GRAPEVINE RECOGNITION ON IMAGES

Author:

Yifei Wu

MSc-Student

Supervisor:

Dr. Tomáš Horváth

Head of Data Science Department

December 15, 2018. Budapest

Eötvös Loránd University

Faculty of Informatics, Computer Science

ABSTRACT

Grapevine pruning is a costly and tedious work in vineyard, thus it is necessary to automatize this task with the help of computer vision. Although there are many existing technologies for this purpose, they are either too complicated to implement or highly demanding on the quality and content of the image itself. One of the issues in automating grapevine pruning is the lack of larger benchmark datasets, i.e. images of grapevines and their structure in an annotated form, which would facilitate the development of machine learning-based solutions. This master thesis presents an approach focusing on recognizing the structure of grapevines on images. The thesis contributes to a pilot application, developed within the department of Data Science and Engineering, with the input in the form of image files of grapevines and output in the form of a structured language describing the structure of the grapevine.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to all those who provided me the possibility to complete this thesis. A special gratitude I give to my supervisor of this thesis, Dr.Tomáš Horváth, whose contribution in wise suggestions and encouragement, he and his PhD students helped me to collect the source images from vineyard several times in the freezing winter, all of these helped me to coordinate my project especially in writing this thesis.

Furthermore I would also like to acknowledge Dr.Dmitry Chetverikov who has provided me huge help when I had difficulty on image processing; Mr.Ahmed Blej who is my teammate of this project and helped me to gain the user input.

Last but not least, I would like to thank my family who provides me endless care and love, also my manager at work who gives me so much understanding and convenience.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

In recent years, science and technology have been continuously improved, result in people's living standards have been improved as well, and the role of agricultural machinery automation in people's lives has become increasingly important. Therefore, the development of agricultural machinery automation has gradually become an important symbol representing economic and social development of any country in the world.

The emergence of a large number of agricultural production machinery has effectively solved the problem of insufficient food, while improving the efficiency of agricultural production, it also effectively reduces the labor intensity of operators, so that agricultural production can achieve rapid, stable and sustainable development. However, in today's social and economic development, people have begun to put forward higher requirements for the efficiency of agricultural machinery automation and the quality of products, and also try to apply different technologies in more and more fields. Therefore, the automation of agricultural machinery must continuously improve the comfort and productivity of labor, so that it can effectively promote the continuous development of agricultural production in the direction of high efficiency and high precision.

As one of the most widely planted fruit in the world, grape is also the fruit with the highest processing ratio, the longest industrial chain and the most products. According to the 2017 WORLD VITIVINICULTURE SITUATION REPORT from the International Organization of Vine and Wine, the global surface area under vines is 7.5 million of hectares and grape production is 7.8 million of quintals in 2016. Around 75% of the world's grapes are used for wine making, drying, and juice processing. In addition, the grapes are rich in nutritional value, diverse in products, beautiful in appearance, and popular among consumers all over the world. It can be said that grape is a very important member of the world fruit family, and the production and trade of grapes have always been valued, thus any kind of study which could possibly benefit the viticulture is meaningful.

1.2 Introduction of grapevine pruning

This thesis aims at grapevine pruning which is one of the critical activity in viticulture. For better explanation, please see the grapevine anatomy Figure 1.1 (picture courtesy of Dave Johnson for Bay Area News Group). We can see composition of a grapevine in the picture and also know what is the name of each part, since we will be dealing with the grapevine pruning which usually happen in the winter, also known as the dormant season for grapevines after harvest and leaf fall, there will not be canopy composed by grape bunches and leaves, we can only focus on the trunk, cordons, canes, shoots and buds.

- Trunk – It is the main steam, it's permanent and supports the above-ground vegetative (leaves and stems) and reproductive (flowers and fruits) structure of

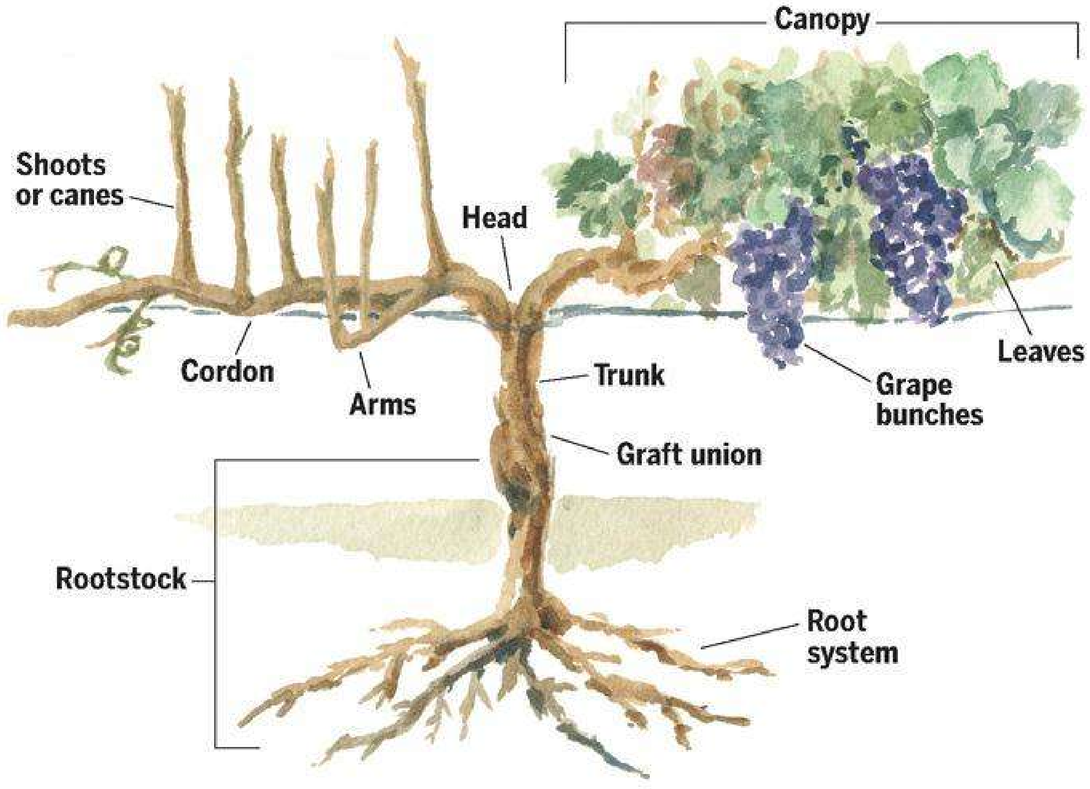


Figure 1.1: Grapevine anatomy

the grapevine.

- Cordon - The cordon, or "arms", extend from the trunk and are the part where additional arms and eventually leaves and grape clusters extend. The cordons are usually trained along wires as part of a trellis system.

- Cane - The principal structure of concern in the dormant season, is simply a mature, woody and one-year shoot after the leaves has fallen off.

- Shoots - They are green stems which develop from buds, and represent the primary growth structure of grapevines. The shoots that arise from primary buds are normally the fruit-producing shoots. The shoot consists of stems, leaves, tendrils and fruits.

- Buds - A bud is a growing point that develops in the leaf axil, the area just above the point of connection between the petiole and shoot. Bud is actually a highly compressed shoot with all its parts, including cluster. In spring, the buds burst. They become shoots bearing new leaves and bunches of grape flowers.

Why do we need to prune grapevine? And why it is considered as one of the critical activity in viticulture? There are several reasons for grapevine pruning:

1. Keep the grapevine in shape. Grapevine belongs to liana, which means it needs to climb by other additional support, and usually in the vineyard, grower needs to train and make sure the grapevine keep similar shape year after year in order to facilitate other operations in the vineyard such as harvest.
2. Keep the grapevine in good health. The excess branches would derive more nutrients and cannot make full use of the nutrients, impeding the growth of other branches. Also for the canopy management which means finding the balance in enough foliage to facilitate photosynthesis without excessive shading that could impede grape ripening or lead grape diseases.
3. Ensure the yield and improve the quality of grapes. Grapes are produced from buds that will grow into shoots on 1-year old canes (the long stems or ”shoots” after they’ve borne fruit for at least one year). The most fruitful canes will be those that were exposed to light during the growing season. A given vine in a given season can only produce a certain quantity of fruit. Its capacity to do so is largely dependent on the amount of leaf area and photosynthetic activity. By consistently limiting the number of shoots and leaves by dormant pruning, one is also working to produce the maximum crop without delaying maturity year after year.
4. Maintain a balance between vegetative growth and fruit production. Where a vine is under pruned, means too many buds left, the vine will produce many

small clusters of small grapes that may fail to ripen properly. If the vine is over pruned, means too few buds left, the yield will be low and the vegetative growth excessive.

The grapevine pruning techniques vary from different varieties of grapevines and different regions the grapevines grow. In general, there are two most common and basic systems of grape pruning - cane pruning and spur pruning, see example in Figure 1.2 (picture courtesy of Eric Stafne, Mississippi State University) of spur pruning. In order to control the number, position, and vitality of branches, as well as the yield and the quality of the fruits, the operation of pruning is required to be conducted manually by experienced operators. Usually, operators walk along the grapevines one by one, and prune each grapevine with secateurs (a dedicated tool for pruning), they need to observe and plan carefully, make proper decision by their experience and knowledge. Such operation could take a group of operators few months to finish, that is simply because nowadays the vineyards are considerably large and there are usually millions of grapevines in the vineyard, even consider a smaller vineyard it might take a couple of weeks to finish.

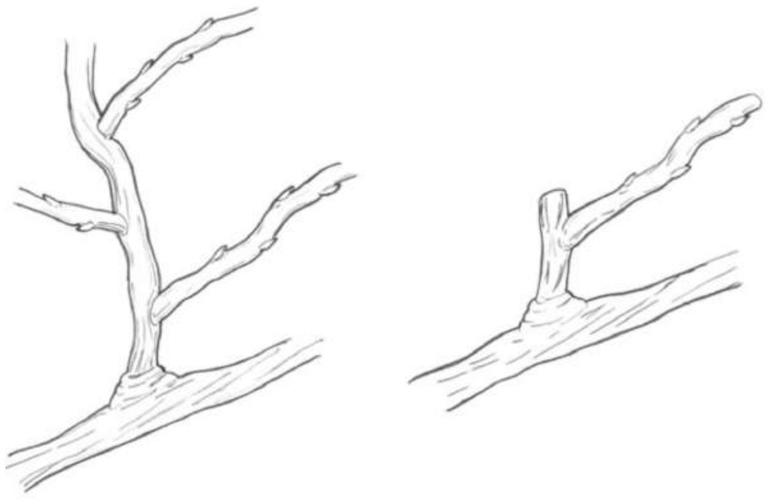


Figure 1.2: Spur pruning example

However, there are so-called semi-automatic pruning machines, see example in Figure 1.3, they are usually installed on standard tractors. Such mechanism can prune the grapevine to a certain extent under the control of human, compare with the manual grapevine pruning, it definitely relieves people from tedious physical work, but it is just impersonal mechanism, it can only prune the grapevine roughly, it cannot perform detailed pruning, the accuracy of the operation depends on the experiences and capability of the driver of this machine to a large extent. It indeed wins on the speed of the process, but loses on the precision of the result when we compare it with the manual work.

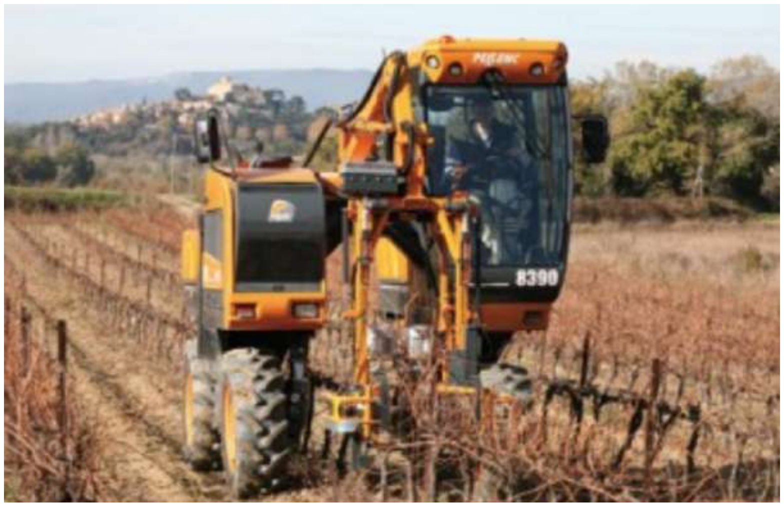


Figure 1.3: Pruning machine example

In practice, such mechanism often leads some stems or branches which need to be pruned get remained while others which need to be remained get pruned, this could seriously damage the grapevine and reduce the yield.

1.3 Objective and preparation

1.3.1 Objective

As mentioned above, grapevine pruning as a time-consuming and labor-intensive work, is the last operation needs to be automated in the vineyard. But, to date, grapevine pruning has not been totally automated due to the variations of grape varieties and environment which make automation quite difficult. The cost of skilled hand labour is rapidly increasing in any country, it is necessary to develop a more efficient and economic way for grapevine pruning.

The main contradiction here is we would like to reach almost the same pruning accuracy as the manual work does without any human get involved. Thanks to computer vision and machine learning, it is possible to achieve our goal, because the knowledge of how to properly prune grapevine can be learned by computer with the help of human intervention and sophisticated algorithms nowadays. However, the objective of this thesis is only about the image recognition of grapevine structure, which could be used for future work.

1.3.2 Preparation

For the first step of our work, we need to gain as many grapevine images as we can, thus we went to a vineyard located in Monor (a town in Pest county, southeast of Budapest, Hungary). We got around 600 pictures of the grapevines in that vineyard before the grower prune them in the winter, all these pictures were taken by a digital camera with tripod, and the horizontal distance between camera and grapevine is approximately one meter. Each picture has 6000*4000 resolution, please see 9 sample picture in Figure 1.4.

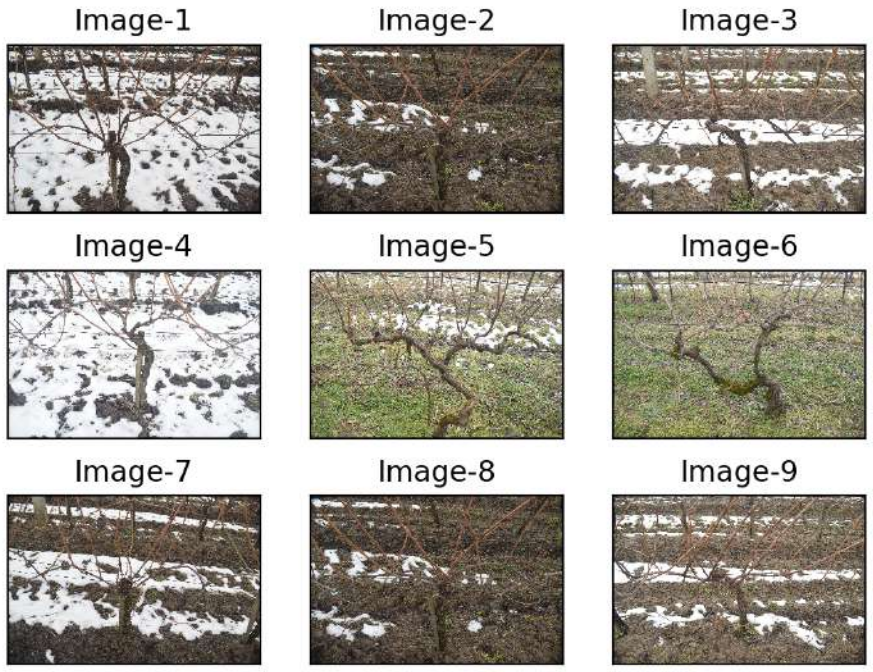


Figure 1.4: Sample pictures

1.4 Outline

The outline of this thesis can be summarized as following:

CHAPTER 1 briefly introduces what is grapevine pruning, the current situation of grapevine pruning, explains the motivation and objective of this thesis, also indicates the preparation we have done.

CHAPTER 2 makes a literature review of several relevant papers, also demonstrates the latest technology in the scope of grapevine pruning, compares the advantages and disadvantages of each approach, highlights the difficulties in our situation.

CHAPTER 3 presents a novel approach to gain grapevine images under any kind of background, this approach mainly depends on the image processing and image analysis techniques. Also expounds all the tools, theories and algorithms we have used to process the image and get the structured representation of the grapevines on images.

CHAPTER 4, a summary of main achievements, contributions, conclusions and future work of this thesis are presented. The defects of the approach in this thesis is also addressed for further improvement.

Chapter 2

Relevant literature review

Since there is huge commercial potential in the field of autonomous grapevine pruning, the research on this matter never stopped. Different people have proposed different ideas and conducted different experiments, some of those ideas also have been implemented to prototypes. In this chapter, we will review a few papers related to autonomous grapevine pruning, demonstrate several creative algorithms and techniques for segmentation, recognition and object detection on grapevine images.

2.1 Existing technology

2.1.1 Systematic solution

In paper [1], they designed and implemented a complicated prototype robot system for grapevine pruning, this system integrated different technologies together. First, a mobile platform straddles the row of grapevines, and it takes picture of each grapevine with trinocular stereo cameras while it moves; secondly, a computer vision system constructs a three dimensional model of the grapevine; then an artificial intelligence

(AI) system makes the decision about which point to prune; finally a robot arm conducts the pruning operation based on the decision made by the artificial intelligence system.

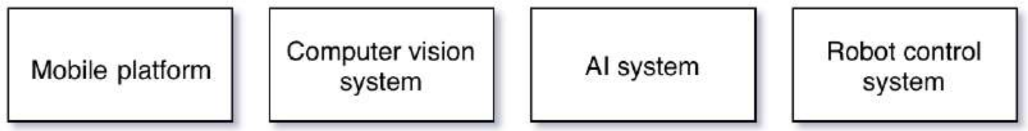


Figure 2.1: Main components of the robot system presented in [1]

As shown in Figure 2.1, this system consists of four major modules, each module has its individual task and needs to provide required output for the next one, detailed introduction of each module is as follows:

1. Mobile platform - The mobile platform provides a desirable and controllable environment for taking pictures of grapevines, the whole construction looks like a small mobile cabinet. This platform straddles a row of grapevines, and blocks out sunlight, three cameras are installed on one side of the wall in this cabinet, and the wall on the other side is colored blue to facilitate the segmentation of grapevines from the background.

2. Computer vision system - The computer vision system takes the images from the trinocular stereo cameras, and then constructs a 3D model of the grapevine's structure based on a set of stereo frames. It uses different technologies such as image segmentation, 2D structure detection, correspondence finding and incremental 3D reconstruction etc. A complete and continuous 3D model of a grapevine is necessary for the next AI system to make decision about where pruning operation should be conducted and which canes should be removed in order to prune the grapevine properly.

3. AI system - This module is in charge of pruning decision support, it gets the 3D

model of each grapevine from the previous module, at the end it commands the next module to operate accordingly. Each grapevine pruning strategy is described by a feature vector of proper attributes of the canes that will be kept. For instance, appropriate attributes can be length, position, distance below wires, angle from head, and whether they grow from the trunk, or another cane. A cost function which is a simple linear combination of features is also used to measure pruning quality using grapevines labeled by viticulture experts. For each grapevine, all the possible pruning scheme is considered one by one, and the highest scoring pruning scheme will be selected.

4. Robot control system - This is the last operation module which includes path planning and robot arm control. The path planner computes a collision-free trajectory taking the robot arm to the selected pruning points, it uses randomized path planner algorithm which requires a collision detector, an inverse kinematics solver and a local path planner. Once the pruning points are selected in the previous stage, the robot arm needs to sweep the cutting tool through the cane to make each cut, this action is called a cut motion, for pruning a cane there can be many possible cut motions. In order to find one that is collision-free, it first samples random cut motions, and then test each one with the local path planner. When it cannot find a collision-free cut motion close to the head, it moves the pruning point farther away. Finally when each cut motion is planned, it will plan a path that visits each cut in turn.

Each module in this robot system for autonomous grapevine pruning has been evaluated against the ground truth, as well as the whole system. The computational performance and evaluation of each module are acceptable, but the result of integrated testing is certainly unreliable. Even though this complicated system does not work well enough to be able to replace manual pruning, it undeniably brings many technologies and inspiration which could possibly enable the grapevine pruning to be totally automated.

There are also some similar systems proposed in different papers, even though they all approach the goal in various ways, the fundamental among these papers are more or less the same, see the flowchart in Figure 2.2 of a proposed process in paper [2] as following:

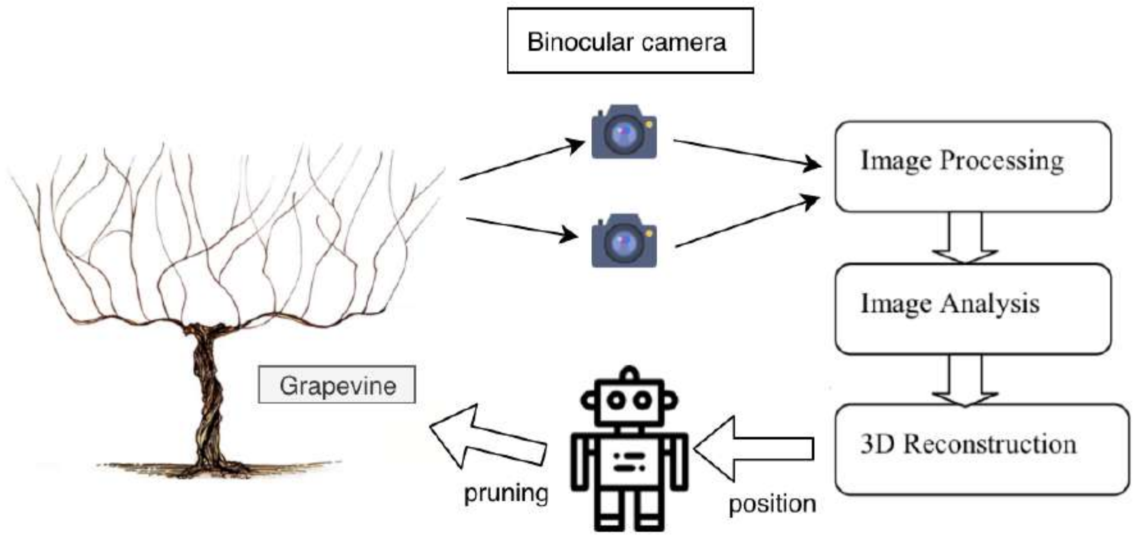


Figure 2.2: Flowchart of a proposed approach from [2]

2.1.2 Structure extraction on images

To our knowledge and based on the research we have done, we can say that the most common opinion on the matter of grapevine pruning automation always includes image processing and image analysis, this part of work is inevitable no matter for 3D model construction or generating source data for future intelligent pruning model training. Usually this part is also where the first big obstacle is located, because we intrinsically expect an output as good as possible from this stage which can benefit the pruning decision support, so the study on grapevine structure recognition becomes significant.

The structure of grapevine in the image should be extracted in the first step, tremendous efforts have been put on this based on different algorithms and perspectives. Also the structure of grapevine is similar with any other kind of fruit trees, theoretically the study on these fruit trees should also be applied on grapevine.

In paper [3], they proposed a new algorithm called simultaneous tracking of two edges to extract the stems of fruit tree. They consider the stem area as a set of scanning beam sequences, mathematically represented by a directed graph G = (V, E). The vertex V is the scanning lines connecting the edges of the two sides of the branches, and the edge E is the path connecting the scanning lines. The goal of simultaneous tracking is to find a shortest path from the source vertex (seed scanning beam) to the target vertex (root scanning beam). Theoretically, there are countless paths from the source vertex to the target vertex. It is not feasible to search for the shortest path by traversing all the paths. To solve this problem, several constraints are added during the search process to limit the number of candidate scanning beams, Figure 2.3 illustrates one step of tracking the scanning beam.

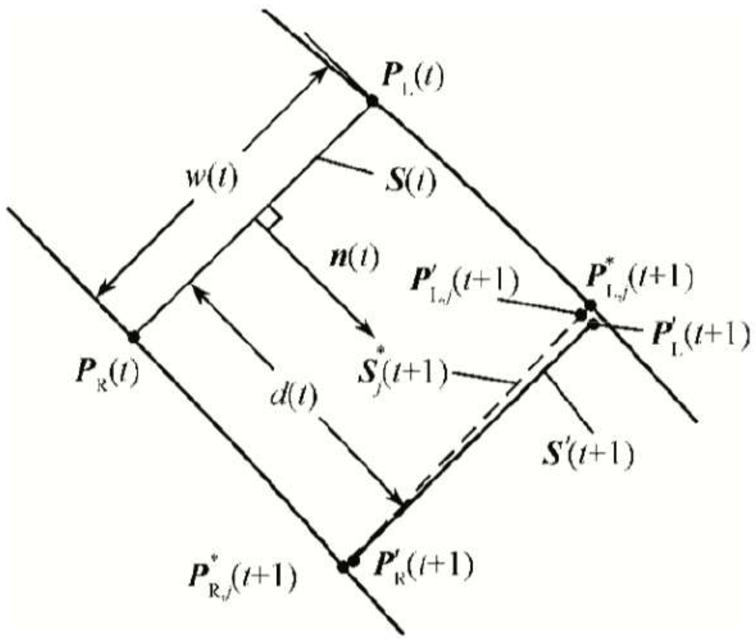


Figure 2.3: One step of tracking scanning beam [3]

Moreover, after getting the stem area of the fruit tree, one can also get the structure information, namely where are those branch points. This can be done by applying image thinning algorithms, there are many image thinning algorithms including several conventional algorithms and some advanced ones [4] [5]. After this we can get the skeleton of the fruit tree, it means we can reduce the structure to one-pixel width, and we can find out the branch points by counting the neighborhood pixels according to some rules. While there can be one problem, the intersection of branches in image could generate pseudo branch points, the solution of this issue has also been presented in [3]. Figure 2.4 shows this problem, N_1 and N_2 are two pseudo branch points generated by the intersection of two branches in image.

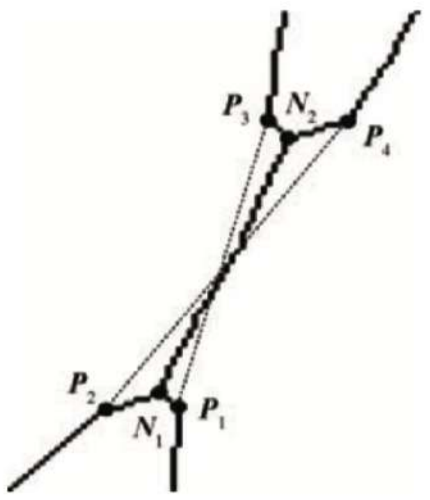


Figure 2.4: Pseudo nodes generated by intersection of branches

Structure extraction and recognition on images is a hot topic in computer vision, especially because of the popularization and rapid development of big data and machine learning recent years, there have been many years of research on this field, and new methods are still emerging. Based on different theoretical backgrounds, thousands of methods have been formed. However, there is no universal method for images under different conditions, and most of them are specifically analyzed for specific problems.

Besides these, there are also some other papers related to image processing which have provided excellent ideas and inspiration for this thesis, so we would also like to refer them here. [6] presents a way of skeletonization of branched volume by shape decomposition; [7] introduces shape deformation using a skeleton to drive simplex transformations; an adaptive Canny edge detection algorithm is presented in [8]; [9] describes a center-line extraction method; [10] introduces an adaptive edge detection algorithm; [11] presents an edge connectivity evaluation algorithm for performance of edge detectors; in [12], a pose-configurable generic tracking of elongated objects are presented, just to name a few research works related.

2.2 Difficulties in our state

From previous chapter, we have seen some sample grapevine pictures we took from the vineyard, due to the limitation, our first difficulty comes from the complex background of images. In those images we took, backgrounds are quite disorderly and messy, it consists of snow, soil, grass, iron wire and some other high-detail regions, all of these components turned the structure extraction and recognition task extremely difficult. Because usually the structure extraction and recognition are done by several different features of images such as color, grayscale difference, gradient, texture, mathematical property and etc., but the complex backgrounds destroy all these features which we can use. And all the literatures we reviewed in the last section, they use a consistent background, for example, using a white curtain behind the grapevine when they take pictures, such way they could get rid of the disorderly background and make the recognition task much more easier.

For instance, if we first simply apply an adaptive thresholding algorithm where threshold value is the mean of neighbourhood area on one of our images, the result shows in Figure 2.5.

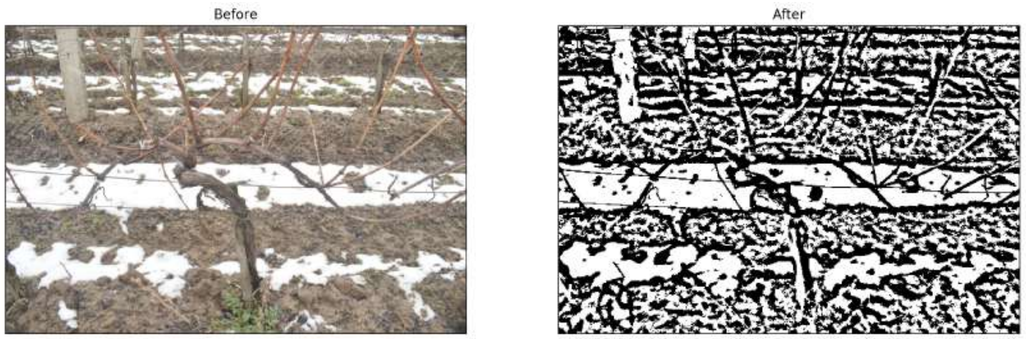


Figure 2.5: Thresholding result

It is obvious that the result is not acceptable and therefore it is not possible to be further processed for getting useful information no matter how we tune the parameters of different thresholding algorithms. Meanwhile we also faced the same difficulty when we apply other algorithms such as edge detection based on gradient filter, the Canny edge detection result shows in Figure 2.6.

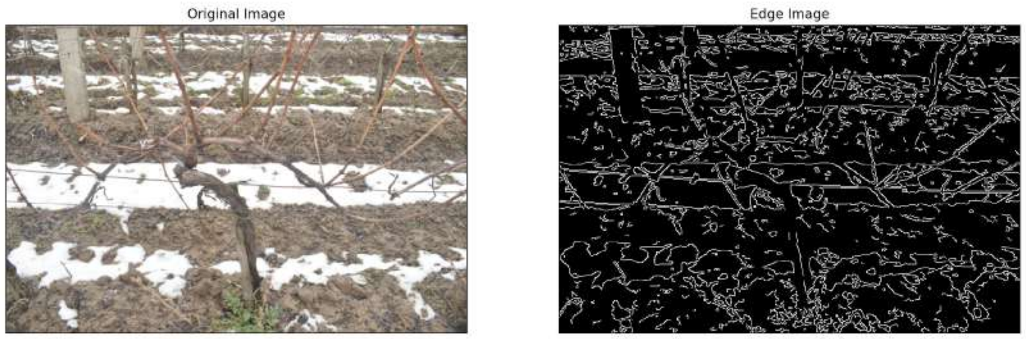


Figure 2.6: Canny edge detection result

Another difficulty is finding hierarchical information of a grapevine, discover where a cane or shoot grow from in the images can be confusing and misleading due to the intersection of different branches. Even though there are some solutions in the former literature review, they are for some other fruit trees which have simpler structure than grapevines, see a grapevine skeleton example in Figure 2.7.

It is clear that the region near the middle part of the skeleton image is quite complicated, the branches cross each other quite often because of the irregularity of the

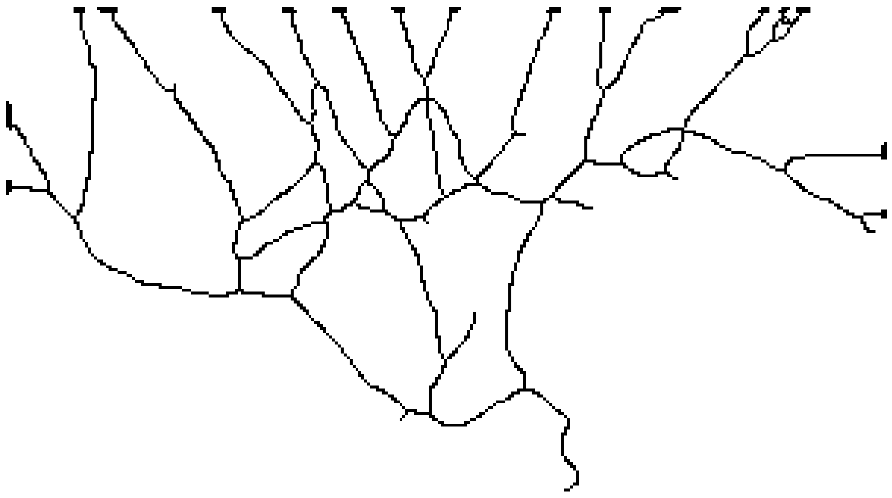


Figure 2.7: A grapevine skeleton example

growth of grapevines, they formed a bunch of closed rings, thus most of the existing algorithms became inapplicable, and this is also the reason why we need to come up with a workaround.

Chapter 3

Our approach

3.1 Overall process

After a long time of trying and thinking, also take the opinions we got from some experts in the field of image processing into consideration, we just realized that it is almost not possible to extract the structure information of grapevines from those images we have because of the complex background, at least no one have ever successfully processed images with such complex backgrounds before. To our knowledge one way that might help is deep learning, but such way requires a lot of data with ground truth, which is unrealistic in our case.

If we want to use any kind of AI technique to help the decision support of grapevine pruning, we first need to transform the image of grapevine to structured representation which can serve as the ground truth, also it requires a mass of images of grapevine. Within the scope of our ability, we are not able to collect enough images as it might requires, also combine the difficulty of image processing under complex background, we think that some kind of user interactions are inevitable, so what we would like to do is providing an application which could benefit us as well as people whoever intend to do the same thing with us. See a general flowchart of such application in Figure 3.1.

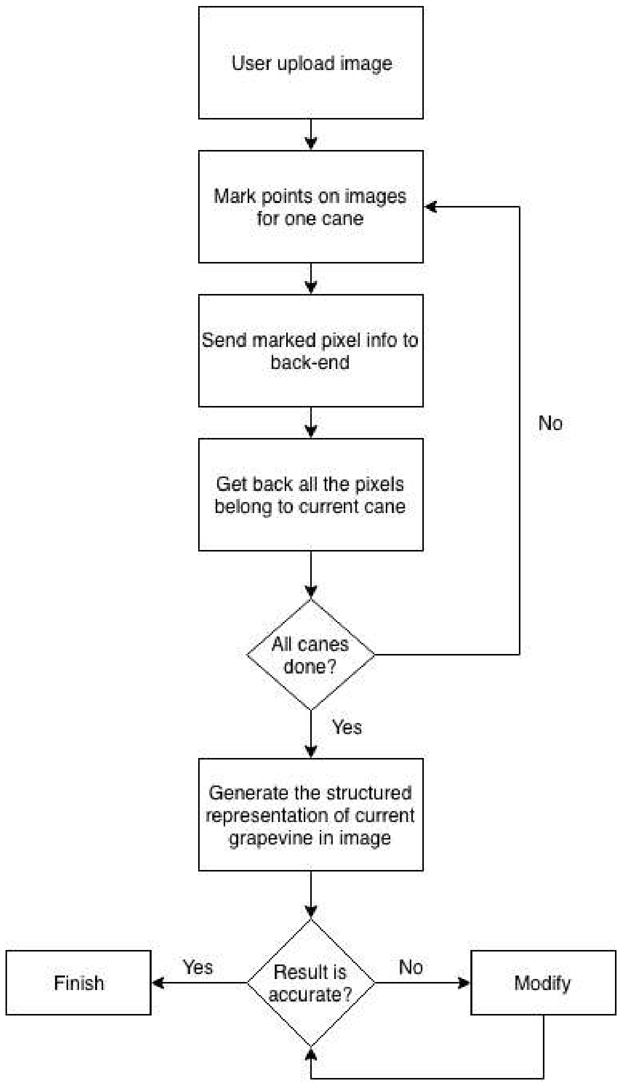

User upload image
Mark points on images for one cane
Send marked pixel info to back-end
Get back all the pixels belong to current cane
All canes done?
No
Yes
Generate the structured representation of current grapevine in image
Result is accurate?
Yes
Finish
No
Modify

Figure 3.1: Flowchart of our approach

Most of the user interactions during the process are basically intuitive, so people without too much technical background can still use this application without any difficulty. The detailed explanation of each step in our approach is as following:

1. At the beginning, we provide an interface for users to upload their grapevine image under any kind of condition, and user could have further operation on this

image. The prototype of this application shows in Figure 3.2.

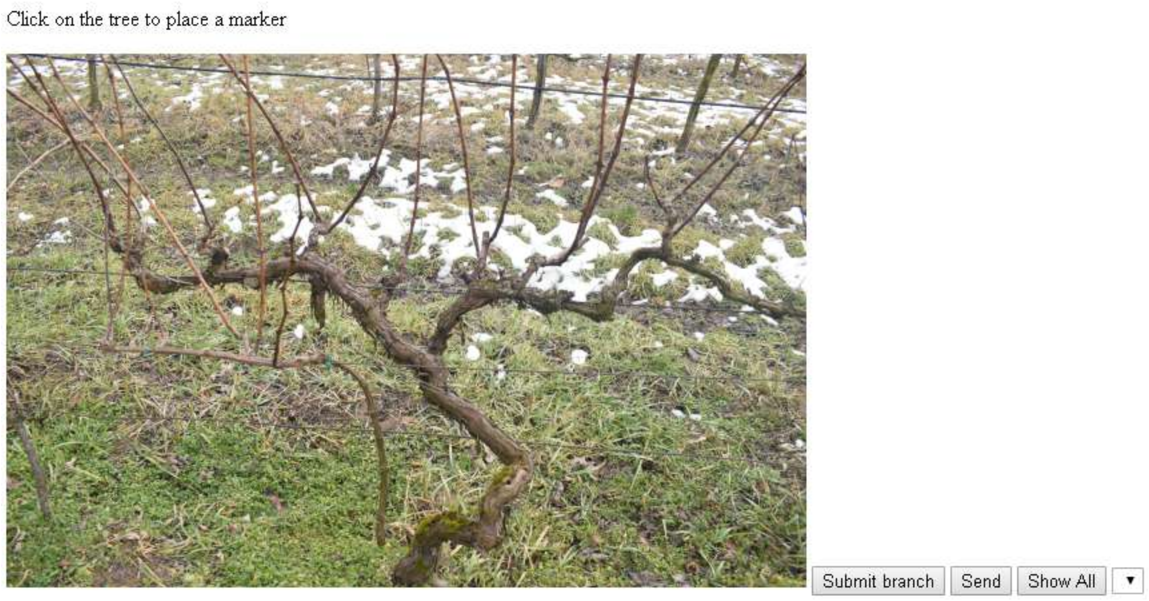


Figure 3.2: Upload image

2. Then the user could click on the image along any branch they choose from beginning to end, so there will be several points that lie on a branch in the image, it looks like in Figure 3.3, all the red markers are generated by mouse click from user.

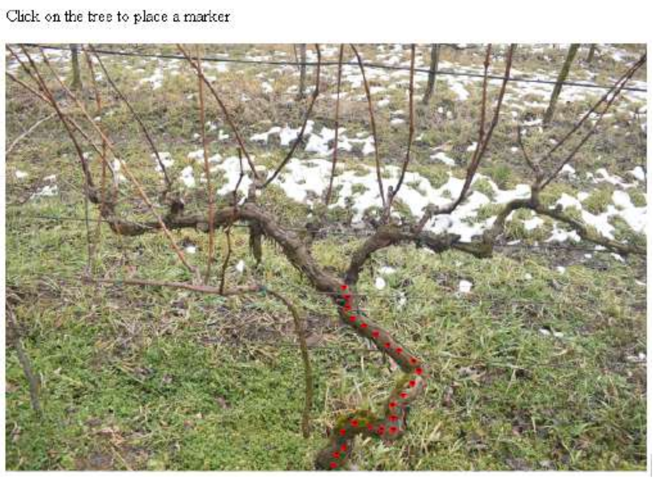
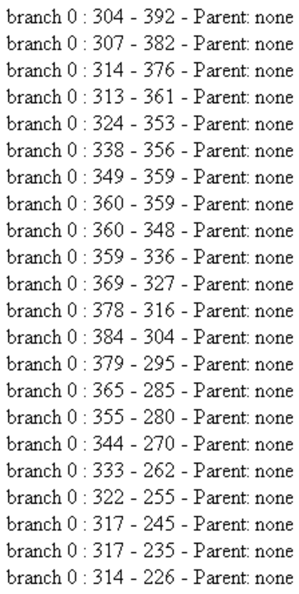


Figure 3.3: Mark the trunk

3. After marking a complete branch from the beginning till the end, user could

submit the pixels information to the back-end for further processing. The submitted information will be in the format like in Figure 3.4, it will send both the coordinates information and the hierarchy for each branch.



```
branch 0 : 304 - 392 - Parent: none
branch 0 : 307 - 382 - Parent: none
branch 0 : 314 - 376 - Parent: none
branch 0 : 313 - 361 - Parent: none
branch 0 : 324 - 353 - Parent: none
branch 0 : 338 - 356 - Parent: none
branch 0 : 349 - 359 - Parent: none
branch 0 : 360 - 359 - Parent: none
branch 0 : 360 - 348 - Parent: none
branch 0 : 359 - 336 - Parent: none
branch 0 : 369 - 327 - Parent: none
branch 0 : 378 - 316 - Parent: none
branch 0 : 384 - 304 - Parent: none
branch 0 : 379 - 295 - Parent: none
branch 0 : 365 - 285 - Parent: none
branch 0 : 355 - 280 - Parent: none
branch 0 : 344 - 270 - Parent: none
branch 0 : 333 - 262 - Parent: none
branch 0 : 322 - 255 - Parent: none
branch 0 : 317 - 245 - Parent: none
branch 0 : 317 - 235 - Parent: none
branch 0 : 314 - 226 - Parent: none
```

Figure 3.4: The selected pixels information of the trunk

4. User needs to repeat the same action for each branch until all the branches are marked, and user will finally get all the pixels within each branch respectively as well as the structured representation of current grapevine.

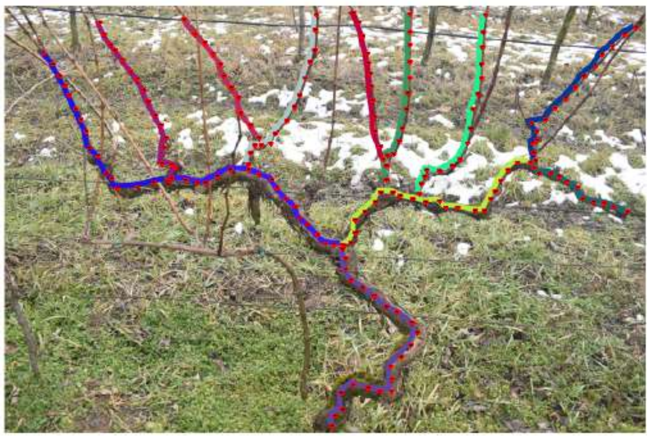


Figure 3.5: All the branches get marked

3.2 Tools we use

In this section, we are going to introduce several tools we used in this thesis mainly, the programming language we used Python, and for image processing and array manipulation we used OpenCV and Numpy respectively.

3.2.1 Python

Python is a programming language developed by Guido van Rossum, which quickly became very popular, mainly because of its simplicity and code readability. It enables programmers to express ideas with fewer lines of code without compromising readability.

Speaking of scientific computing, the first thing that will be mentioned is MATLAB. However, in addition to some of MATLAB's highly specialized toolboxes that cannot be replaced, most of the common features of MATLAB can be found in the Python world. Compared with MATLAB, scientific computing with Python has the following advantages:

1. First of all, MATLAB is a commercial software and it is expensive. Python is completely free, and many open source scientific computing libraries provide Python's calling interface. Users can install Python and most of its extension libraries for free on any computer.

2. Second, Python is an easier-to-learn and more rigorous programming language than MATLAB. It allows users to write code that is easier to read and maintain.

3. Finally, MATLAB focuses on engineering and scientific computing. However, even in the field of computing, various requirements such as file management, interface design, and network communication are often encountered. Python has

a rich library of extensions that make it easy to perform advanced tasks, and developers can use Python to implement the full range of functionality required for a complete application.

Python is slower than languages like C/C++. That is, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create a Python wrapper that can be used as a Python module. This brings us two benefits: First, the code is as fast as the original C/C++ code (because it is the actual C++ code that works in the background), and secondly, writing code in Python is easier than using C/C++.

3.2.2 OpenCV

The full name of OpenCV is Open Source Computer Vision, which is the most popular open source library for computer vision. Here, OpenCV is a library, it is based on the C++ language, of course, can also be used on other platforms, such as Java, Python, this library is used in the field of computer vision, and is open source, that is, we can get its source code and it can be modified to our own needs.

Computer vision includes underlying image processing, middle-level image analysis, and high-level visual technology. Due to the development and maintenance of many experts of computer vision, OpenCV has extended to every field of computer vision, and its functions cover almost every research direction. Moreover, the algorithms it implements follow the frontier of computer vision and incorporate the latest technological algorithms.

OpenCV is very powerful, and its influence in academia and industry is growing. As a classic of numerical calculations, MATLAB is also one of the most commonly used tools in the field of image research. In the past, many authors of papers provided

MATLAB code. Due to the emergence of OpenCV, especially OpenCV2.x, it greatly promoted the C++ programming for researchers in the image processing field, so many researchers now directly give the C++ code implemented in their article. The advantage of OpenCV for MATLAB is that in most cases, C++ has good efficiency. At the same time, using C++ programming has great advantages for researchers to enter the market to find work. In many image-related companies, they also require the job seekers to be familiarized with OpenCV. So OpenCV is indeed a very important tool in the image world and is highly recommended by many experts in the filed of image processing.

In OpenCV, all algorithms are implemented in C++. But these algorithms can be used from different languages like Python, Java etc. This is made possible by the bindings generators. These generators create a bridge between C++ and Python which enables users to call C++ functions from Python. To get a complete picture of what is happening in background, a good knowledge of Python/C API is required. Extending all functions in OpenCV to Python by writing their wrapper functions manually is a time-consuming task. So OpenCV does it in a more intelligent way. OpenCV generates these wrapper functions automatically from the C++ headers using some Python scripts.

3.2.3 Numpy

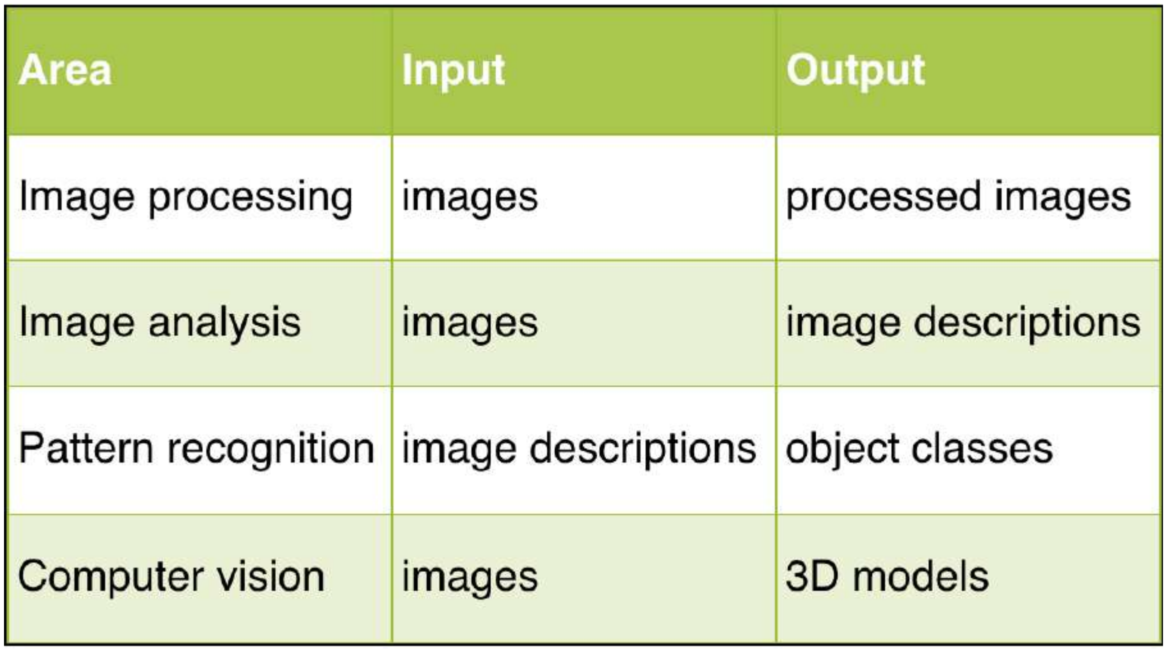
When using OpenCV in Python, it actually reads the image in a data structure called numpy.ndarray, numpy.ndarray is a powerful N-dimensional array object provided by NumPy which is the fundamental package for scientific computing with Python. NumPy is an extension of the Python language, support for multidimensional arrays and matrix operations, in addition to providing a large number of mathematical function libraries for array operations. Besides this, NumPy also contains other useful tools such as linear algebra, Fourier transform, random number capability and etc.

Standard Python uses a list to hold a set of values that can be used as an array, but since the elements of the list can be any object, the object in the list is a pointer to the object. So in order to save a simple [1, 2, 3], you need to have 3 pointers and three integer objects. For numerical operations, this structure is obviously a waste of memory and CPU computing time. The birth of NumPy makes up for these shortcomings. NumPy provides two basic objects: ndarray (N-dimensional array object) and ufunc (universal function object). Ndarray is a multidimensional array that stores a single data type, while ufunc is a function that can process arrays.

3.3 Concepts of visual information processing

We have introduced the overall process in the former section, it is apparent that the main work of this thesis is to process the grapevine images with users' mark, that is, between step 3 and step 4, after users submit the images and pixels information of each branch, we should process it properly in order to identify all the pixels in the same branch as well as related necessary structure information and give users the feedback, finally generate the structured representation of each grapevine based on the segmentation of branches.

Image is considered as one kind of visual information, and in fact visual information processing consists of several different subjects: image processing, image analysis, pattern recognition and computer vision. All of these subjects focus on different stages in the processing and they vary from their inputs and outputs, purposes and applicable algorithms, knowing which page we are on could help us to choose the appropriate approach to complete the relevant tasks. The table in Figure 3.6 shows the differences among these subjects, and the work in this thesis is basically included in the field of image processing and image analysis.



Area	Input	Output
Image processing	images	processed images
Image analysis	images	image descriptions
Pattern recognition	image descriptions	object classes
Computer vision	images	3D models

Figure 3.6: Different terms in visual information processing

Image analysis generally uses mathematical models combined with image processing techniques to analyze underlying features and superstructures to extract information with certain intelligence. Image processing, generally refers to digital image processing, a technique of analyzing an image with a computer to achieve the desired result. A digital image is a large two-dimensional array that is sampled and digitized by a digital camera, scanner, etc. The elements of the array are called pixels and have an integer value called a gray value. The main content of image processing technology includes image compression, enhancement and restoration, matching, description and recognition. Common processing includes image digitization, image encoding, image enhancement, image restoration, image segmentation, and image analysis.

Image analysis and image processing are closely related. There is a intersection between the two to a large extent, but they are different. Image processing focuses on research in signal processing, such as image contrast adjustment, image coding, denoising, and various filtering studies. However, image analysis is more focused on the content of images, including but not limited to the various techniques of image processing, which tend to analyze, interpret and identify image content. Therefore, image analysis and pattern recognition, computer vision in the field of computer science are more closely related. We can have a better understanding about the relations from Figure 3.7.

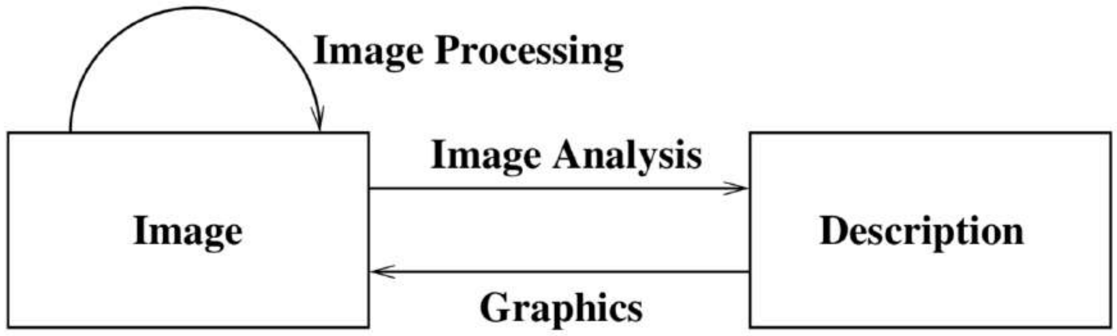


Figure 3.7: Relations among different terms

3.4 Image processing and analysis

3.4.1 Image scaling

The first action we need to perform after we get the images of grapevine is to scale those images, that is, resize the images to a more preferable size. In our case, there are two main reasons for us to do so. First reason is the images we took from the vineyard is too big (high resolution), the resolution of each image is 6000*4000, namely there are 24,000,000 pixels in one single image, it is too expensive to process image with such size since most of the image processing algorithms need to loop through each pixels, moreover if we consider a RGB image with 3 channels as well, the cost will even be triple, it could slow down the performance of our application considerably, moreover, higher resolution image require more storage space. Second reason we need to resize the image is because we also need to process the images from unknown users besides our own images, we can not demand everyone to upload images with a constant size, so we need to resize the images with various size to a consistent size in order to facilitate our further processing.

In computer image processing and computer graphics, image scaling refers to the process of adjusting the size of a digital image. Image scaling is a non-trivial process that requires a trade-off between processing efficiency and the smoothness and sharpness of the results.

In OpenCV, there are two ways to scale a image, one is using mathematical interpolation, another one is using image pyramid. But in image pyramid, it can only shrink or zoom an image proportionally, which means a M * N image will become M/2 * N/2 in case of shrinking and 2M * 2N in case of zooming. And if we want the size of image to be constant (in our case all the images will be resized to 600*400) before we apply further operation, image pyramid is not suitable for us because we can not ensure that any image we get will be a proportion of 600*400. Thus we choose to use mathematical interpolation to scale the images, this way is able to scale any image with any size to a designated size.

Image scaling can be interpreted as a form of image resampling or image reconstruction from the view of the Nyquist sampling theorem. According to the theorem, downsampling to a smaller image from a higher-resolution original can only be carried out after applying a suitable 2D anti-aliasing filter to prevent aliasing artifacts. The image is reduced to the information that can be carried by the smaller image. In the case of upsampling, a reconstruction filter takes the place of the anti-aliasing filter.

In the Python API of OpenCV, the corresponding function is :

$$dst = cv2.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])$$

This function resizes the image src down to or up to the specified size, the size and type are derived from the src, dsize, fx, and fy. All the parameters are as following:

- src - input image.
- dst - output image; it has the size dsize (when it is non-zero) or the size computed from src.size(), fx, and fy; the type of dst is the same as of src.
- dsize - output image size; if it equals zero, it is computed as:

  $$dsize = size(round(fx * src.cols), round(fy * src.rows))$$

  Either dsize or both fx and fy must be non-zero.
- fx - scale factor along the horizontal axis; when it equals 0, it is computed as $(double)dsize.width/src.cols$.
- fy - scale factor along the vertical axis; when it equals 0, it is computed as $(double)dsize.height/src.rows$.
- interpolation - interpolation method

Different interpolation methods are suitable for various occasions and have distinct cost, the interpolation method can be selected among these options:

- INTER_NEAREST - nearest neighbor interpolation
- INTER_LINEAR - bilinear interpolation (default option)
- INTER_CUBIC - bicubic interpolation
- INTER_AREA - resampling using pixel area relation
- INTER_LANCZOS4 - Lanczos interpolation over 8x8 neighborhood

For example, nearest neighbor interpolation is one of the simplest ways of increasing image size, replacing every pixel with the nearest pixel in the output, for upscaling this means multiple pixels of the same color; bilinear interpolation works by interpolating pixel color values, introducing a continuous transition into the output even where the original material has discrete transitions. Although this is desirable for continuous-tone images, this algorithm reduces contrast (sharp edges) in a way that may be undesirable for line art; bicubic interpolation yields substantially better results, with only a small increase in computational complexity.

Preferable interpolation methods are INTER_AREA for shrinking, while for zooming, INTER_CUBIC and INTER_LINEAR are better. So for the grapevine images we have, we use this function with INTER_AREA option to reduce each of them to 600*400, and for images uploaded by users, we reduce the size to 600*400 as well if the image is bigger than this designated size and vica versa. See an example of resized image in Figure 3.8, the original size of this image is 10.5MB while after resizing it becomes 451KB.

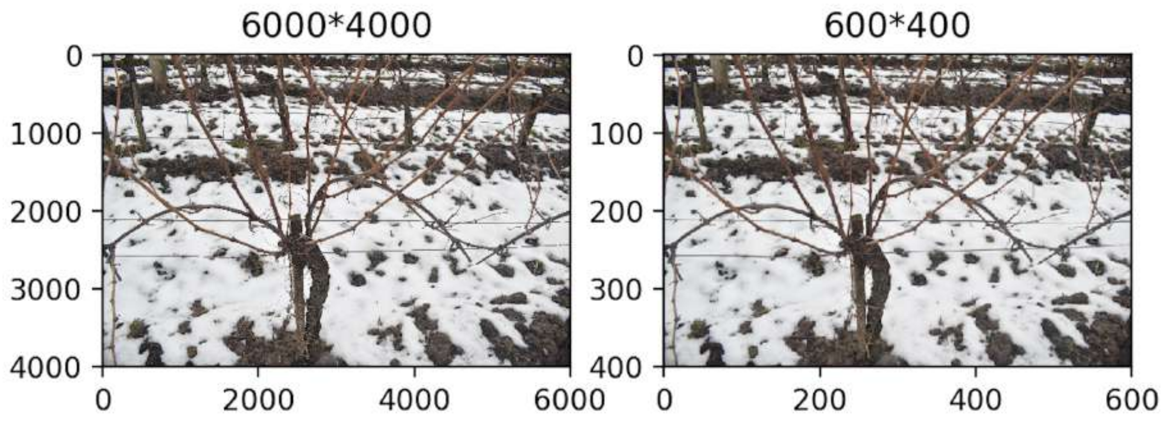


Figure 3.8: Example of resized image

3.4.2 Bilateral Filter

In the process of image acquisition, transmission and storage, the image is often degraded by various effects of noises, and the image preprocessing algorithm is directly related to the effect of subsequent image processing, such as image segmentation, recognition, edge extraction, etc., in order to obtain high-quality digital images, it is necessary to denoise the image to maintain the original information integrity (ie, the main features) while removing the useless information in the signal. Therefore, noise reduction processing has always been a hot spot in image processing and computer vision research.

The ultimate goal of image denoising is to improve a given image and solve the problem of image quality degradation due to noise interference. Through the denoising technology, the image quality can be effectively improved, the signal-to-noise ratio can be increased, and the information carried by the original image can be better reflected. As an important preprocessing method, the image denoising algorithm has been extensively studied. How to find a better balance point in resisting noise and retaining details has become the focus of research in recent years.

In fact there are various noises, therefore there are different algorithms based on diverse theories for different kinds of noises accordingly. Here in this thesis, we choose spatial filtering which is one of the most common and basic approach. Spatial filtering is performed by directly performing data operations on the original image to process the grayscale values of the pixels, besides, it is implemented by image convolution.

The digital image is a two-dimensional discrete signal. The convolution operation on the digital image is to use the convolution kernel (convolution mask) to slide on the image, and multiply the grayscale value of the pixel on the image with the corresponding value on the convolution kernel, and then all the multiplied values are added as the new grayscale value of the pixel on the center of convolution kernel, and finally sliding over all pixels on the image. Figure 3.9 is a classical illustration for one step in the process.

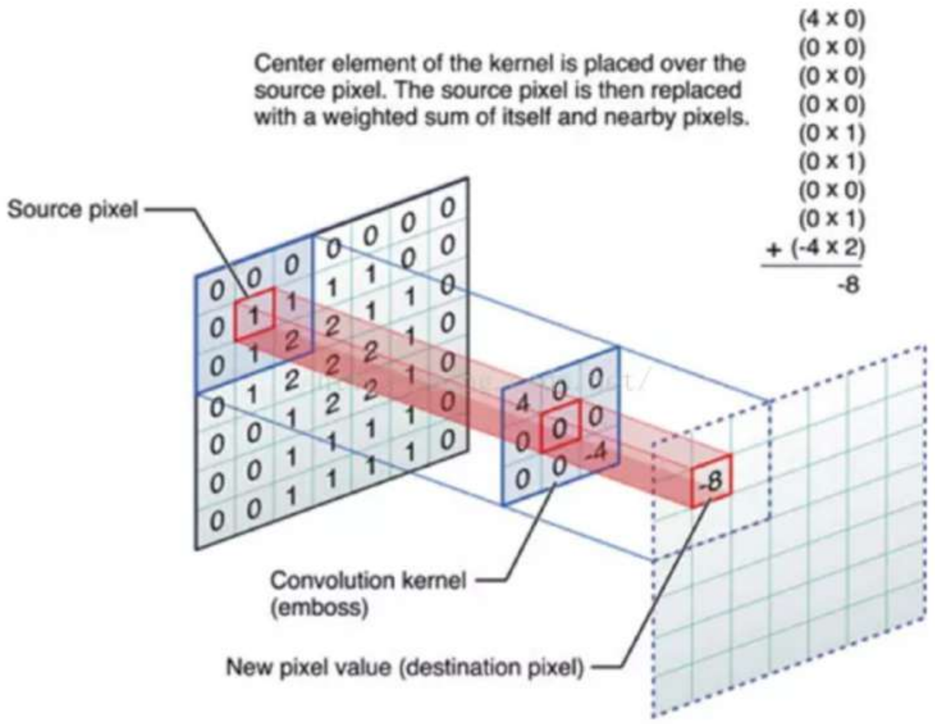

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Figure 3.9: A classical illustration of image convolution

It can be seen that an original image can be transformed into another image after being processed by a certain convolution kernel. For the filter, there are certain rules:

1. The size of the filter should be odd so that it has a center, such as 3x3, 5x5 or 7x7. There is a center, and there is also a radius, for example, the radius of a 5x5 core is 2.

2. The sum of all the elements of the convolution kernel should be equal to 1, in order to ensure that the brightness of the image before and after filtering remains unchanged. Of course, this is not a hard requirement.

3. If the sum of all elements of the convolution kernel is greater than 1, the filtered image will be brighter than the original image. Conversely, if it is less than 1, the

resulting image will be darker. If the sum is 0, the image will not turn black, but it will be very dark.

4. For filtered structures, negative numbers or values greater than 255 may occur. In this case, we can truncate them directly to 0 or 255. For negative numbers, absolute values can also be taken.

By giving different values to the convolution kernel, we can achieve different result. Usually the filter for noise reduction will also cause the loss of details in the image such as edges, because both noise and edge are considered as high frequency content in the image, so when we apply low-pass filter, both of them get eliminated. For example, mean filter and Gaussian filter will blur the edges in the image. But in the later processing, the edge information is important for us, so here we choose to user bilateral filter which was defined for, and is highly effective at noise removal while preserving edges.

Bilateral filtering is a nonlinear filter that achieves the effect of maintaining edges and smoothing noise. As with other filtering principles, the bilateral filtering also uses a weighted average method, it uses the weighted average of the luminance of neighbor pixels to represent the intensity of a pixel, and the weighted average is based on a Gaussian distribution. Most importantly, it also uses one more (multiplicative) Gaussian filter component which is a function of pixel intensity differences. The Gaussian function of space makes sure that only pixels that are 'spatial neighbors' are considered for filtering, while the Gaussian component applied in the intensity domain (a Gaussian function of intensity differences) ensures that only those pixels with intensities similar to that of the central pixel ('intensity neighbors') are included to compute the blurred intensity value.

Two weighted domains are considered simultaneously in bilateral filtering, the spatial domain and pixel value domain. The principle of bilateral filtering is a multiplication of a Gaussian function related to spatial distance and a Gaussian function related to grayscale distance. The Python API in OpenCV provides a bilateral filtering function:

$$dst = cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace[, dst[, borderType]])$$

The most important parameter there are sigmaColor and sigmaSpace, sigmaColor is filter sigma in the pixel value domain, a larger value of the parameter means that farther colors within the pixel neighborhood will be mixed together, resulting in larger areas of semi-equal color; sigmaSpace is filter sigma in the spatial domain, a larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough. In our case, both sigmaColor and sigmaSpace are 75, this is an empirical choice. The resulting images after applying normal Gaussian filtering and bilateral filtering is shown in Figure 3.10, we can see that both of them blurred the image in small high-detail regions in order to reduce noise, while the bilateral filtering did not blur the edge too much.

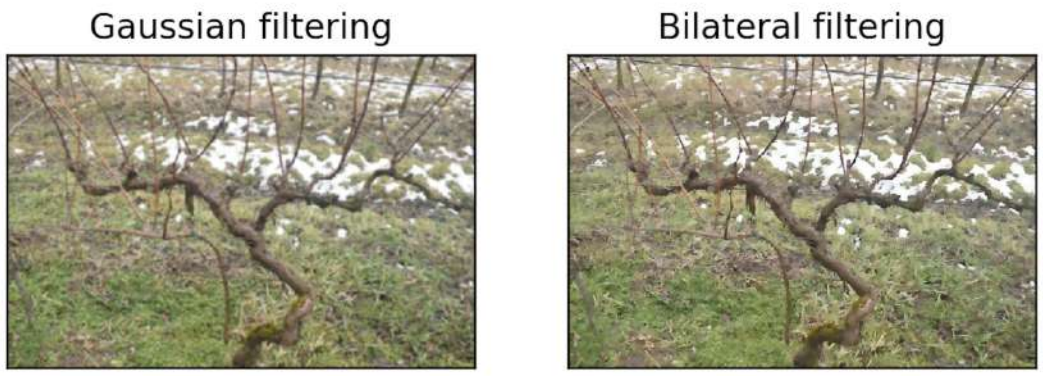


Figure 3.10: Comparison of two filtering

3.4.3 Region of interest

The last two sections were preprocessing of the entire grapevine image, the purpose is to transform an image to a better state for further operation, but from this section, we will focus on each branch in the images separately, this involves region of interest. In the field of image processing, a region of interest (ROI) is an image region selected from an image, which is the focus of your image analysis. Using the ROI to define the target you want to read can reduce processing time and increase accuracy. Usually ROI is a regular shape such as rectangle, circle, ellipse and etc.

So for each cane in the image, we can get the coordinates information from users' input, for example, see in Figure 3.11, red points in this image represent the user's mouse click along one targeted cane.

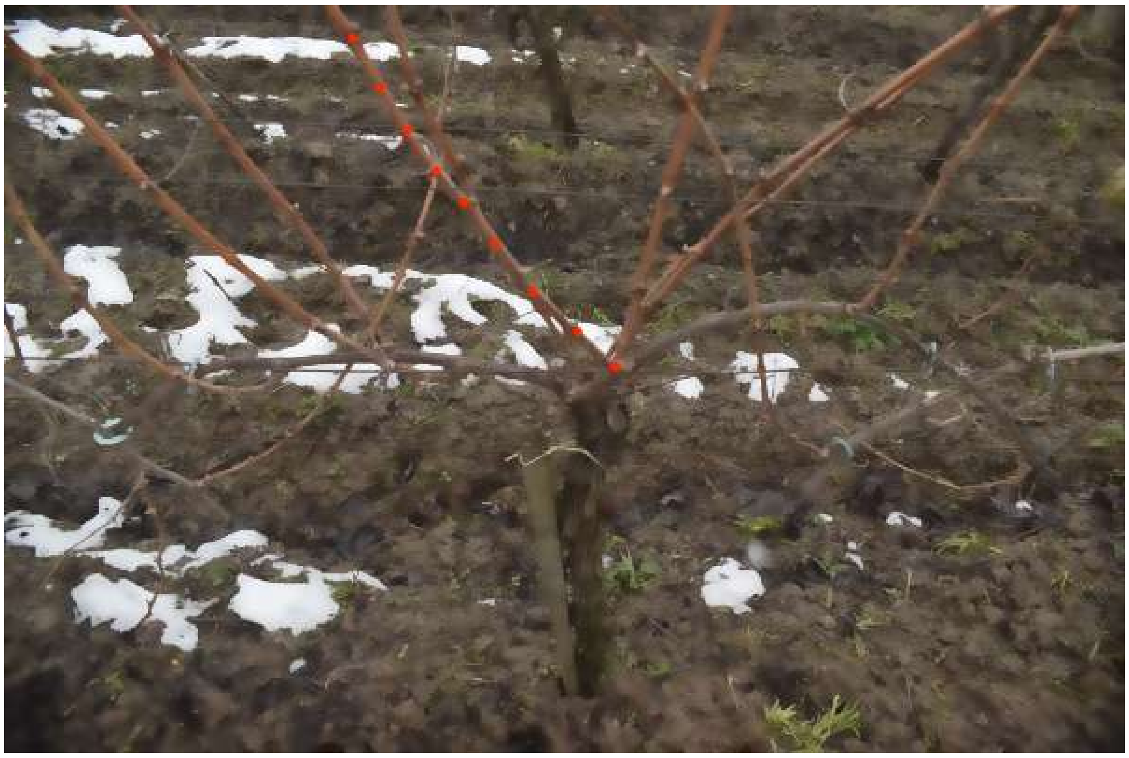


Figure 3.11: Example of one marked cane from user

As long as we get the coordinates information of all these red pixels, we can form a rectangle by finding 4 red pixels in the image. If we consider the image as a standard Cartesian coordinates system, so each pixel has its own coordinate (x, y) in this system, and lets assume the (x_t, y_t) is the coordinate for the topmost pixel, (x_d, y_d) is the downmost, (x_r, y_r) is the rightmost and (x_l, y_l) is the leftmost. The height of the ROI rectangle will be $y_t - y_d$ and the width of the ROI rectangle will be $x_r - x_l$.

Figure 3.12 shows the region of interest for the marked cane in Figure 3.11. Here the topmost pixel happens to be the leftmost and the downmost one happens to be the rightmost, so we calculate the width and height of this ROI based on the coordinates information of these two pixels.

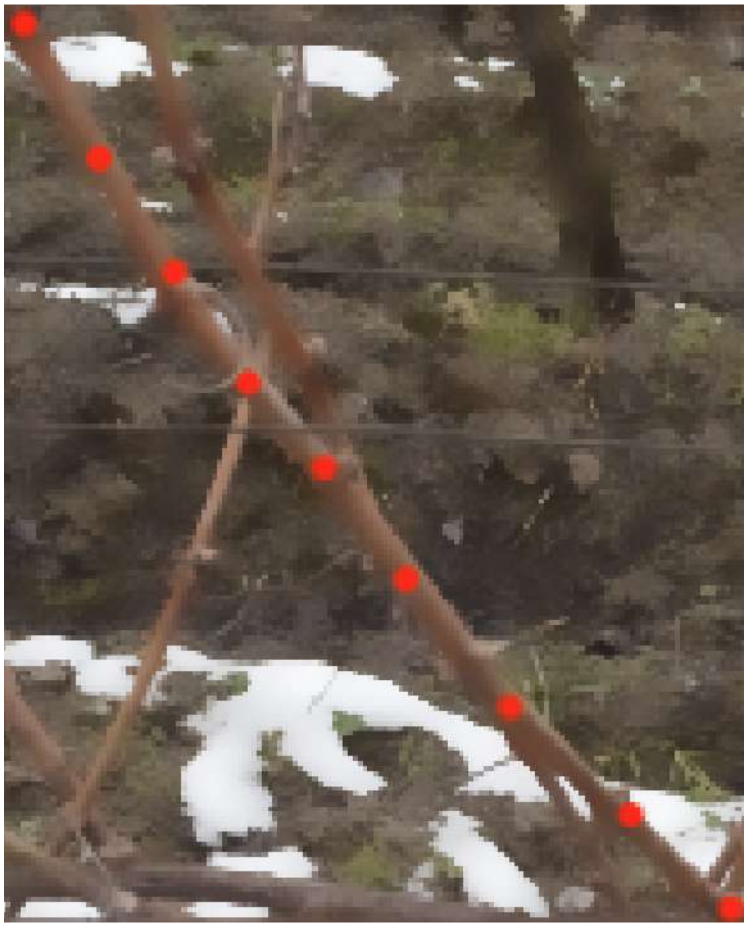


Figure 3.12: ROI for the marked cane

There can be one little problem when a cane is exactly in the vertical or horizontal direction with very small curvature, both situations we can get a very narrow ROI due to the width difference is so little in case of a vertical cane and height difference is so little in case of a horizontal cane, such narrow ROI would not be able to cover the region of the whole cane. In order to solve this issue, we can just add one more condition check by setting a threshold value, if the width or height we calculate as mentioned before is less than this threshold value, we just simply give it additional value. The exact value we should add depends on the situations, obvious a trunk needs more than a cane since trunk is usually thicker than cane.

Moreover, the ROI we get now is not good enough, because it still contains too much irrelevant information and even other untargeted branches, we want to erase them. So we will use a new approach to get a ROI as similar as possible with the targeted branch.

We achieve this by using a polyline to fit the targeted branch. After we get the marked points on one branch, we use $cv2.line()$ function to draw lines on the image, for example, if the user marked ten points $(p_1, p_2, ..., p_{10})$ on a single branch from one side to another side, we will draw line between p_1 and p_2, then line between p_2 and p_3, ..., finally line between p_9 and p_{10}. Thus a polyline is formed to approximate the shape of a targeted branch. Two things we need to care here is the color and width of all the lines we draw. The color can be selected randomly as long as the color doesn't appear in the image, we choose the color RGB(0,255,255), because such color is rare under natural environment, it can be identified easily later. The width of the line we draw should be wide enough to cover all the region of a branch, but at the same time the width of line should not be too wide, otherwise it will cover too much useless information.

Figure 3.13 shows an example of the polyline of a branch generated by all the points marked by user on the image.

After we draw the polyline on image, we can easily convert the image to a binary

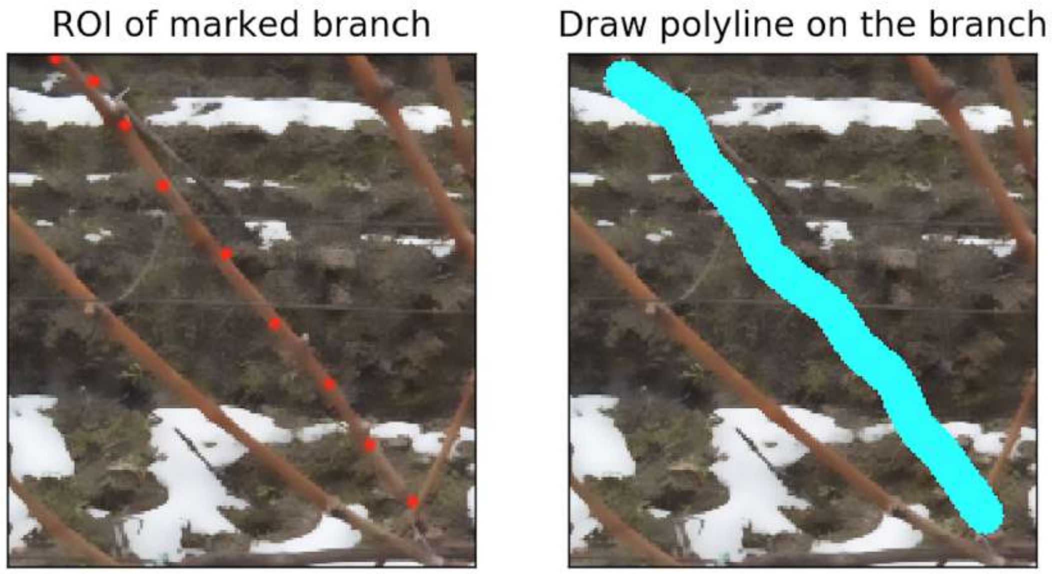


Figure 3.13: Polyline on ROI of marked cane

image where the region of polyline is white and black for the rest part, this is also why before we need to choose an unique color for the lines we drew. Then we use this binary image as mask, pass it to *cv2.bitwise_and*() function with the original ROI image, and we finally get a better ROI.

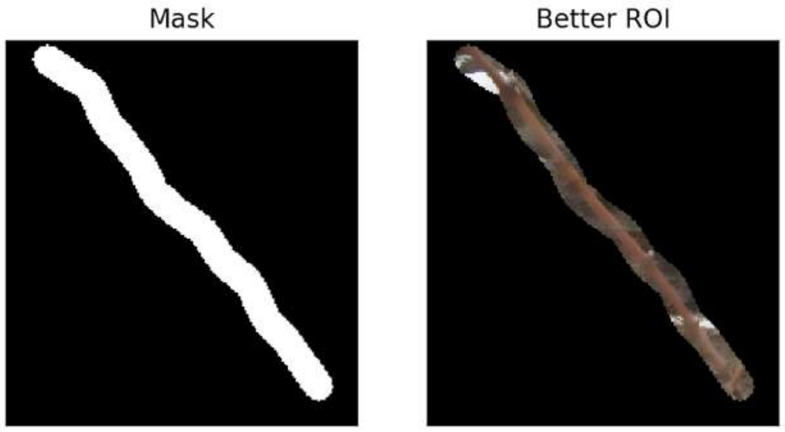


Figure 3.14: Mask and a better ROI

We can see in Figure 3.14 that most part of the original ROI is eliminated, therefore we can only focus on the rest part where we actually care.

3.4.4 Target object segmentation

There are four basic tasks in computer vision: a) image classification, b) object localization, c) object detection, d) object segmentation. Image classification describes the entire image by manual feature or feature learning method, and then uses the classifier to discriminate the object category; object localization not only needs to identify what is in the image, but also gives the position information of the target in the image, briefly speaking, a rectangular box is used to frame the identified objects; in object localization, there is usually only one or a fixed number of targets, and object detection is more general, and the types and numbers of targets appearing in the image are not fixed, therefore, target detection is a more challenging task than targeting; object segmentation is a more advanced task of target detection; target detection only needs to frame the bounding box of each target, while object segmentation needs to further determine which pixels in the image belong to which target.

From the previous section, we can get the improved ROI of each branch selected by users with very less background information, we can consider the task of object localization is done, now what we need to do is the object segmentation, that is, determine which pixels in the given ROI belong to the target branch. Image segmentation divides the image according to features such as grayscale, color, texture, and shape, the main methods of segmentation are:

1. Threshold-based segmentation - The basic idea of the threshold method is to calculate one or more gray scale thresholds based on the gray scale features of the image, and compare the gray scale value of each pixel in the image with the threshold, and finally divide the pixels into the appropriate categories according to the comparison result. Therefore, the most critical step of this type of method is to find the optimal gray threshold according to a certain criterion function.

2. Edge-based segmentation - The collection of consecutive pixels on the boundary line of two different regions in the image is a reflection of the local feature discontinuity of the image, which reflects the mutation of image characteristics such as gray scale, color and texture. In general, the edge-based segmentation method refers to edge detection based on gray values, which is based on the observation that the edge gray scale will exhibit step or roof type changes. There is a significant difference in the gray values of the pixels on both sides of the step edge, while the roof edge is located at the turning point where the gray value rises or falls. Based on this feature, edge detection can be performed using a differential operator, that is, the extremum of the first derivative and the zero-crossing of the second derivative are used to determine the edge. The specific implementation can be performed by image convolution.

3. Region-based segmentation - According to the similarity criterion of the image, it is divided into different region blocks. There are mainly seed region growth method, regional division and merger method, watershed method.

4. Segmentation based on graph theory - Such methods associate image segmentation problems with the min cut problem of the graph. First, the image is mapped to a weighted undirected graph. Each node in the graph corresponds to each pixel in the image. Each edge is connected with a pair of adjacent pixels. A segmentation of the image is a cut of the image, and each region that is segmented corresponds to a subgraph in the image. The optimal principle of segmentation is to make the divided subgraphs maintain the greatest similarity internally, while the similarity between subgraphs is kept to a minimum. The essence of the graph-based segmentation method is to remove a specific edge and divide the graph into several subgraphs to achieve segmentation. The graph theory-based methods currently known include GraphCut, GrabCut, and Random Walk.

5. Segmentation based on energy functional - This kind of method mainly refers to the active contour model and the algorithm developed on it. The basic idea is to use continuous curve to express the target edge and define an energy functional such that its independent variable includes the edge curve. Therefore, the process of segmentation is transformed into the process of solving the minimum value of the energy functional, which can generally be achieved by solving the Euler equation corresponding to the function. The position of the curve when the energy reaches the minimum is the contour of the target.

Each of these 5 mentioned methods has several specific implementations based on different algorithms, but even though there are so many segmentation algorithms, finding the one which is suitable for our case is still very difficult. For example, both threshold-based and edge-based segmentation totally rely on the gray scale value of pixels in the image, if gray scale value of some pixels are so close or even the same, they would be categorized to a same region in case of threshold-based segmentation or can not be detected as the edge in case of edge-based segmentation. Due to the complex background and light condition in our images, the gray scale value of some pixels belong to the region of a branch can be very close or even the same with some other pixels belong to the background, so it is impossible to find a proper threshold to segment these pixels or detect the edge between these pixels if they are adjacent. Moreover, in case of region-based segmentation and segmentation based on graph theory, both of them require user interaction at certain extent, this hard requirement limits us, because once we get the grapevine image and coordinates information of each branch at the back-end of our application, we are no longer able to get any additional user interaction there, thus these methods become inapplicable as well. After some experiments of different algorithms, we found out that the threshold-based segmentation under HSV color space can give us better result.

Usually the images on the screen are represented by RGB color space, RGB color space is derived from a color TV using a cathode ray tube, and RGB represents three primary colors (R-red, G-green, B-blue), and the specific color values are superposed by these three primary colors. In our case, when we want to extract a targeted cane from the ROI, this cane should have a consistent color since it grows in the same year, but in fact some parts of the cane is darker while some other parts are brighter due to the light condition when we took photo, so the color on a branch can be very different in the RGB color space.

But there are other color spaces which are better than RGB color space to represent a color, for example the HSV color space, where H is hue, S is saturation, and V is value. The HSV color space separates the color information from the brightness information and places them in different channels, reducing the effect of light on specific color recognition. The HSV color model is in Figure 3.15.

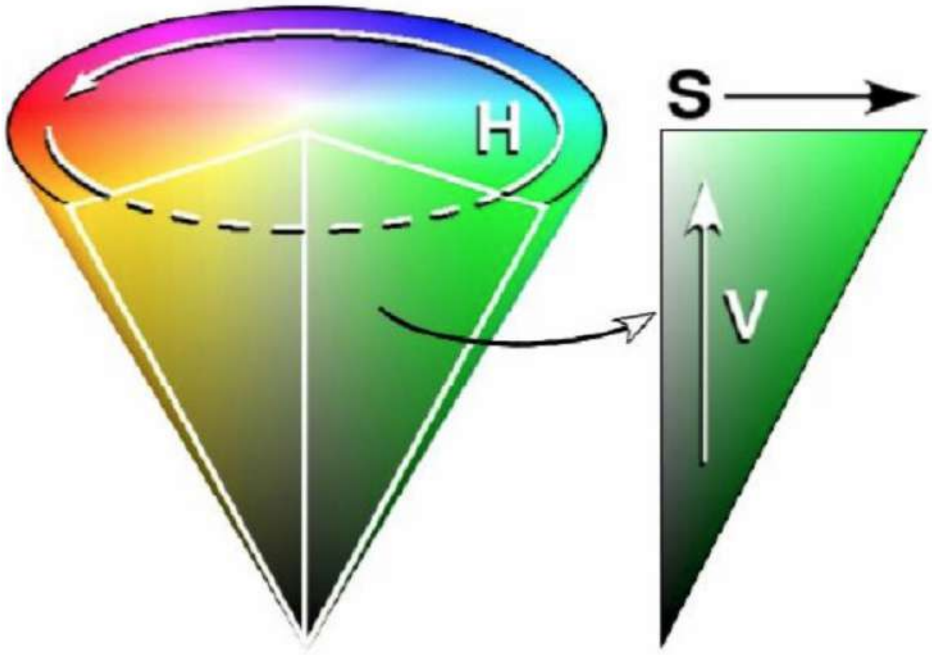


Figure 3.15: HSV color model

- The hue is measured by the angle, and the value ranges from 0° to 360° (in

OpenCV it is 180°). It is calculated from the red counterclockwise direction, red is 0°, green is 120°, and blue is 240°.

- Saturation indicates the extent to which the color is close to the spectral color. A color that can be thought of as the result of mixing a certain spectral color with white. The greater the proportion of the spectral color, the higher the color is close to the spectral color, and the higher the saturation of the color. If the saturation is high, then the color is deep and colorful. If the white color component of the spectral color is 0 so the saturation is the highest. Usually, the saturation ranges from 0% to 100%, the larger the saturation value, the more saturated the color.

- The value indicates the degree of color brightness, for the light source color, the brightness value is related to the brightness of the illuminant; for the object color, this value is related to the transmittance or reflectance of the object. Usually the value ranges from 0% (black) to 100% (white).

For extracting the branch, we should find the correct range in the HSV color space. And there is conversion formula to convert RGB value to HSV value, after we get coordinate of each pixel marked by users, we can get the RGB value of each pixel as well, and then we can convert the RGB value to HSV value accordingly. But since the branches in images are very thin, it is possible that some points marked by a user are out of the region of a branch, this can cause some RGB values we get are not the actual value of pixels within the region of a branch, we can get a bad result if we use these wrong values. Fortunately, we will get not only one or two pixels from one single branch, but several ones, so the error ones are minor and correct ones are the majority, therefore we can analyze all the values together and find the most likely range in HSV color space. See two examples of extraction result in Figure 3.16.

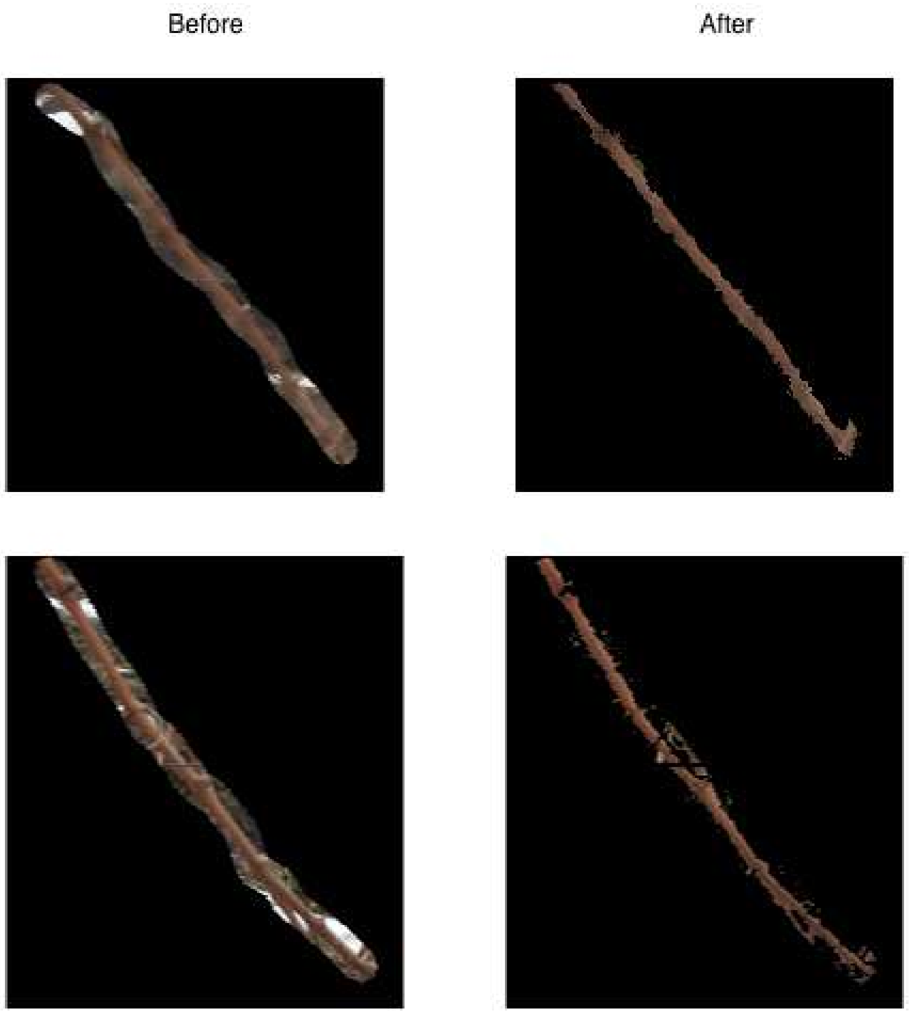


Figure 3.16: Two examples of extraction result

3.4.5 Morphological operation

The results we got from previous stage contain some isolated pixels and also discontinuity caused by the iron wire which crosses in front of the branch. Here in this stage, we can apply mathematical morphological transformations on the images to get a better result.

The term morphology usually refers to a branch of biology that studies the morphology and structure of plants and animals. The morphology in image processing often indicates mathematical morphology, mathematical morphology is a discipline of image analysis based on lattice theory and topology. Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. There are four most common morphological operators: dilation, erosion, opening and closing.

Dilation or erosion is the convolution of an image with a structuring element. The structuring element can be any shape and size, it has a separately defined reference point, which we call an anchorpoint. In most cases, the structuring element is a small solid square or circle with a reference point in the middle. In fact, we can think of the structuring element as a template or a mask. The dilation is the operation of finding the local maximum. The structuring element is convolved with the image, that is, find the maximum value of the pixel of the area covered by the structuring element, then the maximum value is assigned to the pixel specified by the anchorpoint. This will cause the highlighted area in the image to grow gradually. And erosion is just the opposite of dilation.

Erosion erodes away the boundaries of foreground object, all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small isolated points. However dilation increases the foreground object region in the image, opening is just another name of erosion followed by dilation, and closing is reverse of opening, dilation followed by erosion. The opening operation generally smoothes the contour of the object, breaks the narrow discontinuities and eliminates the fine protrusions, and the closing operation also makes the contour smoother, but contrary to the opening operation, it usually eliminates small holes and fills the gaps in the image. In Figure 3.17 we can see that the opening operation reaches better result, but still left some discontinuity generated by the iron wires.

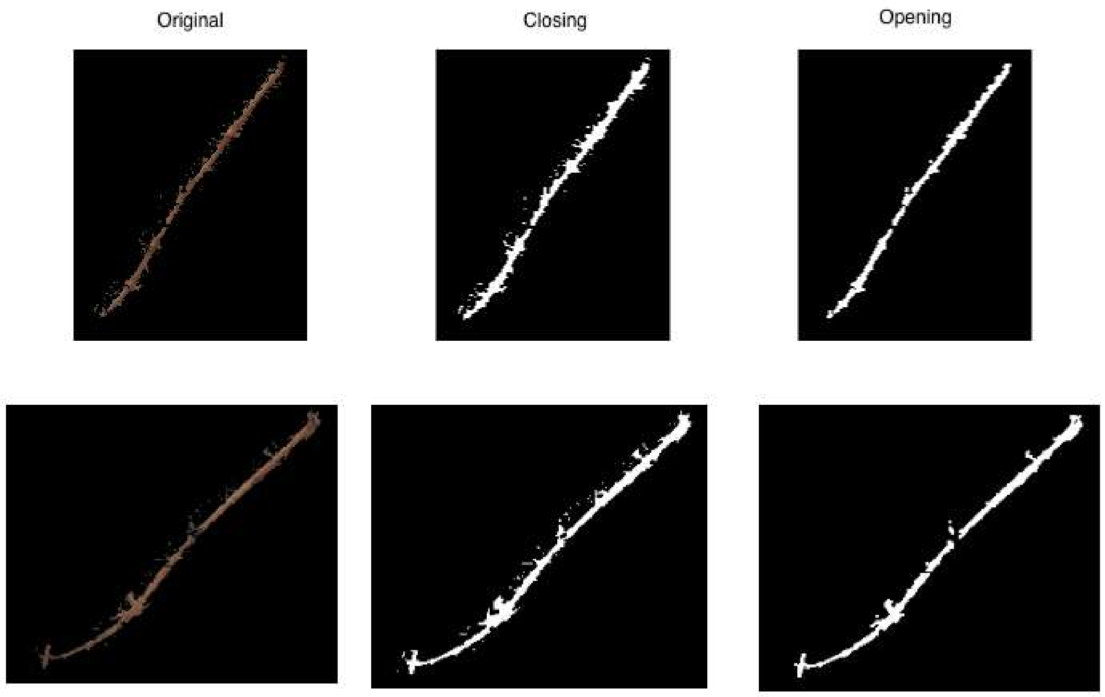


Figure 3.17: Examples of morphological operation

The result of opening operation contains much more less isolated regions which obviously do not belong to the region of branch, this is because the order of erosion and dilation. When we apply opening operation, erosion comes first, all the small isolated white regions are eroded away completely, so the consequent dilation can not restore those isolated pixels which are deleted in the previous operation. But in the case of closing operation, dilation comes first, it dilates those isolated white regions as well, so some relatively bigger isolated regions can not be completely deleted even the erosion operation is applied after.

Now we can easily notice that all the white pixels in the binary image after opening operation are the pixels which actually belong to the region of branch, so we can get the coordinates of these white pixels in the ROI of each branch. The last step is to shift these coordinates from ROI of each branch to the original images, these coordinates we get currently are relative information to each branch, if we want to get the absolute coordinates of each pixel in the original image, we can do it by finding the coordinate of pixel which locates in the top left corner of each ROI, and then add the coordinates value of this pixel to each relative coordinates we have respectively.

Chapter 4

Conclusion

4.1 Result

After chapter 3, we are able to get almost all the pixels within the region of each branch, so based on this result, we can also reconstruct each grapevine image where each branch in the image is identified separately by the absolute coordinates information. At the end, we can generate structured representation for each grapevine on images and present it in the application.

In general, this thesis proposed a novel approach to collect grapevine images and to recognize the structure of grapevine on images under any background with the assistance of some user interactions and a series of image processing algorithms. Such approach does not require high quality images which are relatively difficult to obtain and can be generalized easily. Also it helps to represent grapevine on images by structured languages such as XML or JSON since all branches are extracted individually, the structured representation can directly serve as source data for the decision module training of any AI technique. Moreover, this thesis summarized almost all the existing approaches and opinions in the related field, and also addressed some difficulties and traps of the study on grapevine pruning automation.

4.2 Future work

This thesis is just a preliminary research for a new autonomous grapevine pruning system, and only focus on the back-end image processing in the application, the front-end work is done by other people in our group, there is still a long way ahead to walk through.

Our proposed approach is not perfect and robust enough, there are still several drawbacks in each step of our approach. For example, all the image processing algorithms we chose may not be proper under some special situations, and the result of our approach is actually highly effected by the pixel information chosen by users, wrong click on the images could impede all the subsequent procedures. In the future we should enable our application to have higher fault tolerance to the wrong click information of users, it can be achieved both in the front-end and back-end. One solution of this problem is our application should be able to let user zoom the input image in order to get bigger space to click on each branch, thus we can avoid the wrong click from the user input, but such way should answer the question of how to get correct coordinates information of each pixel from the zoomed image. Another problem is the pixels of each branch we got are not actually accurate enough, some pixels are missed due to the discontinuity caused by the iron wire, while some other pixels which do not belong to a branch are considered to be one part of the branch due to the false segmentation or morphological operation, besides it is hard to develop a benchmark to measure the accuracy. Also, the generation of structured representation of each grapevine images should be implemented accordingly. All of these problems addressed above should be solved and our application should be improved and complete in the future.

Bibliography

[1] Tom Botterill, Scott Paulin, Richard Green. A Robot System for Pruning Grape Vines, *Journal of Field Robotics*, 2016, 34(6).

[2] Ming Gao, Tien-Fu Lu. Image Processing and Analysis for Autonomous Grapevine Pruning, *International Conference on Mechatronics and Automation*, 2006.

[3] He Leiying, Wu Chuanyu, Du Xiaoqiang. Fruit tree extraction based on simultaneous tracking of two edges for 3D reconstruction, *Transactions of the CSAE*, 2014, 30(7): 182-189.

[4] Ben Boudaoud L., Sider A., Tari A.. A new thinning algorithm for binary images, *2015 3rd International Conference on Control, Engineering and Information Technology (CEIT)*.

[5] Abu-Ain W., Abdullah S. N. H. S., Bataineh B., Abu-Ain T., Omar, K.. Skeletonization Algorithm for Binary Images, *Procedia Technology*, 2013, 11: 704–709.

[6] Bo Xiang, Xiaopeng Z., Wei Ma, Hongbin Z.. Skeletonization of Branched Volume by Shape Decomposition, 2006.

[7] H. B. Yan, S. M. Hu, R. Martin, and Y. L. Yang. Shape deformation using a skeleton to drive simplex transformations, *IEEE Transaction on Visualization and Computer Graphics*, 2008, 14(3): 693-706.

[8] Song Renjie, Liu Chao, Wang Baojun. Adaptive Canny edege detection algorithm, *Journal of Nanjing University of Posts and Telecommunications (Natural Science Edition)*, 2018, 38(3).

[9] M. Wan, Z. Liang, Q. Ke. Automatic centerline extraction for virtual colonoscopy, *IEEE Transactions on Medical Imaging*, 2002, 21(12): pp.1450-1460.

[10] Saheba S. M., Upadhyaya T. K., Sharma R. K.. Lunar surface crater topology generation using adaptive edge detection algorithm, *IET Image Processing*, 2016, 10(9): 657-661.

[11] WANG Hongshen, ZHANG Xiangyu, DOU Yongkun. Edge connectivity evaluation algorithm for performance of edge detectors, *Computer Engineering and Applications*, 2018, 54(16): 192-196.

[12] Daniel W., Patrick H.. Pose-Configurable Generic Tracking of Elongated Objects, *2013 IEEE International Conference on Computer Vision*.

[13] Joes Staal, Michael D. A., Max A. V., Bram V. G.. Ridge-Based Vessel Segmentation in Color Images of the Retina, *IEEE TRANSACTIONS ON MEDICAL IMAGING*, 2004, 23(4).

[14] Maria P. D., Christian C., Borja M., Pilar B., Constantino V., Javier T.. Grapevine Yield and Leaf Area Estimation Using Supervised Classification Methodology on RGB Images Taken under Field Conditions, *Sensores*, 2012, 12: 16988-17006.

[15] Mehmet Sezgin, Bulent Sankur. Survey over image thresholding techniques and quantitative performance evaluation, *Journal of Electronic Imaging*, 2004, 13(1):146.