Multimedia Application Final Project

June 10, 2023

Yuze LIU
Yawen SHEN
Qianqiu SHI
Qiyao XU
Shuyuan ZHENG

Contents

Introduction

The C++ programming language holds significant importance in the realm of modern software development. As an object-oriented language, C++ inherits the efficiency and portability of C while introducing additional functionalities, making it a powerful and flexible tool. It finds extensive application in various domains, including system-level programming, game development, graphic image processing, and scientific computing.

OpenCV is written primarily in C++ and provides a C++ interface. This means that we can use C++ to take advantage of the features and capabilities offered by OpenCV. C++ provides the programming language framework for creating applications, while OpenCV provides the computer vision algorithms and tools for processing and analyzing images and video.

Throughout the Multimedia course this semester, we had the opportunity to explore C++ as a programming language. By combining C++ and OpenCV, we can harness the power and efficiency of C++ programming to implement computer vision and image processing tasks. We can also leverage the functionality and algorithms provided by OpenCV in C++ code to perform tasks such as image filtering, feature extraction, object recognition, and more. This combination allows for the development of powerful and effective applications in a variety of areas that require computer vision and image processing.

Function to use in the project

1. Dilatation / Erosion

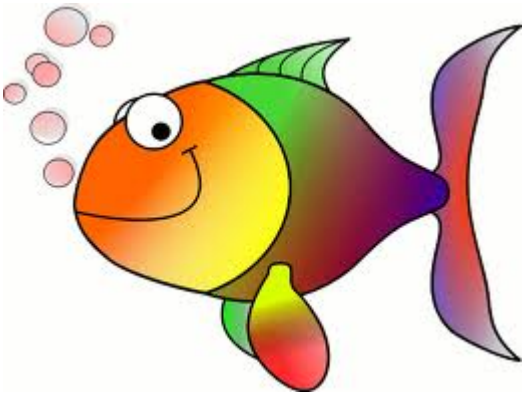

Image1. The Orignal Image

Image 2. The result of erosion(left) and dilatation(right)

Dilatation and erosion are fundamental operations in the field of image processing. These operations are typically used to manipulate the shape and structure of binary images or grayscale images.

- Dilatation: It expands the boundaries or regions of an image, making the white regions larger.
- Erosion: It shrinks the boundaries or regions of an image, making the white regions smaller.

These operations are used for tasks such as changing the shape of an image, removing noise, and connecting lines. OpenCV provides C++ functions to implement these operations, making them convenient for computer vision and image processing applications.

The provided program begins by including necessary header files and namespaces for the standard library and OpenCV. It then defines a function named performOperation that performs the image erosion and dilatation operations based on user input.

“if (inputImage.empty())”: This conditional judgment checks if the input image was successfully loaded. If the image load fails (i.e. inputImage is empty), an error message is output and returned.
“cv::dilate(inputImage, outputImage, structuringElement)”: This function is used to perform an inflation operation on the input image.
“cv::erode(inputImage, outputImage, structuringElement)”: this function is used to corrupt the input image.

It first loads the input image and then creates the appropriate structuring element based on the user-selected operation type and erosion size. Next, the input image is expanded or corrupted according to the user-selected operation type (input 1 or 2 for selection) and the result is stored in the output image. Finally, the output image is displayed in a window using OpenCV's functions.

2. Resizing

A common image resizing method is the interpolation method, which infers the value of the target pixel based on the relationship between known pixels in the image. Common interpolation methods include nearest neighbor interpolation, bilinear interpolation, bicubic interpolation, etc.

- INTER_NEAREST: Nearest Neighbor Interpolation. This method selects the value of the source image pixel nearest to the target pixel as the value of the target pixel. The calculation is simple and fast, but it may lead to a jagged effect on the image edges.
- INTER_LINEAR: Bilinear interpolation. This method considers the four closest pixels around the target pixel and obtains the value of the target pixel by a weighted average based on its position and grayscale value.
- INTER_CUBIC: Bicubic interpolation. This method calculates the value of the target pixel by selecting more nearest neighbor pixels and using three times spline interpolation.
- INTER_AREA: resampling using pixel area relation. this method works better when the image is reduced and determines the value of the target pixel by calculating the average over the area occupied by the target pixel.

Input image path, and width and height:

```
Enter the image file path: ../image/lena.png
Enter the width value: 200
Enter the height value: 280
```

Image 3. Output command line for resizing

During resizing, we use bilinear interpolation (INTER_LINEAR) as the default interpolation method to balance the need for speed and smoothness, which is a good choice in most cases. However, depending on the specific application scenario and requirements, other interpolation methods can be considered to balance the image detail conservation and execution speed.

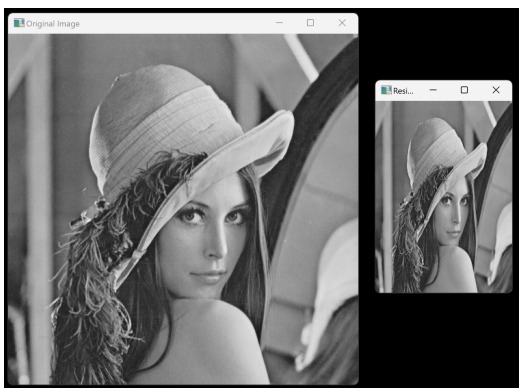

Image 4. The original image(left) and the resizing image (right)

3. Lighten / Darken

We need to adjust the brightness and darkness of an image. This can be done without using OpenCV's proprietary functions. The main thing we need to use is a formula.

$$g(x,y) = \alpha f(x,y) + \beta$$

$g(x,y)$ is an image that has been adjusted.
$f(x,y)$ is original image.
α is the factor to adjust the image contrast.
β is the factor that adjusts the image brightness.
x and y denote pixels in row y and column x

Contrast (α): Usually between 0.0 and 2.0, where 1.0 indicates the contrast of the original image. when α <1, the image contrast decreases and the image becomes 'darker', and α > 1, the image will be more vivid.

Brightness (β): usually between -100 and 100, where 0 indicates the brightness of the original image.

The user can modify the contrast and brightness of the image by modifying the α and β coefficients.

First, we need the user to upload the image they want to change the brightness of and store it in a Mat object in OpenCV. And a new Mat object is created to store the images according to the user changes. And in order to be able to perform this formula, we need to make every pixel accessible. And because the image will have an RGB image so each pixel will have 3 values (R, G, B).

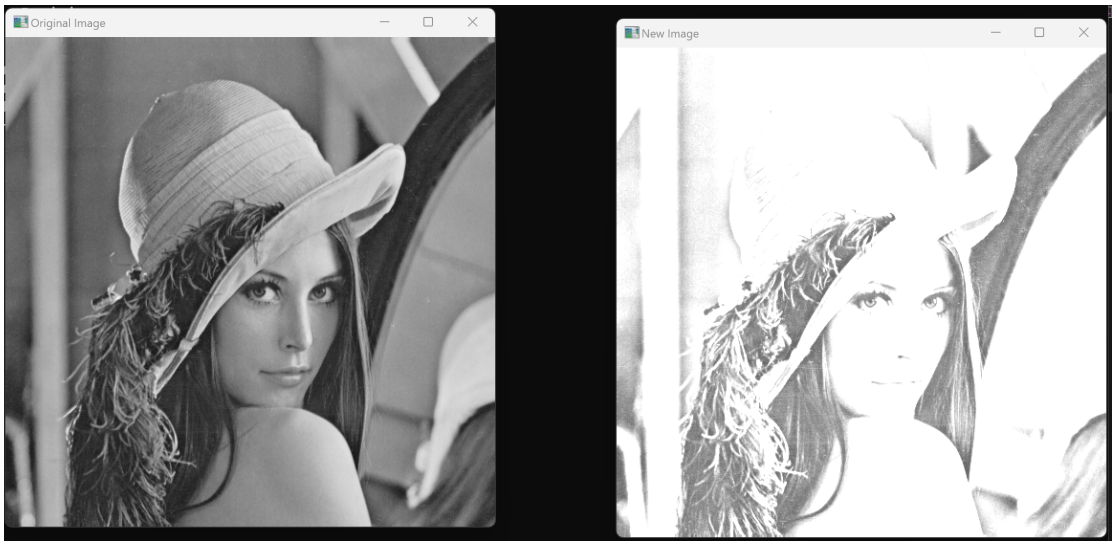

Image 5. The original image(left) and the lighten image (right)

4. Panorama / Stitching

Panorama stitching combines overlapping images to create a wide-angle panoramic view. It involves aligning the images, blending them together, and correcting color variations. The result is a seamless and immersive image capturing a broader scene. Panorama stitching is used in photography, virtual reality, maps, and other applications requiring a panoramic view.

The provided program demonstrates the process of image stitching using the OpenCV library. Image stitching refers to the task of combining multiple overlapping images into a single panoramic image.

To implement this function for image stitching, 2 functions are used: the "stitchImagesInDirectory" function and the "stitchImages" function.

"cv::glob(directoryPath, imageNames)": this function gets a list of the image filenames in the directory given the directory path. It uses wildcards to match the file names.
"images.push_back(image)": this line of code adds the successfully loaded images to the images vector for subsequent stitching.
"cv::Stitcher::create()": this function creates a cv::Stitcher object to perform the image stitching operation.
"stitcher->stitch(images, result)": this method uses the cv::Stitcher object to perform a stitching operation on the input images. the images vector contains the images to be stitched and result is the output stitched result image.
"cv::Stitcher::Status": This enumeration type represents the status of the image stitching operation. You can determine whether an operation was successful by checking the return status of the stitching operation.

In summary, this program provides a convenient way to stitch multiple images together to create a panoramic image. It utilizes the OpenCV library's `cv::Stitcher` class to perform the stitching operation. It first loads the image files in the given path and then stitches these images using the cv::Stitcher object. Upon successful stitching, the resulting image is saved to the specified path.

Remark:

The images to be stitched must have sufficient overlap. Image stitching requires significant overlap between adjacent images in order to align and blend accurately. If the overlap of the images is low or there are no common features, the stitching process may fail.

Image 6. Photos to be stitched

Image 7. Result of stitching

5. Canny edge detection

Canny's goal is to find an optimal edge detection algorithm, optimal in the sense that:

1. Good detection: That's means we need to be able to identify as many actual edges in the image as possible by the canny algorithm.
2. Good localization: The edges to be identified by the Canny algorithm should be as close as possible to the actual edges in the image.
3. Minimum response: The edges in the image can be flagged only once and possible image noise is not due to that flag as an edge.

The Canny algorithm edge detection is applied by the following steps:

1. Some parameters and variables are defined, including the hysteresis threshold and noise reduction kernel value, etc.

2. A callback function 'callBackCanny' is defined, which is used to implement the core logic of Canny edge detection. In the callback function, we first read the color image and create a matrix of the same type and size as it.

3. Convert the color image to grayscale image and perform noise reduction on the grayscale image using Gaussian filter.

4. Edge detection is performed on the noise reduced image using Canny operator and thresholding is done according to the hysteresis threshold.

5. After getting the edge image, we extract the edge regions from the original color image and display them in the window.

6. We provide the adjustable parameters, including kernel values and thresholds, by creating progress bars so that we can adjust the parameters in real time and observe the effect of Canny edge detection.

When running the program, we pass the input image file path to the Canny function, which opens a GUI window showing the original image and a preview of the Canny edge detection results. The user can change the kernel value and threshold value by adjusting the progress bar to observe the edge detection effect with different parameters.

When adjusting the parameters, we can observe the following phenomena:

- An increase of the kernel value causes the edges to become bolder, and the edge detection results may become smoother.
- The increase of the threshold value will cause more edges to be detected, but it may also lead to the increase of noise and spurious edges.

By gradually adjusting the parameters, we can find the best combination of kernel and threshold values to obtain clear and exact edge detection results.

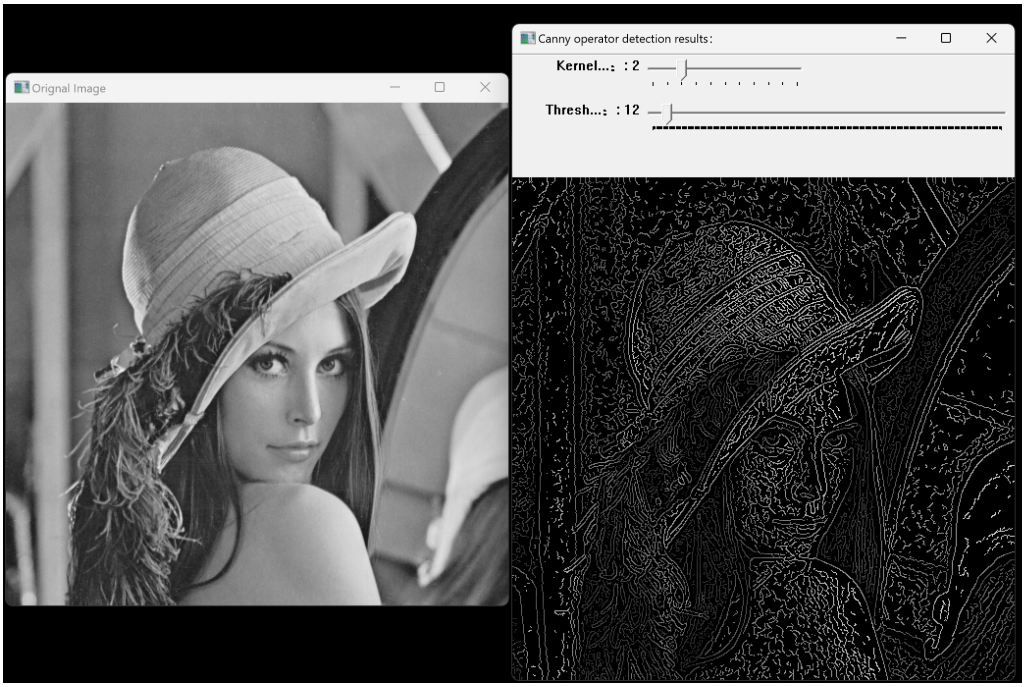

Image 8. The original image(left) and the result image (right)

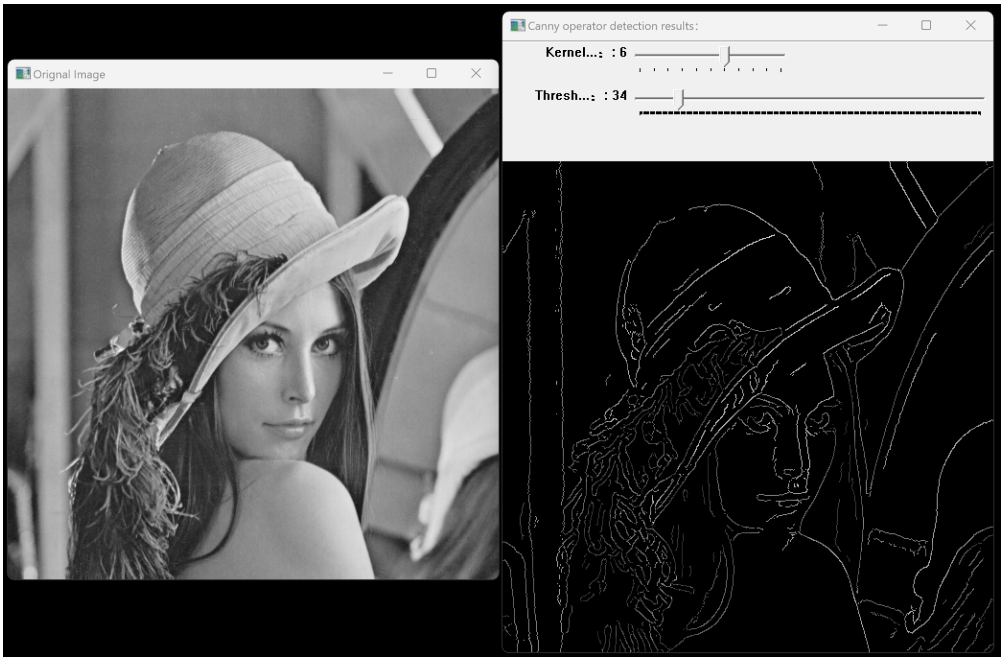

Image 9. The original image(left) and the result image (right)

Interface

For the third part of our development, we chose Qt, a cross-platform C++ GUI framework, and OpenCV, an open source library for developing real-time image processing, computer vision, and pattern recognition programs, as our main technologies and tools.

Members of our team were responsible for developing the image editor with an interface, and we believe that an image editor without an interface is imperfect. Therefore, we used the Qt framework to build a user interface that allows users to use and manipulate the image editor more easily. However, it can be noticed that the interface code in the third part is different from the code in the second part. The result of our group is divided into two main parts, one is the image editor in command line form, and the other is the image editor with interface that I made using QT

The results of our team's work includes an image editor in command line form and an image editor with interface. By combining image processing functions with the user interface, we hope to provide users with a more user-friendly and convenient image processing experience.

I. Function and Structure in QT

- **MainWindow constructor**

The MainWindow constructor is responsible for creating the user interface of the application. It first calls ui->setupUi(this) to set up the user interface. Then, it creates and configures menus and menu items, and sets up the connection of signals and slots to perform appropriate actions when the user interacts. Next, it creates and configures some controls, such as a label (QLabel) and a spin box (QSpinBox), and adds them to the vertical layout. Finally, it creates a graphical view and adds it to the layout and sets the layout as the central part of the main window.

- **MainWindow destructor**

The MainWindow destructor frees the memory dynamically allocated in the constructor, i.e., the UI object.

- **The openPicture function**

The openPicture function gets the path of the file selected by the user through the QFileDialog, then loads and displays the image. If the user cancels the dialog or selects a file that is not a valid image file, this function will not do anything. This application can open single and multiple images. The image file paths are obtained through Qt's QFileDialog class. When one or more image files are selected, these images are loaded into the QImage object and displayed using the QGraphicsPixmapItem object. When multiple images are loaded, each image is displayed in the QGraphicsScene scene with

a certain spacing between two adjacent images.

When we need to open a single picture, we just need to click File in the upper left corner of the application, then click Open Picture, and then we can find the picture by file path or several clicks, and finally click Open to load the picture into the application.

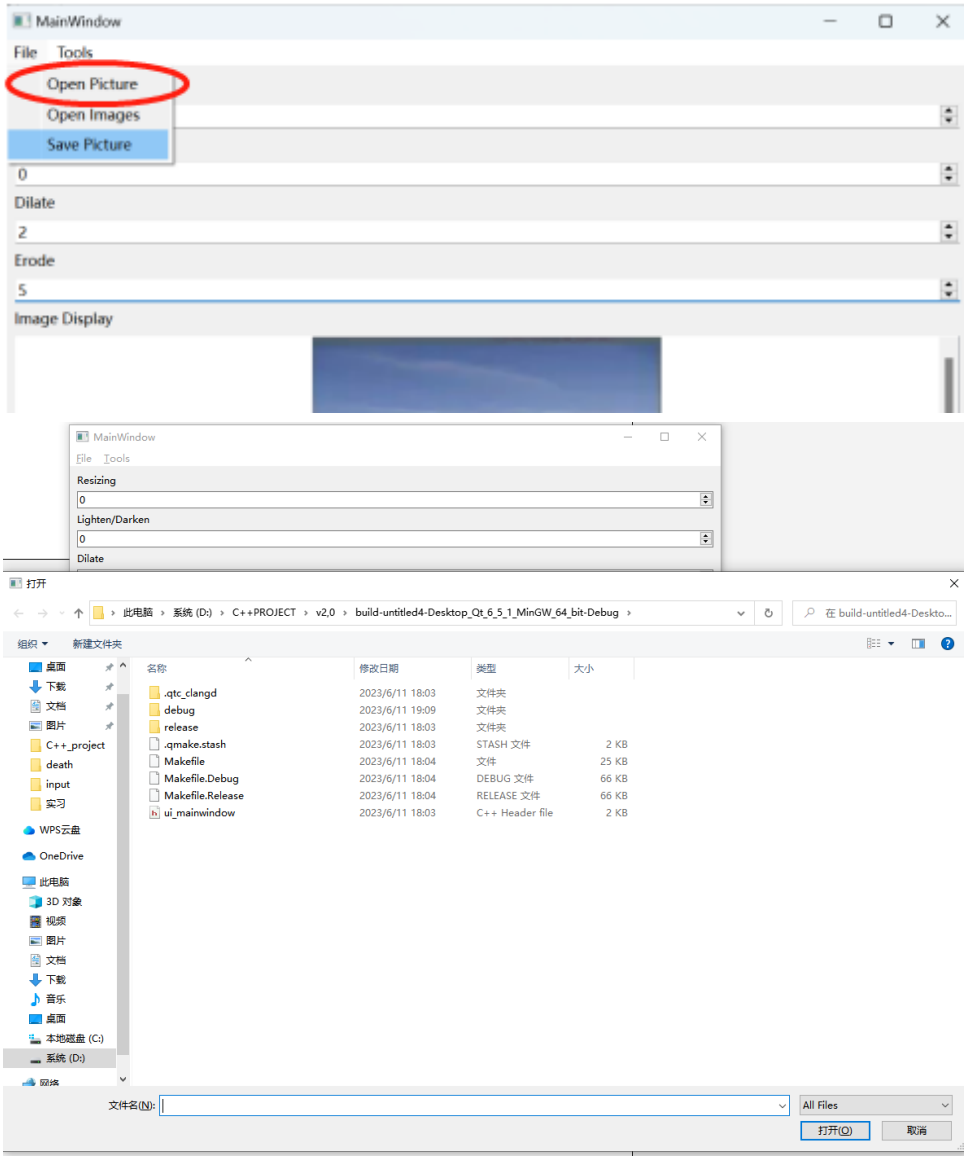

Image 10. Step of open single picture

The openImages function

The openImages function is like the openPicture function, but it allows the user to select multiple files. This function first clears the current list of images, then for each file selected by the user, it will load the image and add it to the image list and graphics scene.

When we need to open multiple photos, we need to click File in the upper left corner of the application, then click Open Images, then we can load a photo into the application in the same way as above, and then we can add multiple photos into the application by the same steps.

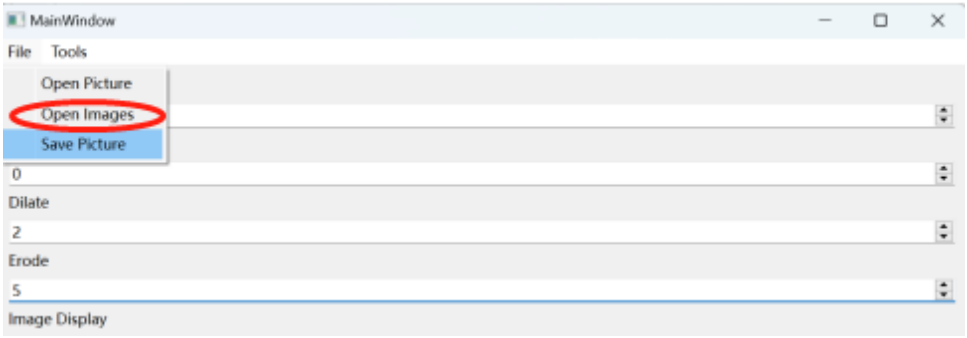

Image 11. Step of open multi pictures

The savePicture function

The savePicture function uses QFileDialog to get the path to the file that the user wants to save, and then saves the currently displayed image to that file. If the user cancels the dialog, this function will not do anything.

The save image function is used to get the path to the file that the user wants to save using Qt's QFileDialog class. Then, the QGraphicsPixmapItem object is converted to a QImage object using the toImage method of QPixmap and QImage is called. images can be saved after processing by using the save function to save the processed images. To use this function you need to click File in the upper left corner and then click Save Picture to select the path to save the image.

II. Interface display

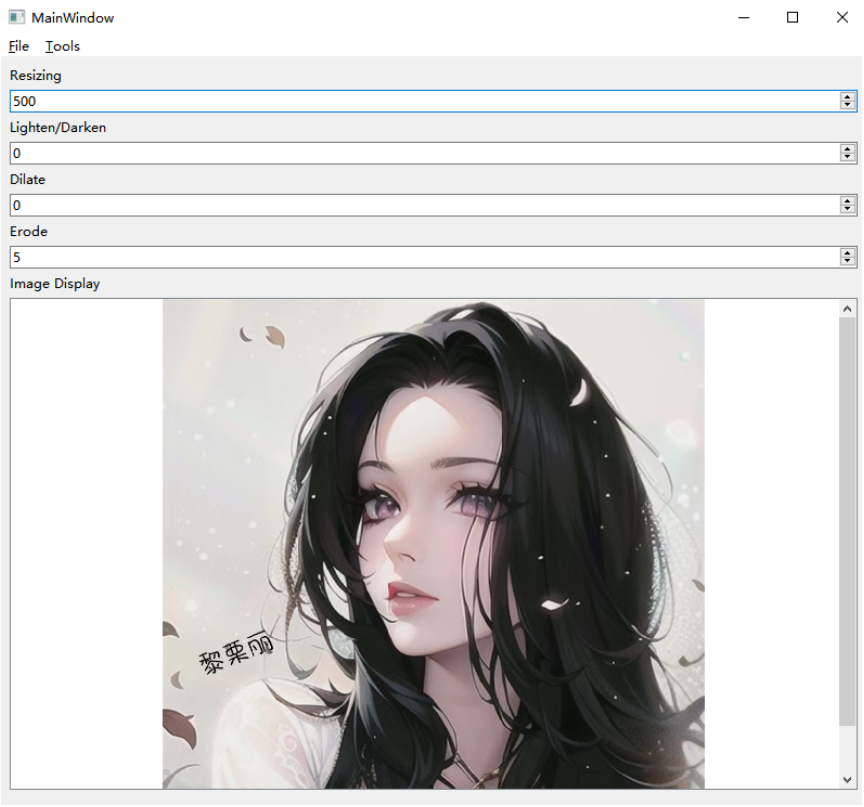

Image 12. Interface of the QT

- **Resize images**

The updateImageSize method is defined in the MainWindow class, which is used to resize the image based on user input.This function is implemented with a valueChanged signal connected to a QSpinBox in the main window, so when the user selects a new value in this SpinBox, the size of the image is automatically updated.

This feature allows the user to see the different sizes of the image in real time and provides more flexibility for further processing of the image (e.g. Canny edge detection or other functions).

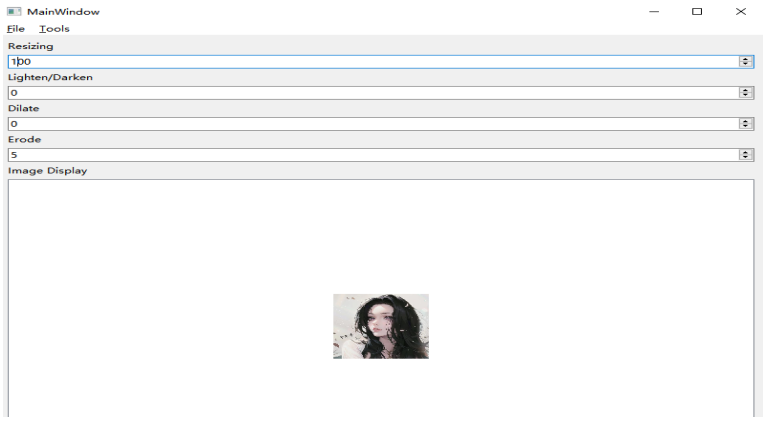

Image 13. Resizing in interface

- **Generating panoramas**

The application can generate panoramas. This is implemented through OpenCV's Stitcher class. First, at least two images are needed, then these QImage objects are converted to cv::Mat objects. OpenCV's Stitcher class accepts a list of cv::Mat objects as input and returns a cv::Mat object containing the panorama. Finally, the panorama is converted to a QImage object and displayed. Users can add multiple associated images by Open Images, and then use this function to generate a panorama.

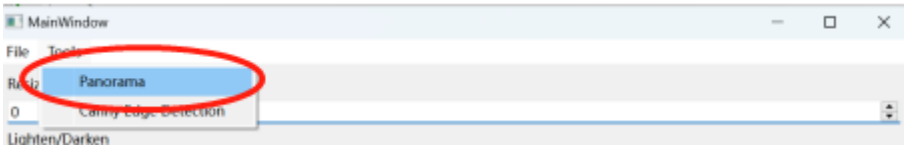

Image 14. Step of panoramas

- **Canny Edge Detection**

Use OpenCV's Canny function for edge detection. First, convert the QImage object to a cv::Mat object, and then convert this object to a grayscale image. Then, call the Canny function and pass in the threshold value for edge detection. Finally, the edge detection result is converted to a QImage object and displayed.

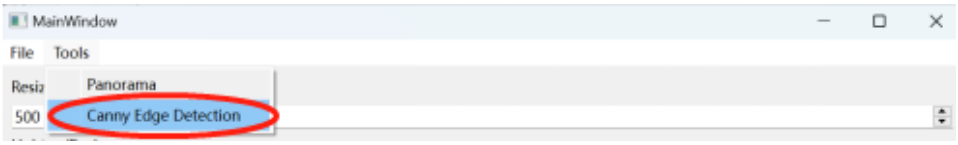

Image 15. Step of Canny

- **Changing Image Brightness**

The application can adjust the brightness of the image according to the value entered by the user in SpinBox. The implementation of this function relies on the convertTo method of the cv::Mat class.

- **Dilation and Erosion Operations**

The expand and erode operations are implemented using OpenCV's dilate and erode functions. Both functions require a structuring element, which is generated by the cv::getStructuringElement function. The user can adjust the relevant parameters to perform the expansion and eroding operations on the image with the values entered in SpinBox.

III. Advantages and Disadvantages

Advantages

Versatility: This application provides a wide range of image processing functions to meet most common image processing needs.
Ease of use: This application provides an intuitive user interface that allows users to select and apply image processing functions intuitively.

Disadvantages

Performance: The application may suffer from performance problems when processing a large number of high-resolution images. Optimizing the algorithm and improving code efficiency can solve this problem.
Extensibility: With the current architecture, adding new image processing features may require multiple code changes. The scalability of the application can be improved if a plug-in design is used.

Navigating the Project: Overcoming Difficulties, Collaboration, and Tools

Through these two months of hard work, we have completed this small image editor and it also has basic functions. Behind the completion of this project, we also encountered many difficulties.

Difficulties

1. Technical difficulties

When we first encounter OpenCV code, we may only be familiar with some basic functions and operations. However, as the project progresses, we will face more challenges and need to explore and understand more functionally rich functions and methods to implement processing of images.

During the project, we not only need to learn and understand the various functions and algorithms provided by OpenCV, but also need to search the Internet extensively for best practices and solutions on image processing. This helped us to choose the most suitable approach for the project needs and to ensure the code was clean and efficient.

In addition, in order to provide a user-friendly interface, we need to delve into and draw from a variety of resources, including online documentation, tutorials, sample code, etc., for best practices on interface design and interaction. This will help us design an intuitive, easy-to-use interface that provides a good user experience.

2. Other difficulties

As a team, we faced a challenge: actually, five of us came from different disciplines, so we rarely had the opportunity to get together in person to discuss the project's progress, except for time spent in Multimedia classes. Nonetheless, we actively collaborate through online communication channels, leveraging our respective expertise and Internet resources to move the project forward.

We rely heavily on online discussions to communicate and coordinate, including the use of chat tools and online meeting platforms. Through these tools, we are able to share progress, solve problems, and discuss matters such as design and feature

implementation. While this approach may present some communication challenges, we recognize the importance of teamwork and do our best to overcome these difficulties through positive communication.

Throughout the development process, although we encountered various problems and difficulties, we were able to continue to improve our skills and overcome difficulties by actively learning, making full use of resources on the Internet, and active and timely communication within the group to complete the project.

Collaboration

In terms of teamwork, we chose to form groups of members from the same major. Specifically, Qianqiu and Shuyuan formed one group, and Qiyao, Yawen, and Yuze formed another. This arrangement made it easier for us to discuss and communicate with each other and better fit our respective schedules. We could more easily discuss the details of the project together and solve problems in a timely manner. This grouping helped us to collaborate and make progress more effectively.

- ✓ Qianqiu and Shuyuan are responsible for the Resizing, Lighten / Darken and Canny algorithms, as well as the layout and organization of the report.
- ✓ Qiyao and Yawen are mainly responsible for Panorama / Stitching, as well as the Erosion/Dilatation part.
- ✓ Yuze is mainly responsible for creating a user interface using Qt.

Tools

1. Visual studio

For this project, we chose to develop and complete it using Visual Studio as the Integrated Development Environment (IDE), which provides powerful code editing, debugging, and building tools that allow us to write and debug code more efficiently. With Visual Studio's intuitive interface and rich functionality, we can easily organize project files, manage code versions, and collaborate in teams. It also offers a rich set of plug-ins and extensions that help us optimize our development process and improve code quality. Therefore, choosing Visual Studio as our IDE interface can effectively support our project development and accomplish our goals.

2. WeChat

We communicate and collaborate online mainly through WeChat. We create groups and share our ideas, discuss the progress of the project and solve problems within the group. WeChat, as an instant messenger, facilitates us to communicate at any time, give feedback and discuss matters related to the project in a timely manner. We communicate through text, voice, and pictures to ensure that every group member can participate in the discussion and stay updated with the project. The convenience and real-time nature of WeChat allows us to collaborate efficiently, improving the team's communication efficiency and the speed of project progress.

3. GitHub

GitHub is a cloud-based code hosting platform that is primarily used for version control and collaborative development. Although our team uses GitHub to a lesser extent, we still use GitHub to transfer final versions of our code. We prefer to use WeChat for communication and exchange because it's easy and quick and better suits our team's collaborative approach, and GitHub, as a code hosting platform, still provides us with the ability to back up and track code versions to ensure that our projects are always neat and organized.

4. Qt

For this project, we chose to use Qt as the tool to build the interface, which has an intuitive design interface and a rich library of functions where we can quickly design beautiful interfaces and implement various functions. In addition, Qt has good documentation and active community support, so when we encountered problems, we were able to find solutions quickly. Therefore, choosing Qt as our application development framework can effectively help us accomplish our project goals.

Conclusion

The combination of C++ and OpenCV allows for the realization of a wide range of image modifications and enhancements. By utilizing C++'s capabilities and OpenCV's extensive image processing features, we gained practical experience in working with digital images. This not only improved our understanding of multimedia programming but also enabled us to unleash our creativity by applying various image manipulation techniques.

Our journey with C++ and OpenCV in the Multimedia course helped us develop essential technical skills and an appreciation for the vast possibilities they offer. By immersing ourselves in image manipulation using C++, we gained insights into its potential for creating visually captivating experiences and solving real-world problems. These experiences have equipped us to tackle future challenges in software development, as C++ continues to be a driving force behind innovative applications and systems.

Overall, this image processing program is a useful tool based on Qt and OpenCV, which provides a variety of common image processing features. However, there are still some areas that could be improved, such as performance and scalability. We don't have time to add some advanced features to it, such as video processing, face recognition/recognition, advanced gui, etc. I will consider adding them to the application when I have time in the future, and I will also optimize the layout of the interface to make the application more beautiful and handsome. In the future, we can consider optimizing the algorithm and code efficiency, as well as using plugin design to improve the scalability of the application and be able to add new features more easily.

In conclusion, our exploration of C++ within the Multimedia course provided us with a solid foundation in the language and its applications. By incorporating OpenCV, we honed our skills in image processing and learned how to utilize C++ effectively in this domain. Armed with this knowledge, we are prepared to pursue further opportunities in software development and leverage the power of C++ to build impactful solutions.

References

- https://blog.csdn.net/tywwwww/article/details/126626804
- https://blog.csdn.net/stq054188/article/details/109136722
- https://blog.csdn.net/m0_62309595/article/details/122591381?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216865020201680018856739 6%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=168650202016800188567396&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-122591381-null-null.142^v88^control_2,239^v2^insert_chatgpt&utm_term=opencv%E4%BE%B5%E8%9A%80%E8%86%A8%E8%83%80C%2B%2B&spm=1018.2226.3001.4187
- https://zhuanlan.zhihu.com/p/47017516