BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

LASER LINE PROFILOMETRY

LASEROVÁ LINIOVÁ PROFILOMETRIE S VYUŽITÍM KAMERY

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **JAN KUGLER**
AUTOR PRÁCE

SUPERVISOR **Ing. PETR MALANÍK,**
VEDOUCÍ PRÁCE

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

157095

Ústav:	Ústav inteligentních systémů (UITS)
Student:	**Kugler Jan**
Program:	Informační technologie
Název:	**Laserová liniová profilometrie s využitím kamery**
Kategorie:	Zpracování obrazu
Akademický rok:	2023/24

Zadání:

1. Seznamte se s principy profilometrie, podrobněji se zaměřte na laserovou profilometrii s využitím kamery. Sestavte přehled existujících zařízení a jejich parametrů.
2. Navrhněte sestavu umožňující měřit profilometrická data v reálném čase s využitím liniového laseru a kamery. Určete teoretické možnosti této snímací sestavy v různých konfigurací systému.
3. Navrhněte a implementujte knihovnu umožňující zpracování obrazu z kamery a následný výpočet profilometrických dat snímaného povrchu.
4. Demonstrujte funkčnost knihovny v ukázkové aplikaci, která bude poskytovat živý náhled z kamery včetně vykresleného profilu snímané oblasti.
5. Proveďte snímání na různých typech povrchu (tvar, materiál, textura, atd.) a sestavte přehled dosažených výsledků.
6. Zhodnoťte dosažené výsledky a porovnejte je oproti teoretickým výpočtům a zařízením dostupným na trhu.

Literatura:

- PARKER, Jim R. *Algorithms for image processing and computer vision*. 2010, ISBN: 978-0-470-64385-3.
- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, 2007, ISBN-13: 978-0495082521.

Při obhajobě semestrální části projektu je požadováno:
Splnění bodů 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Malaník Petr, Ing.**
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1.11.2023
Termín pro odevzdání:	9.5.2024
Datum schválení:	6.11.2023

Abstract

A profilometer is a device used to measure the geometry of a surface. They vary significantly in size, precision and speed of measurement, and price. The aim of this thesis is to create a software library and end-user application for a simple laser-camera setup. A mathematical model of the system is developed and used to determine position of surface features in real space. The demonstration application allows the user to view the raw camera input, the processed image, as well as the resulting profile.

Abstrakt

Profilometr je zařízení sloužící k měření geometrie povrchu. Liší se ve velikosti, přesnosti a rychlosti měření, a ceně. Účelem této práce je vytvořit software knihovnu a uživatelskou aplikaci pro jednoduchý systém složený z kamery a laseru, jehož úkolem bude měřit opotřebení vnitřního povrchu tankových hlavní. Je vypracován matematický popis tohoto systému který knihovna využívá pro určení polohy měřeného povrchu v prostoru. Výsledná knihovna obsahuje nástroje pro identifikaci rysů povrchu. Demonstrační aplikace umožňuje zároveň sledovat snímaný povrch, zpracovaný obraz, i výsledný profil povrchu.

Keywords

profilometry, laser triangulation profilometry, surface profile measurement, surface texture, barrel wear, software library, computer vision, remote sensing

Klíčová slova

profilometrie, laserová triangulační profilometrie, měření profilu povrchu, textura povrchu, opotřebení hlavně, software knihovna

Reference

KUGLER, Jan. *Laser Line Profilometry.* Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Petr Malaník,

Laser Line Profilometry

Declaration

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Malaníka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Kugler
May 9, 2024

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview of profilometry

Profilometry is the measurement of geometry of surfaces - surface texture, and in some cases, the internal structure of translucent volumes. It is crucial across many areas of industry and scientific research. These range from material science, where it can be used to predict material behavior under stress, through quality control and maintenance of machine parts, to microbiology.

Surface texture encompasses several features: form, waviness, roughness, lay, and flaws. „Form“ describes the overall shape of the profile, „waviness“ the periodic deviations due to external vibrations or internal stresses during manufacture, and „roughness“ the finer texture typical of the material and its processing method. „Lay“ refers to the directionality and distribution of these textural features, critical for predicting how a surface will interact with its environment [1].

1.2 Purpose of the library

The primary purpose of this thesis is to produce a lightweight profilometry library that utilizes only rudimentary equipment - a camera and a laser line projector, rather than expensive specialized hardware.

A particular application of interest is the measurement and analysis of tank barrel wear, an area where precision in measurement can significantly influence operational safety and efficiency. Tank barrels, like all firearm barrels, undergo severe mechanical and thermal stresses during use -—stresses that include friction, high temperature and pressure fluctuations, and corrosive chemical reactions from the ignition of propellant. These stresses lead to the erosion of the barrel's bore, which can degrade weapon performance by reducing muzzle velocity, causing projectile misalignment, and decreasing the muzzle spin rate in rifled barrels [7]. If not adequately monitored and managed, such wear can ultimately lead to complete structural failure of the barrel [2]. The developed library aims to offer a practical solution for regular, cost-effective measurement of such wear, ultimately helping extend the operational life and reliability of these components.

1.3 Scope of the thesis

The principal statement of the problem is:

Given the characteristics of the camera and its relative position to the laser plane, identify features in the image and find their position in real space.

This breaks down as follows:

- Image processing
  - Apply edge detection or image segmentation
  - Extract the relevant features
- Position calculation
  - Correct camera distortions
  - Transform from position in screen space to real space

Chapter 2

Existing technologies

There exists a number of methods to perform the measurement of surface profile. The most general distinction is between optical and contact or pseudo-contact devices.

2.1 Atomic force microscopy

An atomic force microscope (AFM) consists of a cantilever probe, a laser diode, and a photodiode. The probe is dragged across, or oscillated above, the measured surface, bending the cantilever and changing the amount of light hitting the photodiode.

The advantages of this method are excellent vertical resolution (∼10 pm), and the ability to measure other material properties as well, such as elasticity or stiffness.

The disadvantages are limited horizontal range (∼100 μm), slow speed and the need for special sample preparation.[9]

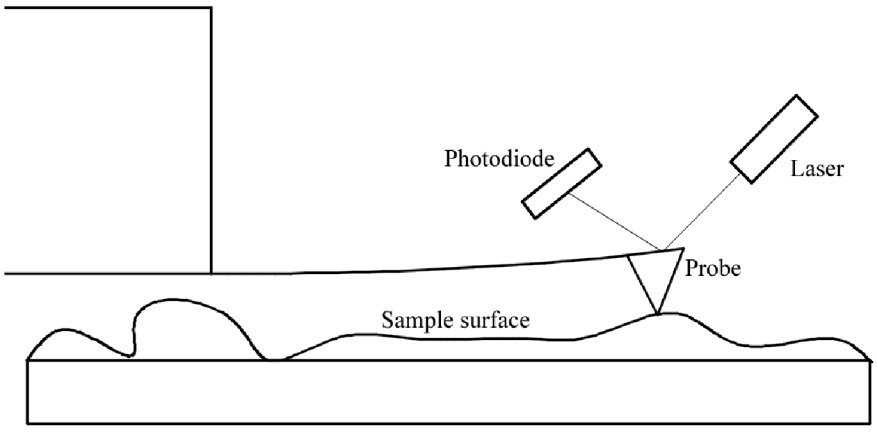

Figure 2.1: AFM diagram

2.2 Confocal microscopy

Confocal microscopy uses a confocal pinhole to block out all light not coming through the microscope's conjugate focal point. The light intensity in the resulting image is used to calculate the height map.[8]

2.3 Focus variation

Focus variation utilizes an optical system with narrow depth of field. The 3D model is compiled from images of the sample taken as the focal plane is shifted along the vertical axis.

The lateral resolution of this method is limited by the wavelength of visible light (∼400 nm). The depth resolution can reach as low as 10 nm. Both lateral and vertical range depend on objectives used, but range in unit mm. An advantage over other methods is the ability to measure very high slope angles (in excess of 80°).[5]

2.4 Interferometry

Interferometry methods work by utilizing beam splitters to split their light source into a reference beam and sampling beam. These are reflected off of a reference surface and the sample respectively, recombined, and directed into the sensor.

2.4.1 Phase shifting interferometry

PSI uses a monochromatic light source. The path length difference Δz of interfering beams of light is proportional to the phase difference $\Delta\phi$.

$$\Delta z = \frac{\lambda}{2\pi}\Delta\phi \tag{2.1}$$

The measured phase ϕ also determines the intensity I.

$$I(x,y) = I_{background} + 2I_{amplitude}cos(\phi(x,y) + \phi'(t)) \tag{2.2}$$

Where ϕ' is phase shift. If we consider four intensity values with phase shift offset by multiples of $\pi/2$, then

$$\phi(x,y) = arctan(\frac{I_4(x,y) - I_2(x,y)}{I_1(x,y) - I_3(x,y)}) \tag{2.3}$$

The depth resolution can reach 0.3 nm.[9]

2.4.2 White light interferometry

WLI uses the full visible spectrum to create a chromatic interference pattern.

Its advantages are good depth resolution (∼0.1 nm), high speed, and lateral range (unit mm). However, it is unable to measure high slope angles and materials with varying refractive index very well.[9]

2.5 Laser triangulation

Laser triangulation uses a laser emitter and a camera in set relative position. The laser projects a point or a line onto the surface. The surface profile is calculated based on the projection's position in the camera's field of view. The setup further examined in this paper falls into this category.

2.6 Stylus profiling

Contact profilometry utilizes a stylus moving across the surface to directly sample its vertical displacement. Both vertical and horizontal resolution are determined by the dimensions of the stylus tip and the force applied to it. The ISO standard defines these to be a cone with 20 μm tip radius, 60° or 90° tip angle, and 750 μN measuring force [6]. The tip material is typically diamond due to its high hardness and low coefficient of friction [4].

The advantages of this method are high vertical resolution ($\sim$0.1 nm) and long scan length (several cm). It is also not affected by the optical properties of the surface.

Like the AFM, this method measures only a single point at a time, so scanning areas is slow.[9]

Chapter 3

Methodology

3.1 System geometry

The physical system is composed of the laser plane P_l and the camera C. The relevant geometrical variables are the camera elevation c, and the view angle α, i.e. the distance of the camera and the laser plane, and the angle between the plane and the camera's center vector $\overrightarrow{v_c}$.

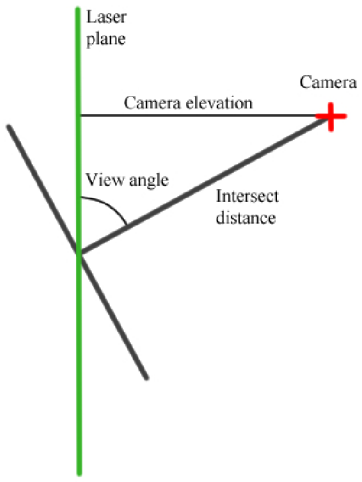

Figure 3.1: System geometry

The relevant properties of the camera are its pixel count cnt_x and cnt_y and the respective field of view angles Φ_x and Φ_y.

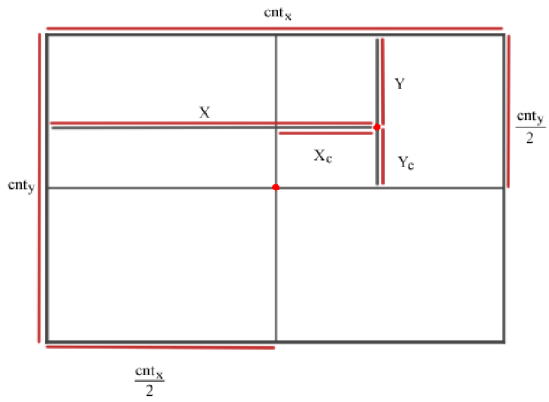

Figure 3.2: Screen space variables

It is further assumed the camera and P_l are „aligned“ – meaning that given any two pixels $(cnt_x/2, y_1)$ and $(cnt_x/2, y_2)$, the triangle formed by their projections into P_l and C is perpendicular to P_l. The exact position of the laser beam emitter E is not relevant. It is assumed that for any point $p \in P_l$ within the camera's FOV the beam incident on a perpendicular surface is visible. In other words $\overrightarrow{Ep} \cdot \overrightarrow{P_l p} > 0$. Effects of laser decoherence are not taken into account, as it is assumed the edge of the beam remains flat across the distances considered.

The pixel plane P_p is defined to be perpedicular to $\overrightarrow{v_c}$, positioned at its intersection with P_l. The distance d from the camera to this point is given by

$$d = \frac{c}{\sin(\alpha)}$$

This intersection point is taken to be the origin of both planes. From the alignment assumption it follows that the intersection line of the two planes is their shared x axis.

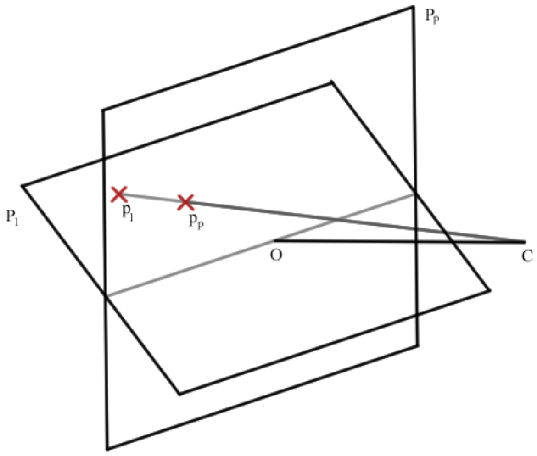

Figure 3.3: Map visual

The angles ϕ_x and ϕ_y are defined as:

$$\tan(\frac{\phi_x}{2}) = \frac{X}{cnt_x}\tan(\frac{\Phi_x}{2})$$
$$\tan(\frac{\phi_y}{2}) = \frac{Y}{cnt_y}\tan(\frac{\Phi_y}{2})$$

Any line passing through C – with the exception of those parallel to P_l – thus uniquely maps a point from P_l to P_p:

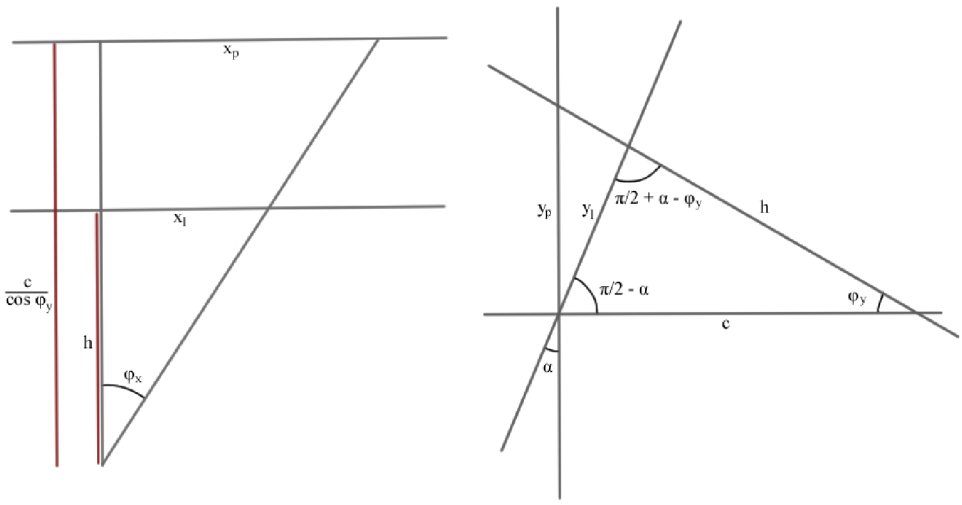

Figure 3.4: x_p and y_p are sides of a rectangle in P_p, one of its vertices being the origin. It maps onto a trapezoid in P_l, which shares with the rectangle the origin x_p side.

$$y_l = d\frac{\sin(\phi_y)}{\cos(\alpha - \phi_y)}$$

$$x_l = d \tan(\frac{\phi_x}{2}) \frac{\cos(\alpha)\cos(\phi_y)}{\cos(\alpha - \phi_y)}$$

Chapter 4

Implementation

4.1 Library architecture

The library is written in C++ and utilizes opencv. It contains global library variables, the setter and getter functions thereof, and library functions proper.

4.1.1 Types

The only new types the library defines are enums. The purpose of the enums is to improve readability and maintainability of the code.

`processingEnum` is used for methods of edge detection and image segmentation.

`extractionEnum` is used for methods of feature extraction from processed images.

`rotationEnum` holds values for no, counter-clockwise, 180 degree, and clockwise rotations. Modular addition and subtraction are defined for this.

`dirEnum` holds values for 8 cardinal directions, starting from east counter-clockwise.

`channelEnum` holds values for the three RGB channels, all channels, and a no channel value.

4.1.2 Global variables

The global variables are a thing that exists.

`int gaussKernelSize` is the size of the blur kernel used throughout various library functions. The default value is 3.

`channelEnum isolateTargetChannel` is used in pre-processing to communicate that only one channel is to be considered. No channel is isolated by default.

`channelEnum amplifyTargetChannel` is used in pre-processing to amplify the color values of a given channel. No channel is amplified by default.

`int regionGrowingThreshold` is the intensity tolerance of the `region_growing` algorithm.

`cv::Point seedPoint` is the starting point of the `region_growing` algorithm and the offset threshold extraction method. Its default value is (0, 0).

`int cannyLowThresholdMax` is the maximal allowed value of `cannyLowThreshold`. Attempting to set this to 0 or less sets it to 1. Should setting this put `cannyLowThreshold` outside acceptable range, it will be set to the new value of this. The default value is 100.

`int cannyLowThreshold` is the low threshold of Canny edge detection hysteresis. Attempting to set too low or too high a value sets this to 1 or `cannyLowThresholdMax`, respectively. The default value is half of `cannyLowThresholdMax`.

`double cannyThresholdRatio` – rather than explicitly define the high hysteresis threshold, it is calculated by multiplying the low threshold by this ratio. The default value is 3, as per Canny's recommendation [3].

`int cannyKernelSize` is the size of the Sobel kernel used internally by `cv::Canny`. opencv only allows values of 3, 5, or 7, attempting to set this to other values is a no-op. The default value is 3.

`double viewAngle`, `double cameraElevation`, `double FOVx`, `double FOVy`, `int cntx`, and `int cnty` hold the values described in 3.1. Specifically α, c, Φ_x, Φ_y, cnt_x, and cnt_y, respectively.

When setting `viewAngle`, a helper variable `double cosVA` is calculated. `viewAngle` is restricted to values between 0 and π.

When setting `cameraElevation`, a helper variable `intersectDist` is calculated.

Likewise, for the two FOV variables there is `FOVxHalfTan` and `FOVyHalfTan`. These angles are also restricted to the above interval.

4.1.3 Algorithm implementation

Many of the library's functions take a `source` and `target` argument. Unless stated otherwise, `source` is unchanged by the function and `target` is overwritten. The gestalt of the library's functionality is implemented in the `process_image` function.

`int process_image(cv::Mat &source, cv::Mat &target, cv::Rect roi, processingEnum pe)` handles region of interest application, color pre-processing, and first stage processing, applied in this order. This function seeks to aggregate the whole processing pipeline while providing a unified interface for all processing methods and to handle the manipulation of global library variables. The branching is handled by a `pe` switch.

- `pe_none` causes the function to only apply region of interest cropping and channel pre-processing. This is the default value.
- `pe_gray` converts the image to grayscale using `cv::cvtColor`.
- `pe_canny` converts the image to grayscale, applies Gaussian blur using `cv::blur`, and Canny edge detection via `cv::Canny`.
- `pe_sobel` converts the image to grayscale, applies blur, and `cv::Sobel`.
- `pe_region` converts the image to grayscale and applies the `region_growing` algorithm.

Pre-processing

Two methods of pre-processing are implemented: channel amplification and channel isolation. They are not mutually exclusive. As channel amplification utilizes values of all three channels, it is executed first.

`void quad_dif_amp(cv::Mat &source, cv::Mat &target, channelEnum ce)` `target`'s `ce` channel is amplified by the square of its difference from the average of the other

two channels. For example for `ce = ce_green` the target pixel's green channel is calculated from the source pixel like so:

$$t[1] = \begin{cases} s[1] & \text{if } s[1] \leq \frac{s[0]+s[2]}{2} \\ s[1] + (s[1] - \frac{s[0]+s[2]}{2})^2 & \text{otherwise} \end{cases}$$

Overflow is truncated to 255. The other two channels are copied from `source`. Calling with `ce_none` or `ce_all` is equivalent to `target = source;`.

`void isolate_channel(cv::Mat &source, cv::Mat &target, channelEnum ce)` `target`'s `ce` channel is copied from `source`, the other two are set to zero. Calling with `ce_none` or `ce_all` is equivalent to `target = source;`.

Edge detection and image segmentation

`void edge_link(cv::Mat &source, dirEnum edge_normal)` is meant to fill in larger gaps in edges that weren't picked up by `cv::Canny` by using a priority matrix to check a directed neighborhood of a pixel for continuation. It is not fully implemented.

`void region_growing(cv::Mat &source, cv::Mat &target, cv::Point seed_point)` starts with a seed point, which is assumed to be a part of the region. Every step of the algorithm iterates over the region's boundary and checks the Moore neighborhood of each point. If the intensity of any of the points within the neighborhood is within `regionGrowingThreshold` of the boundary point, it is added to the region. The algorithm terminates once the region stops being updated.

`std::vector<cv::Point> moore_neighborhood(cv::Point p, cv::Size s)` returns a vector of 8 points surrounding `p` if `s` is not supplied. If it is, only the points which are valid within the image are returned.

`std::vector<cv::Point> moore_boundary(cv::Mat &image)` implements the Moore boundary tracing algorithm. First point of the region is found by scanning the image. Each step turns left or right from the direction of the previous based on whether the current point is in the region. The algorithm terminates when the first point visited is visited again from the same direction.

Feature extraction

The edge extraction functions are rudimentary, relying on the work of previous processing stages. All of them scan the image line by line horizontally or vertically as indicated by the `dir` argument. They expect as input CV_8UC1 type images. Note that the input images may be cropped by the region of interest, so the output point vector needs to have offsets added before being passed to the plane mapper.

`std::vector<cv::Point> edge_extract_strongest(cv::Mat &source, dirEnum dir)` finds the highest intensity pixel in each line. If multiple such pixels exist, only the first encountered is added to the vector.

`std::vector<cv::Point> edge_extract_thr_strongest(cv::Mat &source, dirEnum dir, int thr)` likewise finds the highest intensity pixels. Pixels are only added to the vector if their intensity is greater than the threshold.

`std::vector<cv::Point> edge_extract_thr_first(cv::Mat &source, dirEnum dir, int thr)` finds the first pixel in each line that clears the threshold. After it is found, the rest of the line is skipped over.

Plane mapping

Mapping from the pixel plane to the laser plane is fully implemented using the `mapSS2RS` function.

`void add_roi_offset(std::vector<cv::Point> &edge, cv::Rect roi)` adds the roi's coordinates to each point in `edge`, converting them back to coordinates in the full image.

`std::vector<cv::Vec2d> mapSS2RS(std::vector<cv::Point> &edge)` expects as input the vectors returned by `edge_extract` functions. It calculates coordinates of the edge pixels' images in the laser plane as per 3.1.

Utility and other functions

`void enum_rotate(cv::Mat &src, rotationEnum re)` uses `cv::flip` and `cv::transpose` to efficiently rotate images. This function changes the source image.

`bool point_in_image(cv::Point p, cv::Mat &image)` compares `p` to dimensions of `image` and returns `true` if `p` is a valid pixel.

4.2 Demonstration program

The demo program is likewise written in C++, using the Qt framework. It can be compiled using the `first_build.sh` script, or manually by creating a `build` folder and executing `cmake .. && make` inside it.

It displays the raw image input, intermediate processed image, and the resulting profile. It allows input in the form of existing image files, or camera feed. The images are loaded from the `images` folder.

The sliders around the raw image display allow the user to select a region of interest. This cuts down processing time and resource usage by ignoring irrelevant segments of the image. The region of interest can be rendered in the raw image for ease of use.

Other controls allow the user to select the processing methods and adjust global library variables. `seedPoint` can be selected by clicking anywhere in the processed image.

The bulk of the demo program's code are slots connected to the various ui widgets. These facilitate communication with the library, graphical feedback, and value conversion where necessary.

A notable exception to this are the image rendering functions.

`void putFrameIntoLabel(QLabel* label, const cv::Mat frame, bool keepAspectRatio)` inserts the image into the label widget. The raw image is stretched to fit the label to provide full view, while the processed image label preserves original image's aspect ratio to maintain fidelity. The profile image label likewise maintains the aspect ratio, but in its case this isn't very relevant.

`void renderLabels()` blends the raw image with the region of interest overlay and calls `putFrameIntoLabel` for it. After this it calls the library's `process_image` function and likewise calls `putFrameIntoLabel`.

`void renderProfile()` manages calls to the edge extractors, roi offset correction, and the plane mapping function. When the real point vector is ready it calls `graphRender` and `putFrameIntoLabel`.

`void graphRender(cv::Mat &target, std::vector<cv::Vec2d> &real_edge)` builds a graph image from `real_edge`. It finds its value range in either dimension and,

using this, iterates over the points, rescaling them to fit the image, and connecting them using the Bresenham line drawing algorithm.

Generally speaking, `renderLabels()` is called whenever a processing parameter is adjusted. Likewise, `renderProfile()` is called on camera or system geometry parameter updates. Both are called on new camera frame.

The retrieval of image from the camera is handled by a separate thread, subclassed from `QThread`. The default camera descriptor used is 0, but the user has the ability to select otherwise. The thread cannot be directly called with a descriptor argument, so get/set functions are defined for it. The thread also has get functions defined to retrieve the image width and height directly from the `cv::VideoCapture` object, rather than from the frames it provides.

The `QLabel` class by default does not support the capture of mouse clicks, so this is accomplished by subclassing. `ClickableLabel` can emit this signal, along with the coordinates of the event in the label. Mapping of these label coordinates to image coordinates is handled by the `onProcessedClick` slot function.

Chapter 5

Results and discussion

5.1 Functionality and performance

The library in its current state fulfills the bare minimum necessary for its intended application. It is able to extract the desired features reliably only from a specific type of images. To remedy this alternative means of pre-processing and image segmentation will have to be implemented. Currently the most reliable processing method is region growing. However, it is poorly optimized and therefore not usable for live video processing.

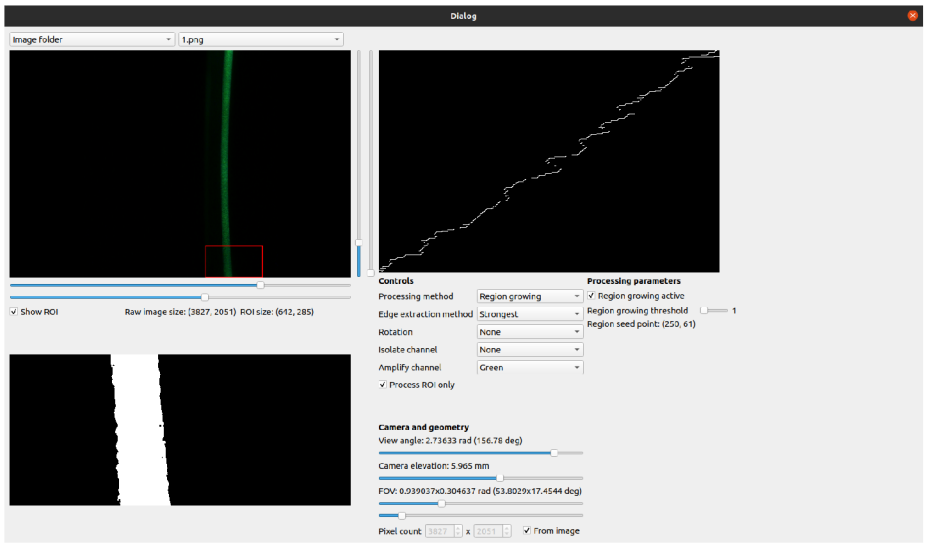

Figure 5.1: Region growing output

5.2 Future developments

As stated above, the library is lacking in both functionality and performance. I intend to further develop the library to amend these shortcomings and expand its scope further, for example using super-resolution techniques to improve the minimal detail.

Bibliography

[1] *JIS B 0601: Geometrical Product Specifications (GPS) – Surface texture: Profile method – Terms, definitions and surface texture parameters.* Japanese Standards Association, 2013.

[2] Balla, J.; Proch'azka, S. and Duong, V. Y. Influence of Barrel Wear and Thermal Deformation on Projectile Ramming Process. In:. 2012.

[3] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986, PAMI-8, no. 6, p. 679–698.

[4] Dagnal, H. *Exploring Surface Texture.* 1st ed. Taylor Hobson Publishing, 1996. ISBN 9780901920072.

[5] Danzl, R.; Helmli, F. and Scherer, S. Focus Variation – a Robust Technology for High Resolution Optical 3D Surface Metrology. *Strojniški vestnik - Journal of Mechanical Engineering*, 2011, vol. 57, no. 3, p. 245–256. ISSN 0039-2480. Available at: https://www.sv-jme.eu/article/focus-variation-a-robust-technology-for-high-resolution-optical-3d-surface-metrology/.

[6] ISO. *Geometrical Product Specifications (GPS) - Surface texture: Profile method - Nominal characteristics of contact (stylus) instruments.* 1996. Available at: https://standards.iteh.ai/catalog/standards/sist/70f20fe5-318d-41eb-ac49-33a6173bc60f/iso-3274-1996.

[7] Li, X.; Mu, L.; Zang, Y. and Qin, Q. Study on performance degradation and failure analysis of large caliber gun barrel. *Defence Technology*, 2019, vol. 16.

[8] Novak, M. *Laser and Coherence Scanning 3D Microscope Capabilities* online. 2014. Available at: https://www.youtube.com/watch?v=CPCQuwPcWJo&list=PLxgv_31xqEV5eMI-KXvDvOQk85vk58GR4&index=6. [cit. 2023-01-21].

[9] Zubaida, A. A. *An Overview of Surface Roughness Measurements: Choice of Technique and Analysis* online. 2020. Available at: https://www.bruker.com/en/news-and-events/webinars/2020/an-overview-of-surface-roughness-measurements-choice-of-technique-and-analysis.html. [cit. 2023-01-21].