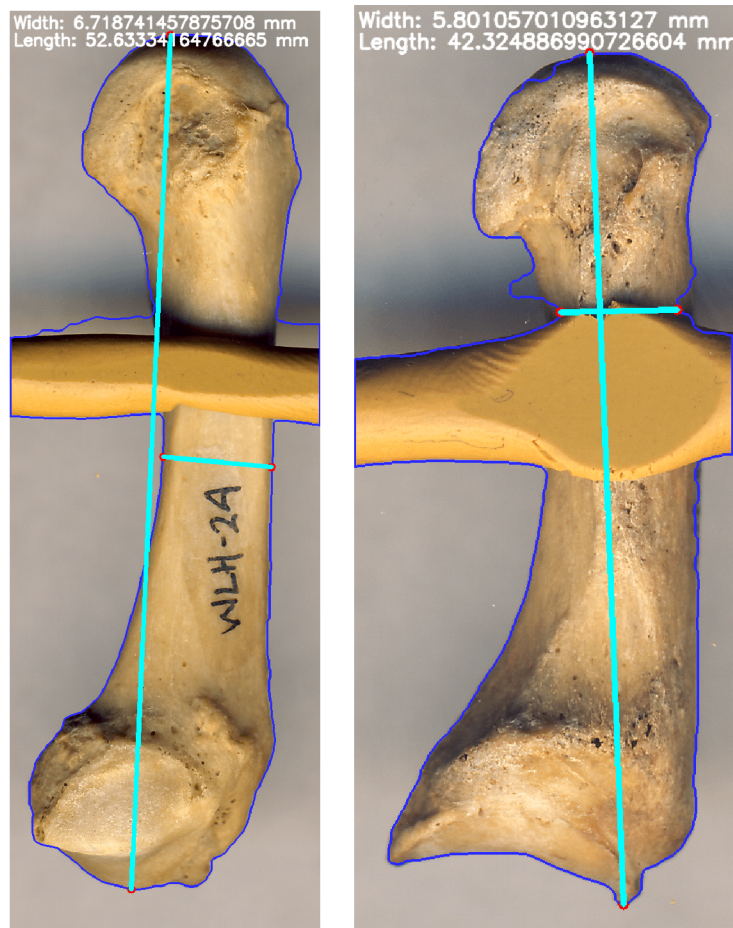
Figure 4.6: Inaccurately processed bones with HSV method.

Alternatively when testing canny edge detection, the outcome was following:

- Number of files computed : 1225

- Number of errors in contour of the bone 9

- Number of errors in measurements based on imperfect contour: 4

From these results I can assess that using canny edge detection method, the contour was faulty in 0.73% of case and measurements were inaccurate in 0.33% cases. Errors were mostly caused by incorrect detection of the metacarpal bones head part. Faulty results are shown on picture 4.7.
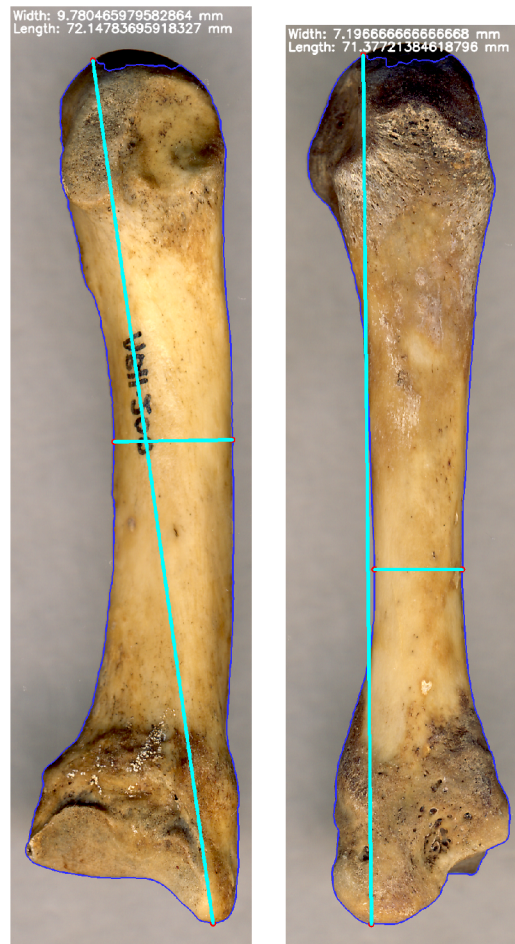
Figure 4.7: Inaccurately processed bones with canny edge detection method in the head part of the bone.

To summarize, we have discussed what methods were tested, how and why I decided to use certain methods and what problems occurred during the development of implementation. Subsequently, the results of the measurements were shown, which have been concluded with canny detection method being clearly better, however it is not able to correctly process metacarpal bone images that contain plasticine. HSV color threshold method also performed quite decently and additionally it is able to process images containing plasticine.

Chapter 5

Conclusion

To summarize, the goal of this thesis has been to find a method to detect and extract metacarpal bones from the scanned image and furthermore measure their height and width. I studied structure and variations of metacarpal bones to get a better understanding of how to process them. I read various studies about image processing methods and how to use and combine them to achieve detection of an object in the image, how to extract contour of an object, how to measure objects dimensions in the image and convert them from pixel measurement into metric units. Based on this knowledge I managed to come up with a solution that is based on either filtering the image with Gaussian filter, converting it to HSV color model or filtering image with median filter and using canny edge detection to create the mask of the metacarpal bone and correct that mask with morphological transformations. In the extraction process boundary is obtained by using threshold of black and white using the mask. Points of boundary are used to determine the height and width of the bone by calculating Euclidian distance between pixels and converting the result to the human metric. Lastly these measurements are outputted to CSV file. During the course of development I tested different methods and approaches to fine tune all algorithms as best as I could, so they produce promising result. I encountered many difficulties and taken steps to resolve them.

5.1 Possible extensions

During the process of creating this thesis it occurred to me that I could expand already existing solution for additional functionality. For instance I could be able to calculate perimeter of the bone base on contour pixels. It would consist of measuring length between neighbouring pixels and adding measurements together. Moreover I could improve solution to work with multiple background colors, because so far it works only with specific background color scheme. Solution could additionally be extended to measure bone based on selection of arbitrary pixels. This way you could measure any length not just width and height.

Bibliography

[1] *Automated bone age estimation* [online]. [cit. 2020-01-10]. Available at: https://bonexpert.com.

[2] *BMA: Bone Microarchitecture Analysis Add-On* [online]. [cit. 2021-01-10]. Available at: https://analyzedirect.com/analyze14/bma/.

[3] *Documentation — PyInstaller bundles Python applications* [online]. [cit. 2021-04-26]. Available at: https://www.pyinstaller.org/documentation.html.

[4] Metacarpal bones. [cit. 2021-04-24]. Available at: https://en.wikipedia.org/wiki/First_metacarpal_bone#/media/File: First_metacarpal_bone_(left_hand)_01_palmar_view_with_label.png.

[5] *Morphological transformations* [online]. [cit. 2021-01-10]. Available at: https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html.

[6] *Tkinter — Python interface to Tcl/Tk — Python 3.9.5 documentation* [online]. Python Software Foundation [cit. 2021-04-26]. Available at: https://docs.python.org/3/library/tkinter.html.

[7] AGOSTON, M. K. *Computer Graphics and Geometric Modelling.* 1st ed. Springer Science Business Media, 2005 [cit. 2021-04-25]. ISBN 9781852338183.

[8] ARIAS CASTRO, E. and DONOHO, D. L. Does median filtering truly preserve edges better than linear filtering? Institute of Mathematical Statistics. 2009, vol. 37, [cit. 2021-01-10]. ISSN 1172–1206. Available at: https://projecteuclid.org/euclid.aos/1239369019.

[9] CC–BY–NC–SA and GETREUER, P. A Survey of Gaussian Convolution Algorithms. [online]. Image Processing On Line. 2013, [cit. 2021-01-10]. ISSN 2105–1232. Available at: http://www.ipol.im/pub/art/2013/87/article.pdf.

[10] CHAVEZ, P. and BAUER, B. An automatic optimum kernel-size selection technique for edge enhancement. *Remote Sensing of Environment.* march 1982, vol. 12, p. 23–38, [cit. 2021-04-25]. DOI: 10.1016/0034-4257(82)90005-0.

[11] DAYA SAGAR, B. S. Computer processing of remotely sensed images: an introduction. 2nd Edition. *Computers Geosciences - COMPUT GEOSCI.* march 2001, vol. 27, [cit. 2021-04-25]. DOI: 10.1016/S0098-3004(00)00092-3.

[12] FRANCESCO BANTERLE, P. C. and SCOPIGNO, R. A Low-Memory, Straightforward and Fast Bilateral Filter Through Subsampling in Spatial Domain. The Eurographics

Association and Blackwell Publishing. 2012, vol. 31, [cit. 2021-01-10]. DOI:
10.1111/j.1467-8659.2011.02078.x. Available at:
http://vcg.isti.cnr.it/Publications/2012/BCCS12/j.1467-8659.2011.02078.x.pdf.

[13] Gray, H. and Carter, H. V. *Anatomy of human body*. 20th ed. Lea Febiger, 1918
[cit. 2021-01-10].

[14] K., A. M. . A. Canny Edge Detection. Institute of Mathematical Statistics. 2013,
[cit. 2021-05-01]. Available at: https://opencv-python-tutroals.readthedocs.io/en/
latest/py_tutorials/py_imgproc/py_canny/py_canny.html#theory.

[15] Oliphant, T. *Guide to NumPy*. January 2006 [cit. 2021-04-26].

[16] OpenCV. *Open Source Computer Vision Library*. 2015 [cit. 2021-04-26].

[17] Popov, V., Ostarek, M. and Tenison, C. Practices and pitfalls in inferring neural
representations. *NeuroImage*. march 2018, vol. 174, [cit. 2021-04-25]. DOI:
10.1016/j.neuroimage.2018.03.041.

[18] Shafranovich, Y. *Common Format and MIME Type for Comma-Separated Values
(CSV) Files* [Internet Requests for Comments]. RFC 4180. RFC Editor, october
2005 [cit. 2021-04-27]. Available at: https://www.rfc-editor.org/rfc/rfc4180.txt.

[19] Suzuki, S. and be, K. Topological structural analysis of digitized binary images by
border following. *Computer Vision, Graphics, and Image Processing*. vol. 30.

Appendices

Appendix A

Comparison between Gaussian, median and bilateral filter with followup methods

HSV color thresholding comparison between filters.

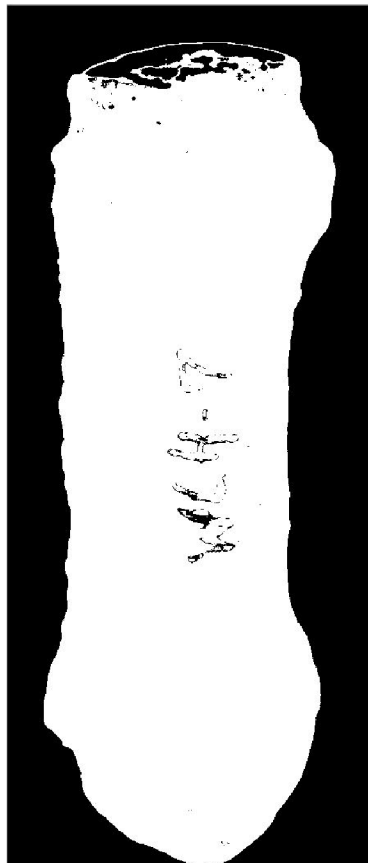Figure A.1: HSV thresholding with Gaussian blur.

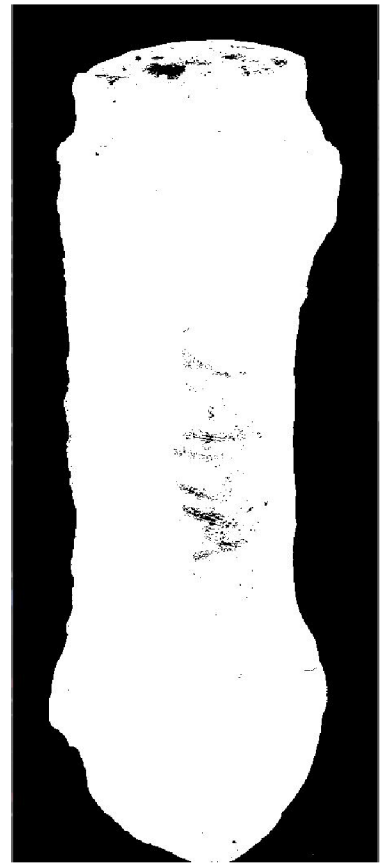Figure A.2: HSV thresholding with median filter.

Figure A.3: HSV thresholding with bilateral filter.

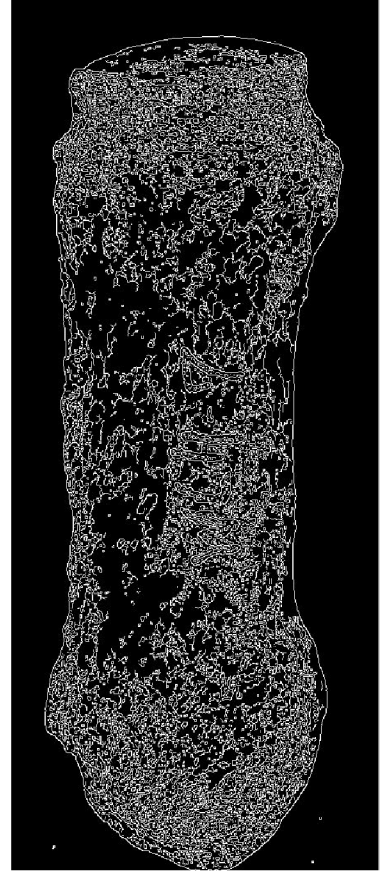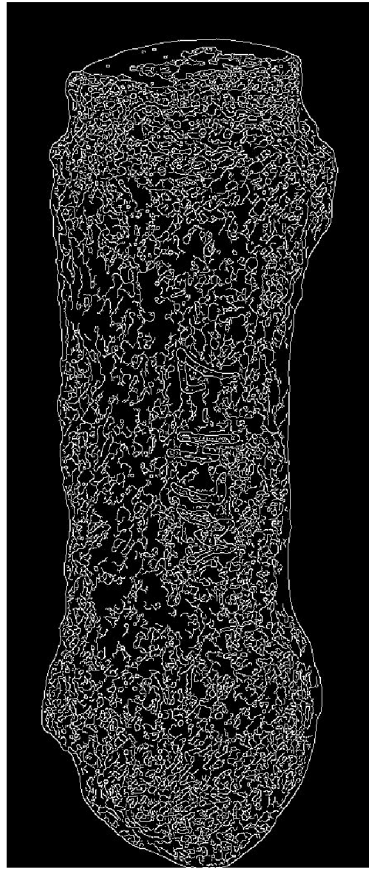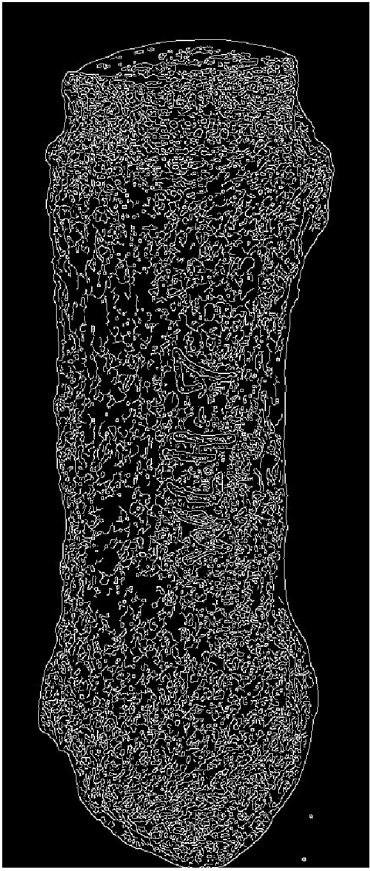Canny edge detection comparison between filters.

Figure A.4: Canny edge detection with Gaussian blur.

Figure A.5: Canny edge detection with median filter.

Figure A.6: Canny edge detection with bilateral filter.

Morphological transformation step comparison between filters and HSV color thresholding/canny edge detection.

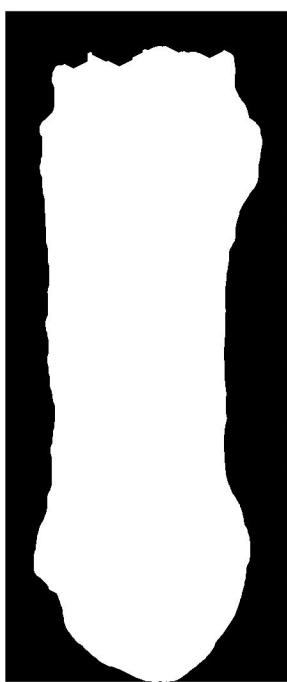Figure A.7: HSV color thresholding and Gaussian blur.

Figure A.8: HSV color thresholding and median filter.

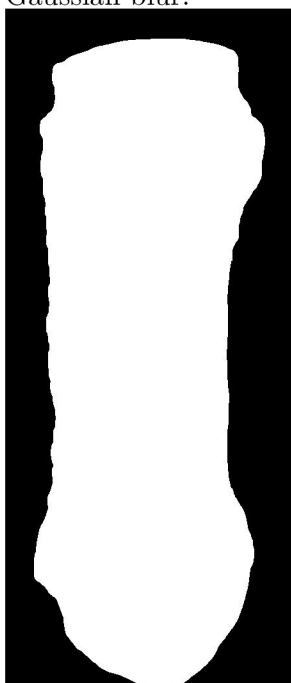Figure A.9: HSV color thresholding and Bilateral filter.
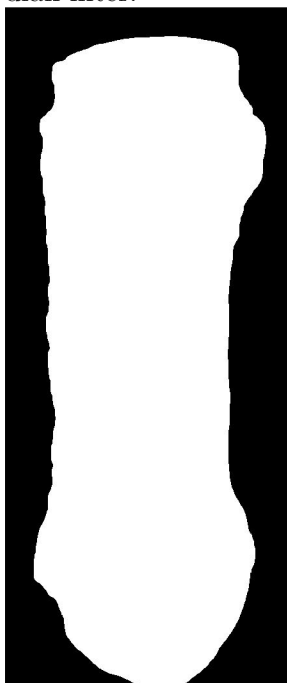

Figure A.10: Canny edge detection and Gaussian blur.

Figure A.11: Canny edge detection and median filter.

Figure A.12: Canny edge detection and bilateral filter.

Final result of detection comparison.

| Original image. | HSV thresholding with Gaussian blur. | HSV thresholding with median filter. | HSV thresholding with bilateral filter. |

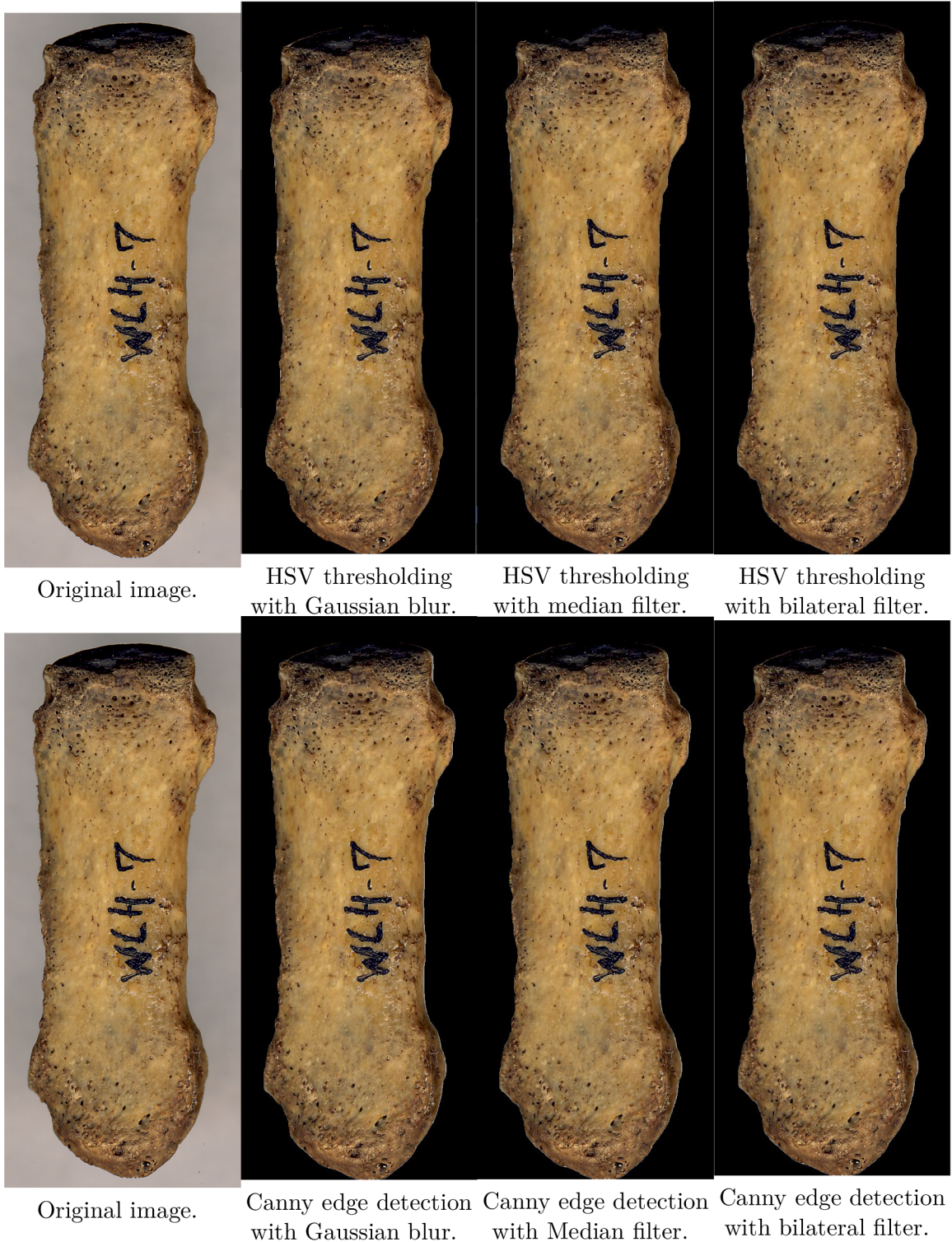| Original image. | Canny edge detection with Gaussian blur. | Canny edge detection with Median filter. | Canny edge detection with bilateral filter. |

Figure A.13: Detection results of HSV and Canny method.