UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Rasmus Saame

Design and Development of Materials for the Course "Digital Image Processing"

Bachelor's Thesis (9 ECTS)

Supervisors: Janno Jõgeva, MSc
Marilin Moor, MSc

Tartu 2025

Design and Development of Materials for the Course "Digital Image Processing"

Abstract:

Digital image processing is an important field that is used, for example, in medicine and in remote sensing. The goal of this thesis is to develop and improve teaching materials for the Digital Image Processing course at the University of Tartu that was not running after the academic year 2020/21. The course is intended to be part of a mandatory module in the Master's level Robotics and Computer Engineering curriculum at the Institute of Technology. Lab manuals focus on implementing and applying classical image processing algorithms. The manuals have been gradually improved based on feedback and comments received since the course's first academic year (2022/23). Materials for 8 labs with theoretical background, tasks and assisting visualisations were created for this thesis.

Keywords:
Digital image processing, pixel level operations, blurring, feature extraction, optical flow

CERCS: T111 Imaging, image processing. P175 Informatics, systems theory. S270 Pedagogy and didactics.

Kursuse "Digitaalne pilditöötlus" materjalide disain ja arendus

Lühikokkuvõte:

Digitaalne pilditöötlus on oluline valdkond, mida kasutatakse näiteks meditsiinis ning kaugseires. Lõputöö eesmärk on luua materjalid Tartu Ülikooli digitaalse pilditöötluse kursuse jaoks ning neid täiendada. See kursus kuulub Tartu Ülikooli Tehnoloogiainstituudi Arvutitehnika ja robootika magistriõppe kohustuslike kursuste hulka, kuid 2020/21 õppeaastast seda enam ei õpetatud. Loodavad materjalid keskenduvad klassikalise pilditöötluse algoritmide implementeerimisele ja nende rakendamisele. Materjalidesse tehti pärast esimest õppeaastat (2022/23) muudatusi varasema tagasiside ning tähelepanekute põhjal. Lõputöö käigus loodi 8 praktikumi juhendit teoreetilise materjali, ülesannete ning visualisatsioonidega.

Võtmesõnad:
Digitaalne pilditöötlus, pikslitasemel operatsioonid, udustamine, tunnuste eraldamine, optiline voog

CERCS: T111 Pilditehnika. P175 Informaatika, süsteemiteooria. S270 Pedagoogika ja didaktika.

Contents

1 Introduction

Digital image processing is an important field due to its use in many sectors. For example, in medical image processing it is used to enhance the quality of the images taken by computerized tomography scan (CT scan) and images taken from satellites can be analysed to monitor environmental changes [8]. There are multiple courses that focus on image processing in University of Tartu that cover different aspects. The Robotics and Computer Engineering (RCE) curriculum at the Institute of Technology also has an image processing course as one of the mandatory courses in the "Basic module" [35]. The course is also part of the Applied Measurement Science curriculum. This course in it's original form was last read in the year 2020/21, but it still remains in the curriculum. Starting from the academic year 2022/23 the course with materials from this thesis has been running. The new course shifted the focus towards image algorithms and their implementation and the environment from MATLAB to Pyhton. Thus, none of the previous materials could be reused.

The goal of this thesis is to create lab materials for the Digital Image Processing course (LOTI.05.037) at the University of Tartu that is mainly aimed towards RCE curriculum students. In addition to that, one of the goals of the thesis was to improve the lab manuals during the next years. The lab manuals have been developed for 3 years. During the 2022/23 academic year the materials were created from scratch by the author and over the next years the materials were improved. There are other linked activities and organisational notes that are related to the course and discussed in the thesis, but are not the sole contribution of the author - a clear distinction will be made in the following chapters.

An overview of other image processing courses at the University of Tartu and other European universities is first provided. This is followed by a description of the procedure used for developing the lab manuals, as well as the overall structure of the course. Next, the created lab manuals are described in detail. Finally, the received feedback is discussed, along with the changes made based on this feedback.

2 Background

The term "classical" image processing is used in text book "Digital Image Processing" by Rafael C. Gonzalez and Richard E. Woods [11] to refer to image processing methods not using machine learning and deep learning. The term is used in the thesis the same way. For instance, thresholding an image using a single pivot value is one example of "classical" image processing algorithm and classifying objects using a convolutional neural network is an example of a topic that is not considered to be part of "classical" image processing.

2.1 Other image processing courses

Previously, several image processing related courses have been created at various universities. This section covers four courses in the University of Tartu and two other courses from other European universities. This is done to give an example of different types of courses that focus on digital image processing.

2.1.1 "Multimedia"

The course "Multimedia", MTAT.03.132, offered by the Institute of Computer Science at the University of Tartu, is led by Sven Aller. This course focuses on the different aspects of multimedia, including overview of image processing techniques, file formats and vector graphics. This course does not focus on programming but more on the concepts and applications of the algorithms in popular pieces of software. All course materials and gradable activities are hosted entirely on the Moodle platform. The course materials are provided on the Moodle platform. The last time the course ran was in the spring semester of the academic year 2024/25. The volume of this course is 3 ECTS [34].

2.1.2 "Computational Imaging"

The course "Computational Imaging", LTAT.02.025, provided at the Institute of Computer Science at the University of Tartu, is led by Kallol Roy. This course focuses on human visual perception. The course covers camera basics, convolutional, recurrent and transformer neural networks and biomedical imaging. The materials are provided on the Institute of Computer Science Courses platform. Students gather points from 4 practical homeworks, a project and a final exam. The last time the course ran was in the spring semester of the academic year 2024/25. The volume of the course is 6 ECTS [30].

2.1.3 "Deep Learning for Computer Vision"

Led by Dmytro Fishman, the course "Deep Learning for Computer Vision", LTAT.02.028, at the Institute of Computer Science at the University of Tartu focuses on neural networks used in image processing. Among other topics the course covers convolutional neural networks, object detection and segmentation, vision transformers and image generation. The homeworks are written using Jupyter notebooks. Students gather points from homeworks and a project. The last time the course ran was in the spring semester of the academic year 2024/25. The volume of the course is 6 ECTS [31].

2.1.4 "Introduction to Polarization Imaging"

"Introduction to Polarization Imaging", LTFY.00.003, is a course at the University of Tartu's Institute of Physics led by Vipin Tiwari, that focuses on polarization of light. The course covers polarization cameras, generation and detection of different states of polarization and basic polarization imaging techniques. Instead of addressing most of the topics in image processing field this course focuses on a small portion of the field and explores it extensively. Students gather points from lab trainings, homework assignments, seminar presentations, a project and a final exam. The last time the course ran was in the autumn semester of the academic year 2024/25. The volume of this course is 3 ECTS [33].

2.1.5 "Advanced Robotized Systems"

The course "Advanced Robotized Systems" at the University of La Laguna focuses on "application of computer vision techniques in the field of robotics". J. Sigut et al. created a mobile application for the "Advanced Robotics Systems" course so that students could more easily get acquainted with the OpenCV framework and its capabilities [23].

The course focuses on computer vision, image filtration, feature extraction and morphological operations. The knowledge acquired during the course is checked with an exam at the end of the course. The exam focuses on the use of OpenCV. The course uses the programming language Octave. The volume of the course is 6 ECTS [23].

2.1.6 "Computer Vision"

The course "Computer Vision" at the University of Craiova in the Mechatronics and Robotics Department focuses on image acquisition, image segmentation and classification among other things. D. Cojocaru et al. updated the course syllabus in 2022. The course focuses on computer vision and its volume is 5 ECTS. It is aimed for the Robotics Bachelor's students [6].

2.2 Problem

At the time this thesis started, no course focusing on algorithmic thinking in classical image processing was offered at the University of Tartu. Algorithmic thinking is useful and to improve it one good option is to implement algorithms from scratch [5]. In classical image processing there are many complex algorithms published that use many rules to describe how the algorithm should behave. Algorithmic thinking helps programmers to more easily understand these rule sets and makes them understand the important aspects and parameters of these algorithms when using these. The presence of a course that focuses on the classical side of image processing is important for developing programming skills in higher education.

In the article "Automated Classification of Skin Lesions: From Pixels to Practice" the authors found "that the algorithm appeared more likely to interpret images with rulers as malignant" [18]. They discussed that since in the input data set the images with malignant lesions had more rulers, then the model learned among other things that the presence of a ruler indicates malignancy. Thus, it is important to understand how the image processing pipeline works and what features is it looking for. This is easier for classical algorithms since it is often hard to understand exactly what is going on in the hidden layers of neural networks [7].

There was a Digital Image Processing course previously led by Morteza Daneshmand in University of Tartu but this course is not running any more. The course ran until the academic year

of 2020/21. The course was aimed towards the Master level students of the RCE curriculum. The course for which the materials in this thesis were developed replaces the course previously taught by Daneshmand. Daneshmand's course covered similar selection of topics, such as image acquisition, noise in images, edge detection, image classification and object tracking. The new course for which the materials discussed in this thesis were developed, focuses more on the implementation details of the algorithms. This was done to direct the focus of the course more towards algorithmic thinking [32].

The lack of the Digital Image Processing course at the Institute of Technology at the University of Tartu affected the students of the RCE curriculum because the course that they have to take according to the curriculum was not available and needed to be replaced with another course. The content given by the course would be helpful in situations where performance is really important. The libraries might not be modular enough nor available for the required platform. In the newly created course, the students focus on implementing well-known classical image processing algorithms in Python. This approach not only deepens their understanding but also makes it easier to later implement these algorithms in other programming languages. Implementing algorithms for a specific use case allows for additional assumptions to be made, leading to a more specialized and efficient program. Through this process, they become more familiar with the digital image processing toolkit, which will help them in future projects involving image processing.

3 Development of lab manuals

For each lab, students receive some starter code from instructors to base their work on. This is done so that the students do not have to write too much code that is not related to image processing. The base code that is given to the students is stored in the instructors' Git repository that is hosted in Bitbucket. All of the students also have their personal repository hosted in Bitbucket.

Both the instructors' repository and students repositories make use of the Large File Storage (LFS) extension for Git. This is due to the fact that Git is designed to work with source files, but in this course we are giving multiple binary files to students as well with the source code, mainly different input images. By using LFS extension the size of the Git repository does not grow so fast and history of the repository does not get cluttered. Creating a local clone of the repository takes less time, since Git does not have to load all of the historical binary files, but only the most recent version [3].

To be sure that all of the tasks are solvable with the described methods, instructors also store all of the example solutions to the tasks in the instructors' repository. To ensure that the base code and solutions are up to date with each other, instructors only keep the solutions of the tasks augmented with specific tags that give the ability to generate the base code from the solutions. This is done by inserting certain code comments into the solutions that represent parts of the code that should not be present in the base code. For generating the base code, there is a Python script that goes through the Python source code file and removes the parts marked by code comments.

The base code generator made development more efficient since during testing of each of the tasks the solutions for the task would be written. After confirming that a task is suitable for the course, all that remained was to include necessary code comments and base code could be generated.

Lab manuals were written using the AsciiDoctor compiler for the AsciiDoc language [2] [9].

Each lab manual development procedure looked as follows:

Algorithm 1: Lab manual development

```
1  year = 2023;
2  while year ≤ 2025 do
3      lab = 1;
4      while lab ≤ 8 do
5          First implementation of the manual and suggestion of the tasks;
6          Feedback from the instructors;
7          Making appropriate changes;
8          Feedback or issues from the students;
9          Making appropriate changes;
10         lab++;
11     end
12     year++;
13 end
```

In the first year of the course the development of the lab manuals required more time because all of the materials were created from scratch. In the following years the development required less time since most of the tasks were reused from the previous year with some exceptions. Most of the effort went into fixing the issues that became clear in the previous year. Issue tracking for this was mostly done via a Trello Kanban board. In the Kanban board instructors stored the issues that came up during testing the manuals or from students during the semester. These issues were fixed before the next academic year.

For developing the lab manuals a combination of development software was used so that the process would be more smooth. To generate the HTML file from the AsciiDoc source file, the file needed to be compiled. Instead of manually compiling the AsciiDoc source file after each change, nodemon was watching the changes to the source file. After each update to the file, nodemon triggered the Asciidoctor compiler to produce a new version of the lab manual. After this Node Package Manager's (NPM) live-server package detected the change in the HTML file and automatically reloaded the webpage in the browser. This automation helped to write manuals little bit faster and avoided the possible moments of confusion when changing the source file but not seeing the changes in the output file due to either not recompiling or not refreshing the page.

4 Course overview

In the 2022/23 and 2023/24 academic year there were 3 instructors in total for the course, 2 of these instructors instructed the practical sessions. In 2024/25 academic year due to larger amount of enrolled students the total number of instructors in the course was 4, 3 of these instructors instructed the practical sessions.

Figure 1 has the timing diagram of academic years 2022/23 and 2023/24. The course consists of 8 different labs. Students have approximately 2 weeks to complete each lab. Each row shows one of the activities taking place during semester. All of the seminars and lectures are grouped into single rows to improve readability. In the 2024/25 academic year instructors of the course decided to increase the frequency of the consultations starting from week 32 to take place every week. There is a possibility to complete the lab later as well but in that case the students will only get 50% of the points intended for that lab. This constraint was relaxed in the academic year of 2024/25, since then the students get 75% of the points if they finish the lab during the next practical session after the deadline and 50% of the points when they finish even later.

All labs from 1 to 7 have to be completed by week 38 at the latest. In week 39 there is only lab 8 which is instructor-led as opposed to the previous self-paced labs.

The expected time division during the academic years 2022/23 and 2023/24 was as follows:

1. Lab practicals - 64 hours;
2. Course project - 35 hours.
3. Taking part in the seminars - 20 hours;
4. Preparing for seminar presentations - 18 hours;
5. Midterm preparations and taking - 11 hours;
6. Taking part in the lectures - 8 hours;

The total is 156 hours which corresponds to 6 ECTS.

4.1 Practicals

The order of the labs given in section 5 was in use in the academic years 2022/23 and 2023/24. In the 2024/25 academic year the labs 2 and 3 were swapped so that students can understand more easily some thresholding algorithms which use convolution.

For each lab the students have to submit feedback on the manuals so that these could be improved for next year. The feedback was collected anonymously using an online form.

To complete a lab, the student has to complete all of the tasks described in the lab manual. The practical sessions are taking place every week for four hours. Instructors offer consultations (taking place every other week) for students, who either encounter problems while working on their own, run out of time during the lab, are ill during the lab, etc. Students are expected to work beyond lab hours if their progress is slower than anticipated.

The Python programming language is used in the labs. In addition to native Python features the students use OpenCV and NumPy libraries in all of the labs. The OpenCV library is used primarily to read in the image data from files, but in most cases, the image processing itself is implemented by the students. In addition to OpenCV and NumPy, students use several other libraries in some of the labs depending on the specific topic covered, such as cProfile for profiling, DepthAI for running vision pipelines on accelerated hardware [17], Keras and Tensorflow for neural networks, Picamera2 for communication between the PiCameraV2 and

Figure 1. Timing diagram of the course in academic years 2022/23 and 2023/24. The diagram uses the following abbreviations: Lec - Lectures; Sem - Seminars; Consul - Consultations; Midt - Midterm; Proj - Project; Demo - Project demo. Every column represents an academic week during the semester and the final column combines term time weeks.

Raspberry Pi model 5, SciPy for model fitting and Scikit-image for some clustering algorithms.

The tasks in the labs are focused on implementing some of the classical image processing algorithms. Implementing algorithms from scratch gives the programmer a deeper knowledge of the algorithm and ability to extend the algorithm [5]. This is deemed to be an important skill in tech industry and science. The topics were chosen to cover the algorithms in classical image processing and so that it would be possible to build an image processing pipeline using only the covered topics with some exceptions. This gave an opportunity to build a full pipeline using only the code written by the students themselves.

First labs start by focusing on the lower level concept of an image, working with the byte string that represents an image and implementing algorithms that work on the image pixel by pixel. As the course advances, the tasks are oriented to take a look into the features of an image that consist of a group of pixels. Finally the focus is directed towards using images in neural networks, giving an even higher abstraction layer and bridging the cap to courses focusing on more modern methods.

The lab manuals used in the lab sessions were created as part of this thesis. Giving the lab sessions is not part of this thesis and not a sole effort of the author.

4.2 Lectures and seminars

During the course there is a single timeslot for lectures and seminars. The division of weeks between the seminars and lectures can be seen from Figure 1. On four of the weeks there is a lecture by the instructors. The topics of the lectures are chosen so that they would match with the topics of the labs. On the rest of the weeks there are seminars. In all of the seminars up to three students present a scientific article. The seminars are divided into 4 blocks by the lectures. Instructors provide students with a list of options to choose their article from. The topics for the seminars are chosen such that they align with the topics covered in the labs. This helps the

students to understand the lab assignments better and solve these more quickly.

Each seminar presentation has 30 minutes allocated. In the first 15 minutes the student presents the research article. In the last 15 minutes instructors and fellow students can ask questions related to the article. The slides for the presentations must be uploaded to the document upload page 24 hours before the seminar itself. This is required so that students would not prepare for the seminar during the night before and can come to the seminar rested.

Preparation and leading of the seminars and lectures is not the focus of this thesis and not the sole effort of the author.

Lecture 1. The first lecture focuses on the overview of the course and organizational info. In addition to that the concept of colour spaces is covered. The colour spaces that students will encounter in lab 1 are explained.

Lecture 2. The second lecture focuses on the blob detection and how it is implemented in the OpenCV library. The implementation is motivated by the article "Topological structural analysis of digitized binary images by border following" written by Satoshi Suzuki and Keiichi Abe in 1985 [26]. The lecture goes through some of the implementation details and provides reasoning for parts of the algorithm. The students will implement the algorithm by Suzuki and Abe in lab 4 and they will use it heavily in lab 5.

Lecture 3. The third lecture focuses on the internal workings of cameras. This is for lab 6 where students work with more specialised cameras such as the camera module for Raspberry Pi and OAK-D made by Luxonis.

Lecture 4. The final lecture focuses on neural networks and how they are used in image processing. This is useful for lab 8 where students will train and use a neural network for identifying the "SET" game cards.

4.3 Course project

In the second half of the semester, students have to complete a project on their own. The students have to create an image processor that can identify the state of a board game and recommend the next move. For example, one of the board games that has been used in the project is Carcassonne, where players have to place tiles to build structures such as roads and castles to earn points [14]. In this project the student implemented recognition of the structure of a single tile and move suggestion where it could be placed to follow the rules of the game. This requires analysing all of the cards on the board as well.

The first stage of the course project is choosing a suitable board game. For this, the students have to create a Moodle forum post about the game and what parts of the game dynamics they are planning to implement. Additionally, they should describe what methods they plan to use to detect the state of the game. Finally, they also have to mark down what do they plan on completing after working 20 hours on the project. The instructors will go over the proposed plan and give feedback. If some of the details do not seem achievable or have a focus which is not placed in digital image processing, the student has to modify their initial proposition. After this the project is confirmed and the student can start their work.

The second checkpoint of the project is a Moodle forum post in which the students describe what they have accomplished during the first 20 hours (out of the allotted 35 hours) of their work. There is no set deadline for this checkpoint, they have to make it before the final demo of the projects. Students have to mention in this post what is already working, how long it took to implement each part, what they are currently working on, comparison with the original plan for the first 20 hours and description of minimal set of activities that are required to finish the project. The times at which students from 2022/23 and 2023/24 academic years submitted their 20 hours update is visible from Table 1. We can see from the table that most of the students submitted their 20 hours update very close to the final demo.

Table 1. Table representing how long before the final demo the students made their post 20 hours update.

Academic year	Time remaining until the final demo after making the 20 hours update per student (hours)
2022/23	25
	90
	90
2023/24	20
	30
	340

The final step of the project is the demo of the projects. In the demo all the students have 15 minutes to present their project and 15 minutes to answer to questions from the instructors and other students. They should also describe one part of their solution in detail and explain how it is implemented in their code.

4.4 Evaluation

The final grade for the course is based on several different components. The following is the list of categories where students collected points that affected the final grade during the academic years 2022/23 and 2023/24.

4.4.1 Seminars

Students can earn up to 7 points from each seminar. The 7 points are divided into the following categories:

1. **Visual** (2 points)
   How good did the slides (or other assisting visuals) look? Were the slides consistent? Was the text clear and easy to read?
2. **Content** (2 points)
   Was the content of the article covered sufficiently and correctly?
3. **Performance** (1 point)
   Did the presenter engage the audience? Were the points articulated well? Was there a good introduction and conclusion to the presentation?

4. **In time** (1 point)
   Were the slides (or other assisting visuals) submitted on time (24 hours before the seminar)? Was the length of the presentation within the required range?
5. **Student factor** (1 point)
   In addition to instructors the students voted on paper on how good was the presentation from their perspective. This was averaged out between all of the students.

4.4.2 Midterm

Students can earn up to 20 points from the midterm. To pass the midterm they need to get $\geq 51\%$ of the points. The midterm is graded in 2 parts - theoretical and practical. Midterm focuses on the topics covered in the first four labs. The lectures, seminars and labs included many overlapping topics. Topics addressed in the seminars or lectures but not in the labs are not part of the midterm.

1. **Theoretical task**
   The students can get up to 8 points in this category.
2. **Practical task**
   The students can get up to 12 points in this category.

4.4.3 Project

Students can earn up to 27 points from the project. To pass the course they need to get $\geq 51\%$ of the points. The projects are graded in a single stage after the final demo.

4.4.4 Bug bounty

Students can earn extra points by pointing out mistakes made by instructors in any of the materials. Finding smaller mistakes (typos that do not change the meaning of the sentence) gives smaller amount of bonus points (around 0.1 points) and finding larger mistakes (conceptual mistakes) gives larger amount of bonus points (around 1 point).

5 Lab manuals

One of the main focuses for the students is the work they need to complete in the labs. The lab materials are also the primary focus of this thesis. In this section the 8 created manuals are described that were in use in the academic years 2022/23 and 2023/24.

5.1 Lab 1: Image files and colour spaces

The first lab is an introduction to the course. The lab covers multiple topics. The focus is on converting to different colour spaces (for example CMYK and HSV colour spaces) and how the conversion algorithms between colour spaces work. In addition to that different ways how to use the NumPy library is demonstrated.

PNG is one of the most common image file formats used on websites as seen in Figure 2 taken from [36]. To explore this format more deeply, the students implement a parser with limited functionality.

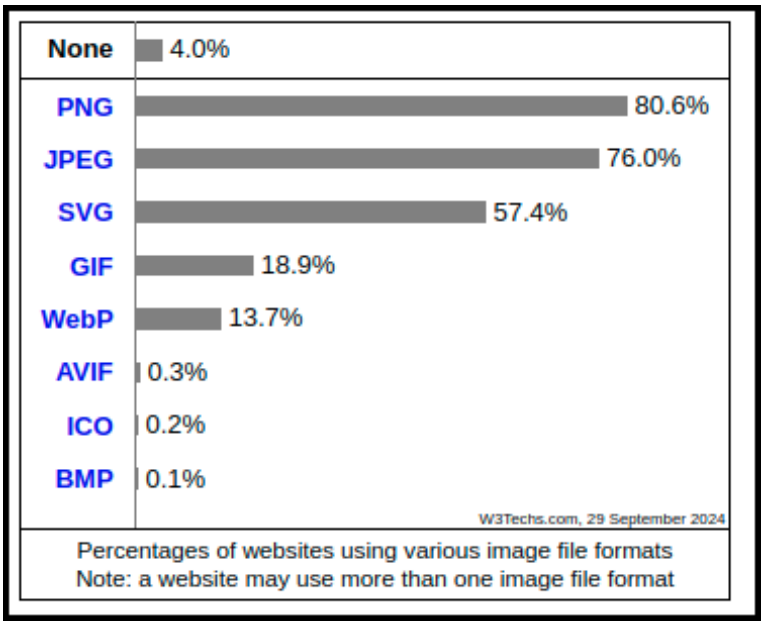


Figure 2. Usage of different image formats in websites [36]

There are four critical chunks defined in the PNG standard: IHDR, IDAT, PLTE and IEND. The students implement functionality to detect 3 of these - IHDR, IDAT and IEND. The fourth critical chunk PLTE is not used in the implementation since it would add complexity to the task and it is not required for the *truecolor* colour mode. The solution implemented by the students assumes that the *truecolor* colour mode is used. In *truecolor* mode all the pixels are represented as red, green and blue colour channel values [4].

After reading in all of the data from the PNG file, the students have to check if the image is interlaced or not. They have to deinterlace it with their own code if needed before showing the image. The PNG image might be interlaced with Adam7 algorithm [4].

5.2 Lab 2: Thresholding and gamma correction

The second lab focuses on implementing different thresholding algorithms. Image thresholding is an approach for image segmentation.

The first algorithm the students have to implement is thresholding a greyscale image using a single pivot value. The pixels below the pivot value are assigned into one class and pixels above the pivot are assigned into a different class.

After that, the students have to implement an algorithm that uses 2 pivot values per colour channel. If all of the colour channel values lie between the corresponding pivot values, the pixel is assigned to one class otherwise the pixel is assigned to a different class. In addition to that, the algorithm includes the possibility to select all of the values outside of the specified range for some colour channel and still be able to select values included in the range for some other colour channels. This is not an option in OpenCV library's *inRange* function [28] which this algorithm is based on.

Thirdly, the students create an adaptive thresholding algorithm for greyscale images. In adaptive thresholding the pivot value is calculated separately for each pixel. The pivot is defined as some constant subtracted from the mean value of the pixels in some neighbourhood of the pixel. The mean value is calculated in 2 different ways - using arithmetic mean and Gaussian weighted mean, where pixels closer to the central pixel have a higher weight.

Finally, the students implement automatic thresholding for greyscale images using Otsu thresholding [20]. Otsu thresholding finds a pivot value that maximises the separability of the classes. Students find this value using K-means clustering algorithm with $K = 2$, that they implement on their own [21].

In addition to thresholding, students implement algorithms for bit plane slicing and gamma correction. Bit plane slicing is introduced as a way to compress raw images by keeping only the four most significant bits of each pixel. The students have to implement an algorithm to compress an image by keeping only the four most significant bits of every pixel and combining two adjacent pixels into a single pixel value. After that, they also have to implement an algorithm that can show an image compressed in this way.

Gamma correction is introduced as a way to fix illumination in images. For example, if an image was captured in dark conditions or the settings on the camera were not configured properly then gamma correction can be used to fix some of these problems. The effect of gamma correction can be seen from Figure 3.

(a) Image before gamma correction

(b) Image after gamma correction

Figure 3. Example input (a) and output (b) of the gamma correction algorithm to make the image brighter

5.3 Lab 3: Morphological operations and convolution

The third lab focuses on morphological operations and convolution. The first task covers fundamental morphological operations such as dilation and erosion. The students have to implement these algorithms. In addition to that the students also have to implement morphological operations based on dilation and erosion, such as opening, closing and skeletonisation.

After that, the lab focuses on convolution and its applications. Firstly, the students have to implement an algorithm for two-dimensional discrete convolution. For this they are given the following formula:

$$(f * g)[y, x] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n, m] \cdot g[y - n, x - m]$$

where f and g are two-dimensional arrays, $*$ is the convolution operator and N, M are the dimensions of the first operand of the convolution.

After implementing the convolution for greyscale images, the students have to implement an algorithm to convolve images with multiple colour channels. This is done by convolving all of the colour channels separately.

The lab also covers some example kernels that are useful in different situations. The students have to find and implement the kernels for box blurring, Gaussian blurring, Sobel edge detection and sharpening.

Lastly, the students look at a more intricate algorithm for blurring - bilateral filter. It calculates a new pixel value by calculating the weighted average of the pixel values in the surrounding neighbourhood. The weight for each neighbouring pixel is calculated using 2 components - spatial component and intensity component. The pixels that lie further from the central pixel spatially have smaller weight since farther pixels should affect the value less, this is the same idea that is used in Gaussian blurring. The pixels that have an intensity value that is more different from the intensity value of the central pixel have smaller weight since these pixels have a higher probability of being part of a different structure in the image and thus should affect the final value less.

5.4 Lab 4: Blob detection

The fourth lab focuses on implementing a feature extraction algorithm inspired by OpenCV library's SimpleBlobDetector class [27]. The purpose of this is to show students how the OpenCV library's SimpleBlobDetector class works so when they work with it in the future, they will know the quirks and parameters of this algorithm. Additionally, most of the students have used the SimpleBlobDetector class [27] in the Robotics I course that is a mandatory course for the Computer Engineering Bachelor's curriculum at the University of Tartu. First step of the solution is to extract the contours of the thresholded image using the border following algorithm by Suzuki and Abe [26]. This is an example of an algorithm that traverses through the image pixel by pixel. It uses more complex data structures and logic to achieve the detection of the contours (borders) from the input binary image.

After the students have implemented their contour detection algorithm they look into the attributes of this contour. For this they use image moments. Image moment of order p, q of an $M \times N$ is defined as follows:

$$m_{pq} = \sum_{y,x} I[y, x] \cdot x^p \cdot y^q$$

where $I[y, x]$ is the pixel value at coordinates y and x [11].

These moments depend on the location of the contour in the image coordinate system. There is also a variant of image moments known as central moments that are calculated using the centre of mass of the contour:

$$\mu_{pq} = \sum_{y,x} I[y, x] \cdot (x - \bar{x})^p \cdot (y - \bar{y})^q$$

where $I[y, x]$ is the pixel value at coordinates y and x, $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$.

Image moments are useful since combining them in a certain way gives results that are "invariant to translation, scale change, mirroring and rotation of the contour" [11]. Some of these combinations describe physical attributes of the shape of the contour, students use these moments to calculate the area and the inertia of the contour.

The students also calculate the perimeter of the contour which is defined as the sum of Euclidean distances between adjacent pixels.

Knowing the perimeter and area of the contour, the students calculate the circularity of the contour which is defined as the quotient of the area of the contour and the square of the length of perimeter of the contour.

Finally, the students calculate the convexity of the contour. The convexity is defined as the quotient between the area of the contour and the area of its convex hull. The students have to implement Graham's scan algorithm to find the convex hull of the contour [12]. After that they calculate its area using moments as well and find the convexity by taking their quotient.

To calculate the image moments one option would be to iterate through all of the pixels inside of the contour using the flood fill algorithm [13]. This algorithm has quite high complexity and it takes a considerable amount of time and memory to use it on bigger contours. Instead the students calculate the moments using a mathematical shortcut known as Green's theorem [37]. Green's theorem is a mathematical theorem that gives a way to calculate the integrals of the area bounded by the contour by iterating over the points on the contour. This reduces the complexity significantly.

This lab has many complex algorithms that the students have to understand. For this reason, the students are provided the pseudo-code for these algorithms, that they have to implement.

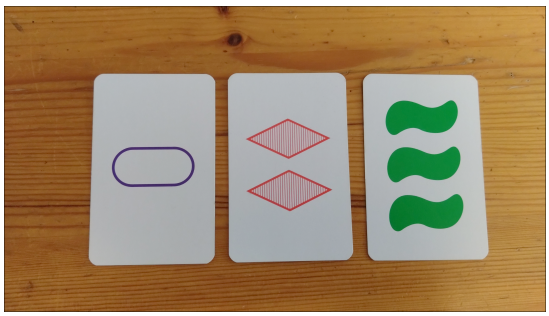
5.5 Lab 5: "SET" game solver

The goal of lab 5 is to combine previously learned knowledge and implement a fully working image processing solution that analyses the cards of the card game "SET". The task is to create a program that can classify "SET" playing cards from an image containing 12 different cards. After the cards have been classified, the program has to find triplets that form a set according to "SET" game rules [22].

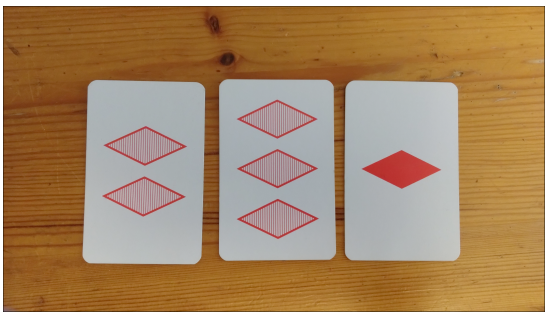
"SET" is a card game where each card contains different shapes. Each card has the following 4 features: [22]

1. Colour - Every card has shapes that are either red, green or purple. All the symbols are the same colour on one card.
2. Shape - Every card has shapes which are either diamond, oval or a squiggle. All the symbols on one card are of the same shape.
3. Shade - Every card has shapes that have the same shading - either empty, partial or full.
4. Count - Every card has either one, two or three identical shapes.

"A valid triplet consists of 3 cards in which each of the cards' features, looked at one-by-one, are the same on each card, or, are different on each card" [22]. Figure 4a shows an example of a valid triplet. From the figure it can be seen that all the cards have different colour, different shape, different shading and different count of symbols on them. Whereas Figure 4b shows an example of an invalid triplet. Although the cards have the same colour, same shape and different counts, two of the shapes have the same shade but the last symbol has different shade.



(a) Valid triplet



(b) Invalid triplet

Figure 4. Examples of valid (a) and invalid (b) triplets

This lab requires students to build an entire system that solves a classification problem combining classical image processing methods learned in previous labs. This is a complicated task that consists of many different subtasks. To easily toggle between different subtasks the students are given a Python base code that gives the user opportunity to turn on and off different parts of the code. It also helps with the solutions being quite slow and running all of the different subtasks continuously would have taken a long time. The overview of the image processing pipeline structure is shown in Figure 5.

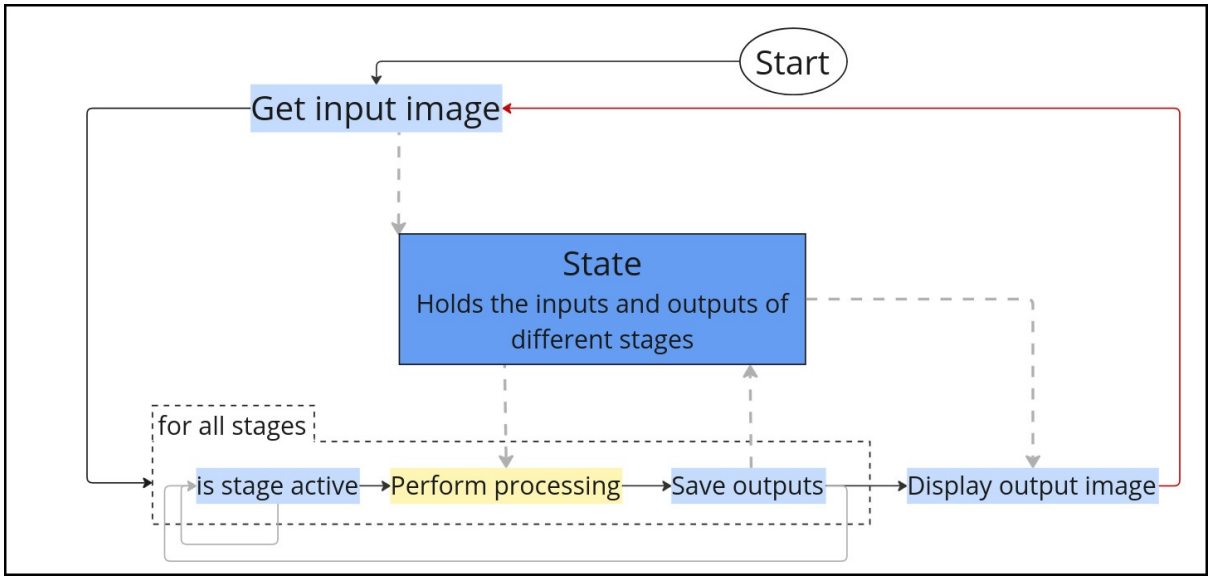


Figure 5. Image processing pipeline structure. Shows the processing of a single image in the Pipeline class structure. Black solid lines represent the flow of the program. Grey dashed lines represent the data flow between different parts of the program. Red line represents the infinite loop that starts the processing operations for the next frame. This loop continues until the user stops the program.

The students have to use knowledge from previous labs to blur an image, convert an image's colour space, apply colour correction algorithm, threshold an image, apply morphological operations on an image and detect contours from an image. Using this, the students can detect the shading and shape of a symbol using flood fill and Canny edge detector which are covered in this lab. To compare shapes of symbols from an image to benchmark shapes, mean squared error (MSE) is used [25].

5.6 Lab 6: Cameras

The purpose of this lab is to work with a camera that has a higher frames per second (FPS) than cameras the students are accustomed to (e.g. HP Elitebook G7 web camera, Logitech web camera C920) and to work with a camera that does some image processing already in the hardware. The first half of the lab is completed on the Raspberry Pi using Picamera V2, the second half of the lab is completed on the classroom computer using the OAK-D camera.

The students' task is to measure the coefficient of restitution (COR) of a ping pong ball using a camera. This is a task where the FPS of 30 is not enough to accurately measure COR. Most of the web cameras have FPS of 30 or lower [24]. Thus, a camera with higher time resolution is needed, such as the Picamera V2, which is capable of achieving a FPS of higher than 100. For this the students take a series of images with timestamps of the ping pong ball right before and after the impact with the floor. Assuming the air resistance is negligible, the height of the ball from the table right before and after the impact is a quadratic function depending on time. The camera is positioned perpendicularly to the ground so that the students can use mean pixel value of the ping pong ball in the image as an estimate for the current height of the ball as seen from Figure 6. The setup used in the lab can be seen from Figures 7 and 8. Example set of images taken with the camera can be seen from Appendix I.

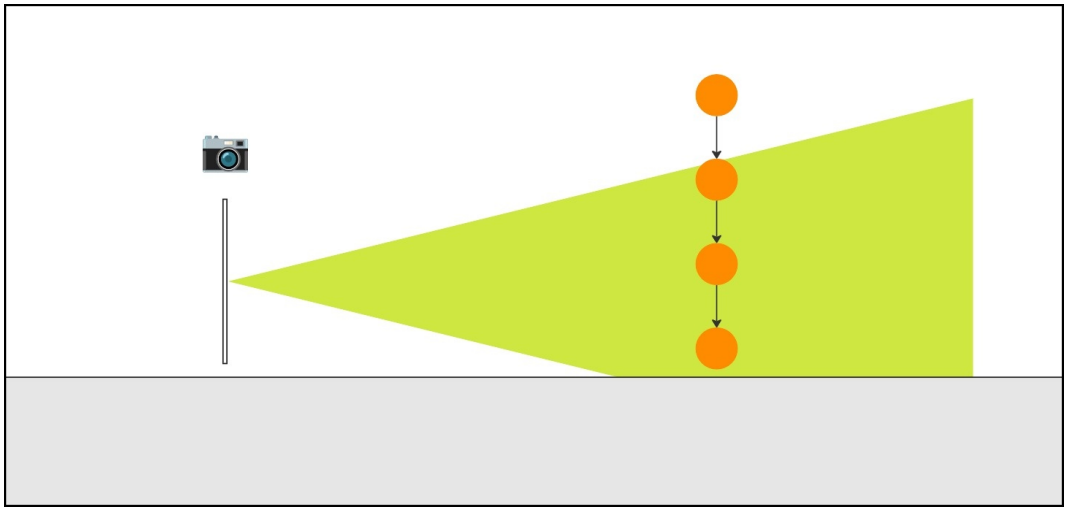


Figure 6. Sketch of the experiment used in lab 6 for measuring the COR of a ping pong ball

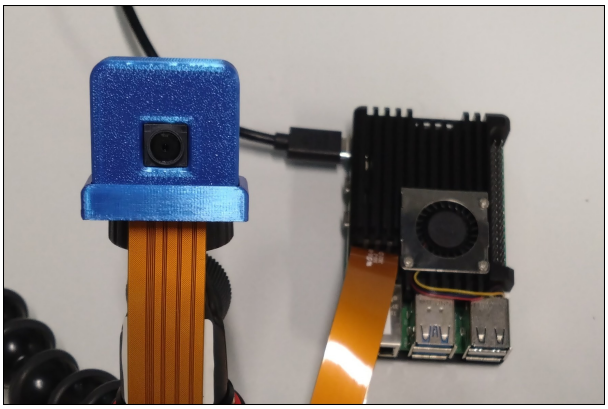


Figure 7. Raspberry Pi camera setup used in lab

Figure 8. Raspberry Pi camera and a ping pong ball; the camera was used to calculate the ball's coefficient of restitution (COR).

Using the timestamps and height of the ball, the students can fit parabolas that model the trajectory of the ball right before and after the impact as seen from Figure 9. The exact timestamp of the collision can be calculated by finding the intersection point of these two parabolas. By knowing the parabolas and collision timestamp the students can calculate the velocity right before and after the impact by taking the derivatives of these parabolas and

evaluating them at the time of the impact. By taking the quotient between these velocities the students can calculate the COR value.

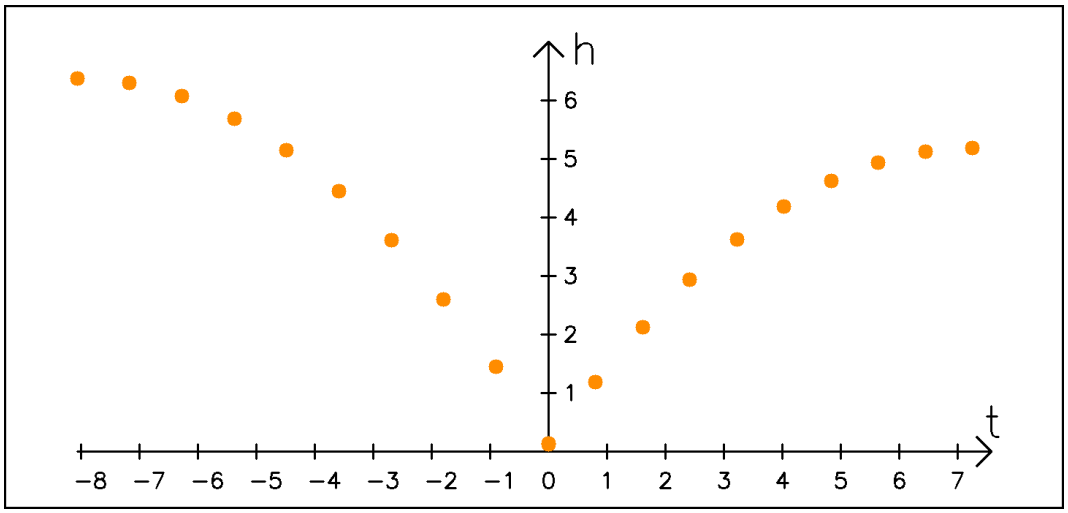


Figure 9. Sketch of the graph students are expected to see after plotting ball height at different timestamps. Units are not specified as to not reveal the solution to the task.

In addition to this the students work with the OAK-D camera that is capable of doing some image processing in the hardware before sending the image to the computer. The camera has multiple camera modules that can be used together to get a stereo image. The stereo image consists of 2 images taken with two adjacent cameras. The OAK-D camera uses the stereo image to calculate the distance of each pixel in a similar way how humans perceive depth. Only the resulting depth map is sent to the computer for software processing. There are multiple other options to configure the OAK-D camera to perform different tasks.

After getting the depth map from the camera the students use simple linear iterative clustering (SLIC) segmentation algorithm on the image to divide the image into superpixels. SLIC is a fast, memory-efficient clustering algorithm that is based on the K-means clustering algorithm [1]. Superpixels are primitive regions of an image that are more perceptually meaningful than individual pixels [11]. The concept of SLIC algorithm is also covered in one of the seminars in this course.

5.7 Lab 7: Optical flow

The purpose of this lab is to introduce the concept of optical flow to the students. Optical flow is a technique for tracking pixels - comparing 2 or more images that are taken right after each other and calculating the velocities that the pixels have between the images [29].

The students implement one algorithm for sparse optical flow (Lucas-Kanade algorithm [16]) and use one algorithm from the OpenCV library for dense optical flow (Farneback algorithm [10]). Difference between sparse and dense optical flow is the amount of pixels that are used to calculate the velocity of a given pixel. Dense optical flow uses all of the pixels to calculate the velocity while sparse optical flow only uses a smaller area around the pixel.

The outputs of the optical flow algorithms used in the labs are in the form of 2 numerical components per pixel which represent the vertical and horizontal components of pixels' velocities. The students implement 2 different algorithms for visualising these. One way to visualise these vectors is to find the longest vector for the neighbourhood and draw it on the image. Another

option would be to encode the vector into a colour space representation in some meaningful manner so that it is easy to recognise which of the velocities are bigger and what are the directions of the velocities. For this, the polar representation of vectors is used, the direction of the vector is encoded into the hue channel of the HSV colour space while the length of the vector is encoded into the value channel.

5.8 Lab 8: Neural networks

Starting from the academic year 2023/24, students solve the same shape and shade classifying task in lab 8 as in the fifth lab but instead classical methods, convolutional neural networks (CNN) are used. The goal of this lab is to give a small introduction to students on how neural networks are used in digital image processing. This lab is structured differently from the other labs in the course - it is instructor-led and completed simultaneously by all students, whereas the other labs are done individually and at each student's own pace.

In this lab the students need to train 2 CNNs (the architecture of the models can be seen in Figure 10). One CNN is used to classify the shade of a symbol on a "SET" card, while a second CNN classifies the shape of a symbol of a "SET" card. The only difference between these CNN-s is the type of pooling layer used: the shade-classifying CNN uses average pooling layer, whereas the shape-classifying CNN uses max pooling. The models are trained using a GPU provided in Google Colab. Keras [19] framework is used to handle working with neural networks.

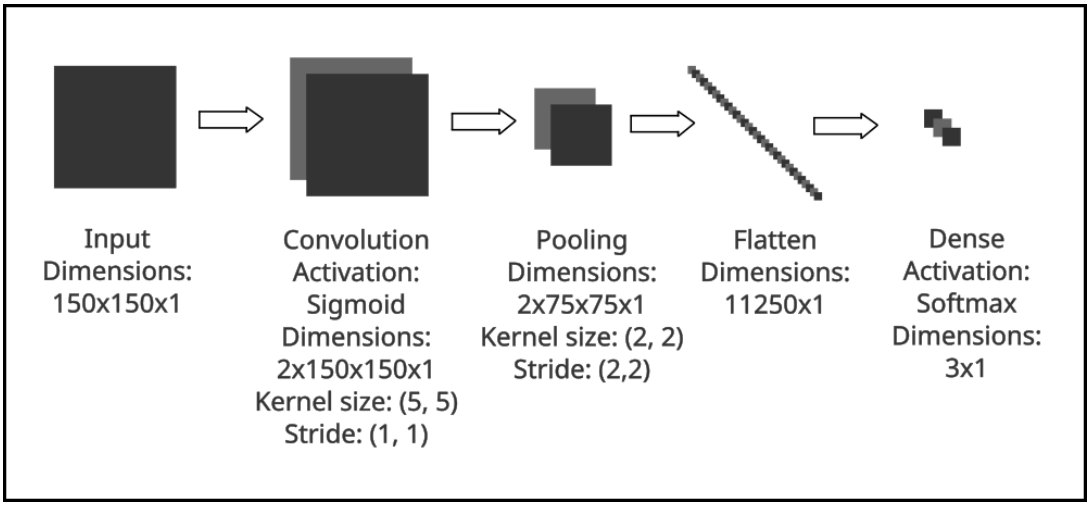


Figure 10. General architecture of the CNNs used in this lab. To classify the shade of a symbol, average pooling was used. To classify the shape of a symbol, max pooling was used.

After training the CNNs, the students save the weights, structure and other metadata of the networks into their computer using the HDF5 file format. Using these weights the students implement pipeline stages to detect the shade and shape of a symbol.

Before feeding the input image into the neural network, the students have to perform some pre-processing on the images such as cutting out only the part of the image where the symbol is located, detecting the orientation of the symbol, and rotating so that it would match the orientation of the symbols from the dataset.

6 Discussion

The 8 lab manuals have been in use since the academic year 2022/23. During these academic years there have been multiple aspects brought out about the manuals by the students and instructors that can be improved. This chapter covers the feedback submitted by the students regarding the created lab manuals.

The feedback for the lab manuals was collected anonymously via Google Forms. Feedback was collected from all of the students who participated in the course either during the 2022/23, 2023/24 or 2024/25 academic year. Students filled the form after completing a lab. Feedback was collected in all of the academic years. Some examples of the changes made based on the feedback can be seen from Table 2. For example, it can be seen that in lab 1 PNG parsing task was too long and it was separated from a single task into 2 smaller tasks.

The concept of lab 8 was changed from digit classification of the MNIST dataset [15] to "SET" shade and shape classification to better link with the previous labs. The application of these models was added to lab 8 as well. Students now have to integrate their lab 8 solution with their lab 5 "SET" classifier to compare the changes of these algorithms.

One plan for the future is to create more bonus tasks that students can complete to earn extra points. In addition, some extra topics could be covered in the labs as well, such as deconvolution. The course is scheduled to grow at least 2 times larger so that all of the students from the RCE curriculum could take it. For this, the course requires more lab and seminar groups. The course should have some extra capacity so that students from other curricula could continue to enrol as well. Another future development proposed by one of the students is the addition of unit tests to the solutions. This would give some assurance that the written solution is indeed correct.

General ideas brought out by the students include the following:

1. Some of the tasks could have more test data to make it easier for students to debug solutions independently.
2. In some cases, the lab manuals were overly prescriptive and listed very specific set of steps that the students need to complete. It was proposed that some of these steps could have been left to figure out for the solver and some of these could have had more argumentation on why it is good do things this way.

Table 2. Some of the changes made based on the feedback

Lab	Comment	Before	After
1	The PNG parsing task was too long.	Whole parsing of a PNG file was a single task.	The parsing of a PNG file was moved into 2 tasks, parsing the PNG file and processing of the PNG byte stream.
3	Indices description in the convolution task was confusing.	Different notations were used to denote the same thing.	Duplicate notations were removed.
3	The theory behind bilateral filter was hard to understand based only on the manual.	Only source of information provided by the instructors was the lab manual.	Lab intros by the instructors were introduced to cover more convoluted topics from the labs.
4	It was hard to see images with small resolution using OpenCV library's visualisations methods.	The students were expected to use OpenCV library's *imshow* function to look at all of the images.	Helper function was added for showing small images by interpolating the image to a larger resolution.
2, 5	The classifier in lab 5 did not achieve as good results as it could due to not ideal luminance conditions.	There was no task that focused on the luminance level corrections.	New task was added to lab 2 that implemented the gamma correction algorithm.
5	The solution structure was messy and the logic behind division of solutions to files was not clear.	Solution was divided between 2 files.	All different stages were assigned a separate file making the division more clear.
5	The accuracy of the shade detection algorithm described in the manual was not high enough.	All of the pixels inside of the contour were considered for determining the shade of a symbol.	Only pixels on the longest diagonal are considered for determining the shade of a symbol.

7 Conclusion

The goal of this thesis was to create 8 lab manuals for the Digital Image Processing course at the University of Tartu that is aimed towards Master's level RCE curriculum students studying at the Institute of Technology. 8 lab manuals were created that focus mostly on the algorithms of classical image processing and the goal of the thesis was fulfilled. Some examples of other digital image processing courses were given. Developed over three years, the manuals were iteratively improved based on feedback from students and instructors. The methods used to create the materials, the structure of the course, and the content of each lab were described. Future revisions may focus on enhancing clarity and accessibility for a broader audience. The course now has a solid foundation and, starting in the 2025/26 academic year, it will available as a mandatory course for all students in the RCE curriculum.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2282, November 2012.

[2] Dan Allen and Sarah White. Asciidoctor | a fast, open source text processor and publishing toolchain for converting asciidoc content to html5, docbook, pdf, and other formats. `https://asciidoctor.org/`. Accessed: 2024-08-21.

[3] Atlassian. Git lfs - large file storage. `https://www.atlassian.com/git/tutorials/git-lfs`, 2025. Accessed: 2025-05-14.

[4] Thomas Boutell. PNG (Portable Network Graphics) Specification Version 1.0. `https://www.rfc-editor.org/info/rfc2083`, Mar 1997.

[5] Jason Brownlee. Benefits of implementing machine learning algorithms from scratch. `https://machinelearningmastery.com/benefits-of-implementing-machine-learning-algorithms-from-scratch/`, 2020. Accessed: 2024-08-21.

[6] Dorian Cojocaru, Liviu Florin Manta, Marian Abagiu, Andrei Dragomir, and Alexandru Marin Mariniuc. Updating the content of a computer vision course for students from the stem programs. In *2022 31st Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE)*, pages 1–6, 2022.

[7] Coursera staff. What is a hidden layer in a neural network? `https://www.coursera.org/articles/hidden-layer-neural-network`, 2024. Accessed: 2025-04-28.

[8] Eastgate Software. The power of image processing: Techniques, applications, and future trends. https://medium.com/@eastgate/the-power-of-image-processing-techniques-applications-and-future-trends-9a3f455e2554, 2024. Accessed: 2025-05-06.

[9] Eclipse Foundation, Inc. Asciidoc | publish presentation-rich content from a concise and comprehensive authoring format. `https://asciidoc.org/`. Accessed: 2025-04-28.

[10] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[11] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing, Fourth Edition, Global Edition*. Pearson, 330 Hudson Street, New York, NY 10013, 2018.

[12] R.L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972.

[13] Yixuan He, Tianyi Hu, and Delu Zeng. Scan-flood fill (scaff): An efficient automatic precise region filling algorithm for complicated regions. *IEEE Access*, 7:78118–78127, 2019.

[14] Klaus Jürgen-Wrede. Carcassonne. `https://www.zmangames.com/game/carcassonne/`. Accessed: 2024-12-09.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Bruce Lucas and Takeo Kanade. Optical navigation by the method of differences. In Aravind K. Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, USA, August 1985*, pages 981–984. Kaufmann, Morgan,

1985.

[17] Luxonis. Luxonis docs | software. https://docs.luxonis.com/software/. Accessed: 2025-04-28.

[18] Akhila Narla, Brett Kuprel, Kavita Sarin, Roberto Novoa, and Justin Ko. Automated classification of skin lesions: From pixels to practice. *Journal of Investigative Dermatology*, 2018.

[19] ONEIROS. Keras | a superpower for ml developers. https://keras.io/, 2015. Accessed: 2025-04-28.

[20] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 62–66, 1979.

[21] Chris Piech. K means. https://stanford.edu/~cpiech/cs221/handouts/kmeans.html, 2012. Accessed: 2025-04-28.

[22] setgame.com. Set. the family game of visual perception. instructions. https://www.setgame.com/sites/default/files/instructions/SET%20INSTRUCTIONS%20-%20ENGLISH.pdf. Accessed: 2024-08-29.

[23] Jose Sigut, Miguel Castro, Rafael Arnay, and Marta Sigut. Opencv basics: A mobile application to support the teaching of computer vision concepts. *IEEE Transactions on Education*, 63(4):328–335, 2020.

[24] Melvin Stanley. Streaming in sync: Is 30 fps good enough for your webcam? https://nexttools.net/is-30-fps-good-for-webcam/, 2025. Accessed: 2025-05-12.

[25] Ken Stewart. Linear regression. https://www.britannica.com/topic/linear-regression, 2025. Accessed: 2025-05-12.

[26] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[27] OpenCV Team. cv::simpleblobdetector class reference. https://docs.opencv.org/4.11.0/d0/d7a/classcv_1_1SimpleBlobDetector.html, 2024. Accessed: 2025-05-13.

[28] OpenCV Team. Opencv: Core array operations — norm() function reference. https://docs.opencv.org/4.11.0/d2/de8/group__core__array.html#ga48af0ab51e36436c5d04340e036ce981, 2024. Accessed: 2025-05-13.

[29] University of Southampton. Computer vision demonstration website. https://www.southampton.ac.uk/~msn/book/new_demo/opticalFlow/#:~:text=Optical%20flow%20is%20a%20technique,in%20the%20next%20image%20sequence. Accessed: 2024-12-09.

[30] University of Tartu. Computational imaging (6 ects). https://ois2.ut.ee/#/courses/LTAT.02.025/details. Accessed: 2025-05-07.

[31] University of Tartu. Deep learning for computer vision (6 ects). https://ois2.ut.ee/#/courses/LTAT.02.028/details. Accessed: 2024-08-21.

[32] University of Tartu. Digital image processing (6 ects). https://ois2.ut.ee/#/courses/LOTI.05.037/details. Accessed: 2025-05-12.

[33] University of Tartu. Introduction to polarization imaging (3 ects). https://ois2.ut.ee/#/courses/LTFY.00.003/details. Accessed: 2025-05-07.

[34] University of Tartu. Multimedia (3 ects). https://ois2.ut.ee/#/courses/MTAT.03.

132/details. Accessed: 2025-05-07.

[35] University of Tartu. Robotics and computer engineering (120 ects). https://ois2.ut.ee/#/curricula/136637/version/2024/details. Accessed: 2025-04-28.

[36] W3Techs. Usage statistics of image file formats for websites. https://w3techs.com/technologies/overview/image_format. Accessed: 2024-08-29.

[37] Weisstein, Eric W. Green's theorem. From MathWorld—A Wolfram Web Resource. https://mathworld.wolfram.com/GreensTheorem.html. Accessed: 2024-09-16.

Appendix

I. Images of falling ping pong ball

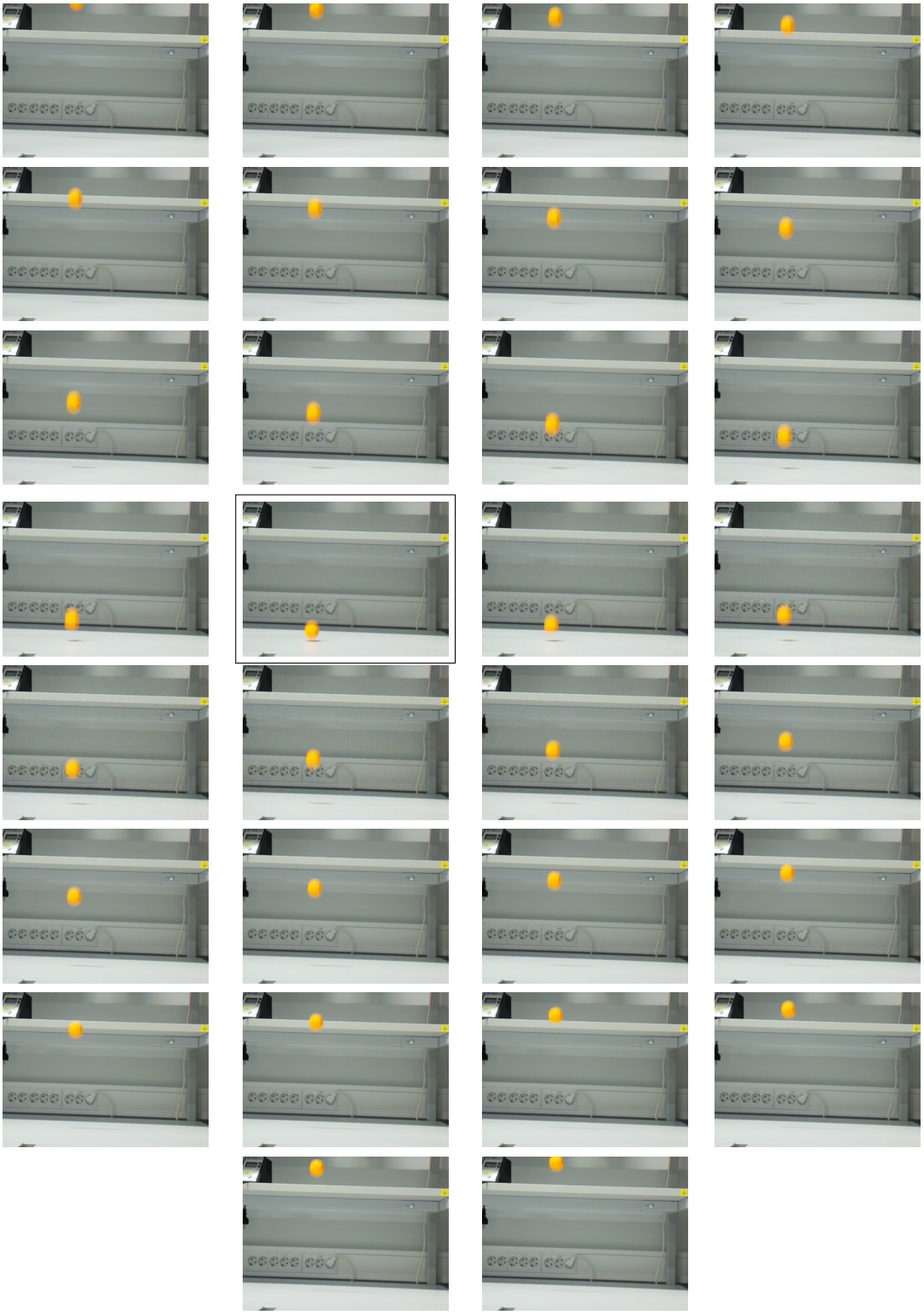


Figure A1: In this appendix there are 30 pictures taken in the timespan of 0.3 seconds of a ping pong ball falling and bouncing from the table. The image of the bounce is highlighted with black border.

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Rasmus Saame**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Design and Development of Materials for the Course "Digital Image Processing",

   supervised by Janno Jõgeva and Marilin Moor.
2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Rasmus Saame
15.05.2025