Seam carving

Fengyu Wu

Tuesday, May 8, 2018

TABLE OF CONTENTS

Introduction:

We have witnessed a lot of advances in general multimedia systems in the recent years with the combined of computer vision. For example, the self-driving car is a perfect example. The success of the self-driving car is based on the combination of computer vision, artificial intelligence, and many multimedia technologies. For example, the self-driving car needs a camera to record the road data, and use this data to train the neural network and improve the performance of the self-driving car. Also, the car needs to use a laser and radar to detect the object surrounding the car and use computer vision technology to identify car and pedestrian. This data is then transferred to the computer in the car in real time. and the computer will then use these data to take action accordingly. The car also needs to locate itself using GPS and contain a dashboard that will connect seamlessly to the car like the Tesla's dashboard. This allows the user to visualize the surrounding of the car in real time and take action and gain control of the car if necessary. The dashboard also has other functionality like watching a movie, playing music and perform auto parking. There are tons of research right now about the self-driving car, for instance, I was a researcher at UB motion simulator lab, and our research focuses on the dynamic text display on a self-driving car and how the text affect people when the self-driving car is operating. There are still a lot of research need to be done since a recent new reported that a self-driving car had killed someone when it is self-driving. So, I am particularly interested in improving the safety of the self-driving car. And that is also my motivation, to make self-driving cars safer so that in the future, it will have fewer car accidents. Moreover, the self-driving car is closely related to computer vision and multimedia media technologies. The reason why I select this particular project is that seam carving it is an elegant algorithm to solve computer vision problems.

For instance, image resizing is an important part of our life such as image editors in Adobe Photoshop, image and file viewers in MeituPic, software magnifiers in Microsoft Windows built-in Magnifier, digital zoom in Apple iPhone products. The current challenges in resizing image are that sometimes, we will lose important information about the image if we are not careful. And I will propose a seam carving algorithm to overcome this challenge. Moreover, Seam carving algorithm can help to improve user experience, since it allows the displayed content to be resized without sacrificing too much of its quality. So, users with the different display size devices can enjoy the content without issues.

I had done some literature research papers before I implement this project. In these papers, the authors proposed different seam carving algorithm that fit in different situations. My proposed seam carving algorithm used a different approach to calculate the energy map which will give improved result in many situations. As there is no perfect algorithm, every algorithm has its advantage and disadvantage. It is important for me to learn the pros and cons of a different approach by other authors so that I can improve my own algorithm if necessary.

For instance, I read the paper [1] "Optimized image resizing using seam carving and scaling" where they propose a novel approach for seam carving algorithm. Their algorithm is also categorized as software approach because they did not use any hardware to achieve the performance they claim to have. In this paper, they optimize the image distance function to achieve a better result by finding the best seam carving number. They also combined image scaling technique with seam carving technique together. Their algorithm can support video resizing, but it is not perfect, they need to tweak their technique a little bit to make that to work. My algorithm is somewhat inspired by this paper and so I implement a functionality that allows the user to adjust the number of the seam to remove for the seam carving algorithm to stop. By implementing this, the user will have a chance to find the best seam carving number and generate a good result. My algorithm approach is an improvement upon them because, in my algorithm approach, the user can explore and set the seam carving number to anything they want and get a different result. Compare to their algorithm approach, my algorithm has less time complexity since my algorithm only considers image resizing, and the limitation of their algorithm is that the computation time is relatively long. I can achieve a better runtime in my algorithm compare to their algorithm.

Another interesting paper [2] "Seam carving for content-aware image resizing", and their algorithm is categorized as software approach as no hardware is used in their algorithm. They propose a seam carving algorithm that supports image resizing in both expansion and reduction and support many types of energy functions. My algorithm approach improves upon this paper and hence my algorithm has similar performance and functionality. The original algorithm does the computation automatically, however, not all images will work if the algorithm is run automatically, it can produce an unexpected and even bad result. So, I improve this algorithm by adding the functionality of user-defined parameters and the ability to view the progress of the algorithm. I also reduce some unnecessary functionality along the way. Compare to their algorithm approach, my algorithm is cleaner, faster and less complex than the original algorithm purposed by the author of this paper because my algorithm has an adjustable parameter and the computation can be tuned by the user to suit their preference and the paper can only do the computation in an automatic fashion.

I also reviewed the literature [3] Depth-Aware Image Seam Carving, and I find out that the authors propose a novel depth-aware single image seam carving approach by taking advantage of the modern depth cameras. And the seam carving algorithm they propose need hardware support, the modern depth camera to achieve the performance they claim to have. So, their algorithm is categorized as hardware approach and it is complicated that my approach and other similar software approaches because all we need is the algorithm implementation itself. My approach is different differs from their approach, because my algorithm does not need the support of a modern depth camera. Compare to their algorithm approach, my algorithm can be very easy to run and set up and it does not need to have a hardware to capture the additional supplement information and details and in turn, give a better run time.

In another interesting paper [4] “Stream carving: An adaptive seam carving algorithm”, the authors proposed a new content-aware image resizing scheme, Stream Carving. This is a novel algorithm that based on the original seam carving algorithm. And their algorithm is also categorized as software approach as no hardware is used in their algorithm. In their algorithm, they improved the result of the algorithm by introducing the concept streams. The stream is a larger seam in the retargeted image. And their algorithm will reconstruct the lost by filling the holes in the image with an inpainting method. They claim that by applying these techniques, we can obtain better results. However, by recovering the image the inpainting method, it is very time consuming and it cannot reconstruct textures and information in the original image may be lost. My algorithm approach differs from this algorithm, I removed seam carefully and only the seam with the least energy will be removed, and my algorithm does not need the process of recovering the image. Compare to their algorithm approach, my algorithm improved the time complexity and the result. So, the overall performance of my approach will be better.

In the paper [5] “Improved seam carving for image resizing”, the author proposed a new seam carving algorithm which incorporates anti-aliasing and thresholding technique to generate a better result. And their algorithm is categorized as software approach as no hardware is used in their algorithm. They claim that their algorithm can achieve a better seam carving result. However, their algorithm is unstable. Sometimes their algorithm performs better but sometimes it performs even worse than a normal seam carving algorithm. And due to the fact that they use backward-energy seam carving technique, and it needs times to determine the threshold factor, the runtime will be higher than my algorithm approach. My approach is differing from this algorithm because I have recommended parameter setting and the user can choose the parameter to what they like. Compare to their algorithm approach, my algorithm is more adjustable and it can find the best parameter to obtain a better result. Until they have an automatics way of determining the best threshold factor, my algorithm approach can achieve a better run time, better result and have less complexity in the code.

The overview of the project is that I will implement a seam carving algorithm that handles image resizing. The program contains an interactive user interface that contains menu bar and buttons. It allows the user to input different parameter to get different image result. The user will be able to see the steps and progress of the seam carving algorithm. The user can also display the energy map and compare result for a different algorithm. Finally, the user will be able to save the image file if they wish to do so.

My Proposed Approach:

My assumptions are that the seam carving algorithm result is based on the calculation of the energy map and the User have some knowledge of the algorithm's parameters like the iterations and direction of the seam. Since the iteration parts just remove the seam with the minimum energy which is the same across other types of seam carving algorithm. The meat of the algorithm is the way of calculating the energy as this can affect the result significantly. If the energy is miscalculated, the main object or sight in the image will be corrupted and removed.

I will then formulate the problem as follows. The user gives the program an image file and a set of algorithm parameters likes the direction to perform the algorithm and the iteration of the algorithm. Then the program needs to output an image that is resized based on those parameters.

To solve this problem, I had implemented an interactive user interface that allows the user to tune the parameters of the algorithm using Python. The user will also have the ability to select an image file from its own file system and import it into the interactive user interface to process the image.

Basically, the overall system design is the following. I implemented an interactive user interface in Python. The user interface is implemented using the power library PyQt5. I used this library to create a window to fit all the buttons with different functionality. I also added a menu bar with file open functionality which allows the user to open their own image and run the algorithm from there. The user interface window is hardcoded to be an 1100 * 600 window using set geometry method. Then I named the application to be "Seam Carving GUI" using set Window Title method. The menu bar and buttons are configured so that when they are pressed or triggered by the user, a specified function associated with that action will be called.

For example, if the user navigates the mouse to the top left corner where the menu bar resides in the user interface, and then click "file" and Open File, the file open method will then be called. And the user will be prompted by a Message Box, asking for the seam direction. The user will then click Yes if they want the seam carving algorithm to perform horizontally. If the user clicks No if they want the seam carving algorithm to perform vertically. If the user clicks the close button and does anything else, the action will be vertical by default. After this Message Box, the user will then need to input the number of seams to be removed by the algorithm. By default, this number is set to 50 and the recommended maximum number is 500. This is because if the seam to be removed is too much, the image will lose too much information and the algorithm will be meaningless as the main objects in the image are likely to be removed. Also, if the number of seams to be removed is high, for a large image, the runtime will be high. If this is set to 0 then, the algorithm will be meaningless as nothing got removed. In this case, the original image will be returned by the program.

After this information is gathered by the program, the user can now open an image file they wish to open. The program will then ask for seam carving algorithm type to be run. In my implementation, there are two types of algorithms in calculating the energy map.

I also include other algorithm types which are implemented by an external library for comparison reason. In my implementation, the algorithm type 1 and 2 is implemented by me. And other algorithm types 3,4,5 are from an external library for compassion reason. By default, algorithm type 2 is selected. In any case, the program will use that type of seam carving algorithm calculate the energy map for this user-selected image. After a few seconds of wait. The algorithm will be done. The runtime is depending on the type of seam carving algorithm the user selects, the size of the image the selected for the seam carving algorithm, and the number of seams to be removed. Then, the program will run the seam carving algorithm. After the run is finished, the user will have a chance to save the image if they want to do so or they can click No and the resulting image and the original image will be displayed on the screen.

The button Original is another function that allows the user to open an image file in their file system. This functionality will simply return the original image for the image file and display it to the user with the image name being the original image. This is for checking purpose so that the user knows it is the right image file they are expecting to work with.

The button Gray scale is a functionality that when the user clicks on it, it will ask the user if they want to save the image file after the processing or not. If No is selected, the program will ask for the image file that the user wishes to work with. Then after a few seconds, the gray scale of the selected image file will be displayed. If Yes is selected when they are asked to save the image file or not, the algorithm will run and then save the image on the file system afterward and then display it on the screen.

The button Energy map is a functionality that when the user clicks on it, it will ask the user for an image file in their file system. This image will then be processed using five different algorithms. After a few seconds of delay, the energy map for that selected image will be displayed. There are five images corresponding to different algorithms. Two of them are implemented by me, and the other three algorithms are for comparison purpose.

The button Compare is a functionality that when the user clicks on it, it will perform all five-energy map calculation algorithm. This will print out each of the algorithm runtimes to the console and I can find out which one has the best performance at runtime.

The button Energy map is a functionality that when the user clicks on it, it will ask the user for an image file in their file system. This image will then be processed using five different algorithms. After a few seconds of delay, the energy map for that selected image will be displayed.

There are five images corresponding to different algorithms. Two of them are implemented by me, and the other three algorithms are for comparison purpose. So, when the algorithm is done, five images will be displayed. These images are the energy map of the original image. Algorithm type one is an edge detection algorithm implemented by me to located the important pixels in the image using 1D Convolution. Algorithm type two is another edge detection algorithm implemented by me that is based on 2 D convolution. These two algorithms perform differently and can give a different result based on the input images. Algorithm type three is an algorithm from ndimage that find the energy map of the image. Based on some experiment, I find this algorithm incur some noise around the image and the result is not that good compared to my algorithm. Algorithm type four is an algorithm from open cv2 that find the energy map based on Sobel filter, it works well but there is still some noise point which can be critical to some image with small details. Algorithm type five is an algorithm from open cv2 that find the energy map based on the Canny filter, it works well and performance of this is similar to my algorithms.

The button Result is a functionality that when the user clicks on it, it will ask the user for seam direction. The user can click Yes if they want to process horizontally. Or they can also click No or close to process vertically. By clicking close, will cause the program to get the default setting. In this case, it is vertical. Then the program asks for the number of seams to be removed from the algorithm to stop. By default, this is set to 50, and the user can choose between 0-500. After these parameters are set, the user will be able to select the image in their own file system that they want to work with. And then they can choose the type of algorithm to run. By default, this is set to algorithm type 2. After that, the program will execute the code based on these parameters. After the algorithm is finished, the user will have a chance to save the processed image if they want to or skip this step to view the final image result. In the end, the image will be named as "result" and displayed to the user.

The button progress is a functionality that when the user clicks on it, it will first ask the user for a seam direction. This can be vertical or horizontal. And then, the user needs to decide stopping condition of the algorithm. It is in the range of 0-500, and 50 is the predefined value if the user did not specify any value. This is the number of seams to delete for the algorithm to stop. After setting these parameters, the user needs to pick an image of their choice from the file system and specify the algorithm type for the seam carving algorithm. Although they can choose from 5 types of algorithms, the default is set to 2 if nothing is specifying. After all these steps, the code will execute the seam algorithm based on these settings and output the progress of the image. There is no save option here because it will save a lot of images if the parameter is set too high.
The button Quit is a functionality that when the user clicks on it, a Message Box will pop up, it will warn the user that all unsaved change will be lost if they exit the program now. The user can decide if they really want to exit the program or not. If they click Yes, then the program will be terminated. And if No is clicked, the program will resume as if nothing had happened.

With the interactive user interface, buttons functionality and user-defined parameters, the seam carving algorithm can be executed based on the user preference. So, the resizing problem is solved because within this seam carving algorithm that I am implementing, I preserved the most useful information in the original image by removing the seam with the least energy. By calculating the energy map of the original image, I am able to identify the pixels with the important information and the pixels with less important information. Then I recursively remove the seam with less important information until the user specifies stopping condition is met. In this way, the information that is not important in the image will be removed and the information that is important is preserved. This approach can let the user resize the image with preference while minimizing the loss of quality due to the resizing.

And my proposed component fits because my seam carving algorithm with the interactive user interface can solve this resizing problem elegantly.
The hardware used in this project nothing but a laptop and there is no hardware implementation. The laptop is used for development, demo, and presentation. But there is software implementation. So, the software used in this project is Python and PyCharm IDE. The details of the implementation are the following:

In the Python file work.py, I implemented a method called work_work, this method takes four parameters. They are img, remove_direction, num_of_seam_to_delete and algo_type. The img parameter is the image that the user wants to process. And the remove_direction parameter is the direction of the seam algorithm, it can be either vertical or horizontal. The parameter num_of_seam_to_delete is the stopping criteria of the seam carving algorithm. When this number is reached in a loop, the algorithm will stop executing and output the result at that point. algo_type is the algorithm type selected by the user. The purpose for this is that the user can compare the result of different seam carving algorithm and find the best fit for their images.
If this parameter is set to 1, then the energy map will be calculated using 1D Convolution.
If this parameter is set to 2, then the energy map will be calculated using 2D Convolution.
If this parameter is set to 3, then the energy map will be calculated using ndimage 's Sobel filter.
If this parameter is set to 4, then the energy map will be calculated using cv2.Sobel.
If this parameter is set to 5, then the energy map will be calculated using cv2.Canny. This method will return an image result.

The work_work_progress is a method that also takes four parameters. They are img, remove_direction, num_of_seam_to_delete and algo_type. Similarly, img is for user input image, and remove_direction is set by the seam direction. And num_of_seam_to_delete is the number of seams that the user wants to delete. And lastly, algo_type is the algorithm type that the user wants to execute. In this method, the program will display the original image and the progress of the seam carving will also be shown. I put the images at every ten iterations into an array. And when the whole algorithm is done, the algorithm will output all the images that is previously stored in the array. Finally, the algorithm will output the original image, progress images, and the resulting image.

In the Python file other_algo.py, I implemented three external library algorithm that accepts parameters which is an input image from the user and returns the resulting image after applying the corresponding algorithm.

The method algorithm_3 uses the SciPy library to apply Sobel filter on the image.

The method algorithm_4 uses the OpenCV library to apply Canny edge detection on the image.

The method algorithm_5 uses the OpenCV library to apply Canny edge detection on the image.

In the Python file algo.py, I implemented a method called two_d that accepts a parameter that is the user input image and the gradient parameter which is the value of the computation of gradient in either x or y directions. The method will first store the shape of the input image in variables and then traverse through the whole image to compute the energy map by finding the edges.

I also implemented a method called one_d that accepts a parameter that is the user input image and the vertical parameter and the horizontal parameter. This is very similar to 2D filter expected that has two parameters instead of one and their purposes are eventually find the edges in the image. This method used these parameters and then applies the 1D-filters specified to obtain resulting images. This resulting image will be the energy map of the original image.

In the method my_algorithm, it required two parameters. Parameter img is the image that the user wants to work with, and the parameter algo_type is the algorithm type the user specifies.

If the algorithm type is 2, it will first turn the color image into the grayscale image then do a Gaussian blur on it. then it will execute my algorithm 2 which is the two D convolution algorithm. The preprocessing step is to apply a Gaussian blur on the original image to smoothing the image so that the noise is reduced so that it can give a better result. Then it constructs the filter and apply the two D convolution algorithm on the image and return the result. This result is the energy map of the original image.

If the algo_type is 1, then it will first turn the color image into the grayscale image, then apply a Gaussian blur on the original image, and then construct the filter and apply the one D convolution algorithm on the original image. This algorithm will detect the edges in the image and return the energy map after the function return.

If the algo_type is 3, then it will turn the image into grayscale first, then apply the method algorithm_3 which is the SciPy Sobel filter in other_algo.py with the parameter of the gray-scaled image.

Similarly, If the algo_type is 4, then it will turn the image into grayscale and then apply the method algorithm_4 which is the Sobel filter defined in other_algo.py with the parameter of the gray-scaled image.

Finally, If the algo_type is 5, then it will turn the image into grayscale first, then apply the method algorithm_5 which is the Canny filter defined in other_algo.py with the parameter of the gray-scaled image.

In the Python file gui.py, I implemented a class contacting different method that renders the user interface.

In method __init__, it will set the user interface dimension. In this case, I set the length to be 1100, and the width to be 600. After that, I set the name of the application to be Seam carving GUI. I then create the menu bar and the option in the menu bar and attach it to the main window. Finally, I link the action for "open" file to the method file_open. So, when the action is triggered by the user, the linked method will be executed.

In method only_result, I implement the button Progress functionality. So, when the button Progress is pressed by the user, the program will run this method. In this method, it will first use a message box to ask the user for the seam direction. The default value here is set to vertical. If the user pressed Yes, then the the value will be set to horizontal, and if the user pressed No or do nothing about the message box, this value will be set to vertical. Then this method will ask the user for the stopping condition of the algorithm which is the Number of seams to be removed. The default value here is 50, and the recommended range is 0 -500. This is implemented by an input Dialog. And if the value for will be set to whatever the user enters, If the user enters nothing, then the method will assume the value to be the default value which is 50. Then, the program will ask the user to select the image file on its file system using File dialog, then the program will store the name of the path into a variable. After that, the user will be asked to enter the algorithm type that he want to use. This is in the range of 1 -5, and it will return an energy map after the it is finished. After collecting these information, the program will then call the work_work method with these parameters to perform the seam carving algorithm. After a few seconds later, the algorithm return, and the user can choose to save the result or not by answering the message box. If Yes is selected, the program will ask for the image file name and extension. The user will have a chance to name the result image and set the file extension to either JPG or PNG. By default, this is set to JPG to save storage space. After the user enter those information, the program will save the image on the user file system. And the program will run and display the result image on the screen. If No is selected, the program will just display the result without saving the image.

In method file_open, I implement the open file functionality. So, when the drop-down menu bar is File is clicked by the user, and the open file, this method will be executed. In this method, the program will ask the user for the seam direction, the number of seams to remove, algorithm type, and the file they wish to open using the same mechanism similar to method only_result. Then the program read the user input image and store the parameters provided by the user. and then use these parameters to execute the work_work method to run the seam carving algorithm. After the run is done, the user will be able to save the image file in PNG or JPG if they wish to, or they can just skip the saving step and output the result after the algorithm is done. Finally, the program will display the original image and the resulting image to the screen.

In method home, I implement the user interface with buttons. This method is used to set the location of the button, the height and width of the buttons and the action of the buttons when they are pressed. This method basically attaches the buttons to the main window of the user interface and name the buttons. For example, the Result button is defined here. It is named as Result. Then I connect this button to method only_result so that when this button Result is clicked, the only_result method is executed. And then I set the width and height of this button to be 100. Finally, to move this button to a location to the horizontal position of 600, and vertical position of 250.

In method see_seam_progress, I implement the functionality of button Progress. So, when this button is pressed by the user, it will execute the code in this method. First of all, the program will ask the user for the seam direction using a message box, the user can choose vertical or horizontal. Secondly, the program will ask the user for the stopping condition, that is the Number of seams to be removed from the seam carving algorithm to stop. If 0 is selected, then the algorithm will immediately return as there is nothing need to be removed. On the other hand, if a large number is selected, then the algorithm may take a long time to run or most of the information in the image got removed. So, the recommended range here is from 0-500, and by default, this is set to 50, and the user can increase it if they wish to. And then, the user can open the image file from their file system and select an algorithm type to process that image. Finally, this method will call the method work_work_progress with these parameters. That method will show the progress of the seam carving algorithm and display the steps to the user. There is no save option here in because this button Progress is for viewing purpose, and may cause a lot of image being stored in the user file system if the parameter is not set properly. The user is welcome to save it manually once the image is displayed.

In method close_application, I implement the functionality of button Quit. So, when this button is pressed by the user, it will execute the code in this method. What this method will do is that it will ask the user for confirmation and if the user really wants to terminate the program it can click Yes, and all the unsaved changes will be lost and the program will be terminated because the sys.exit() is executed. If No is clicked, then the program will resume as there is nothing happened.

In method check_performance, I implement the functionality of button Compare. So, when this button is pressed by the user, this will get the input image from the user and the program will use this image as the parameter for all five-seam carving algorithm and determine their run time. Finally, it will print the run time of each method to the console so that I can see which algorithm has the best runtime performance.

In method open_ori_img, I implement the functionality button Original. This method will let the user select an image in their file system and display it. The program will also print the image path to the consoles for viewing purpose. This is implemented so that the user can verify the input image is actually the one that they are intended to process.

In method open_gray_img I implement the functionality of button Grayscale. This method will let the user select an image in their file system and then the program will execute this method to turn the image into grayscale and then the user will be able to save the grayscale image if they want to do so by clicking the message box. If the pressed Yes, then they can also name the resulting image and change the extension to JPG or PNG. After the user is done picking the name and extension, the program will save the grayscale image and display it to the screen. If No was selected, then the program will just display the grayscale image without saving it and the title will be Button showed gray Image.

In method open_energy_img, I implement the functionality of button Energy map. Basically, this method will first let the user select an image, and then the program will read the image and store it in a variable. And then I turn this image into grayscale and apply a Gaussian blur to reduce noises. After the image is done with these preprocessing steps. I will store this processed image in a variable. After that, I will run all five algorithms with this processed image as the parameters. These five algorithms are used to find the energy map of an image. So, after a few seconds, all five algorithms are finished, and I will display the result of these algorithms to the user. These images are the energy map of the original image under different algorithm. This is also a way to compare the performance of different algorithm because I can see the different energy map generated by a different algorithm and evaluate them. The user can use this button to find the best algorithm for their image and get use the algorithm that gives them the best result.

In method run, implemented the main method of the whole program, so when it runs it will launch the program and start up the user interface. There are also 12 sample images in the code folder, these are used in images taken from the internet and it is used for testing, debugging and demo purpose.

The Resource used for implementation are the following. I first want to build an Android app and use Java, but then I switch to the Python programming language since it is more readable and concise. So, I used the PyChram IDE as the development environment. Meanwhile, I pick up Python programming tutorial from sentdex on YouTube. I also consulted the and PyQt5, OpenCV, Scipy, Numpy official documentation on their website to understand the different method usage. And I learn the implementation from the research paper about seam carving algorithm from the midterm report.

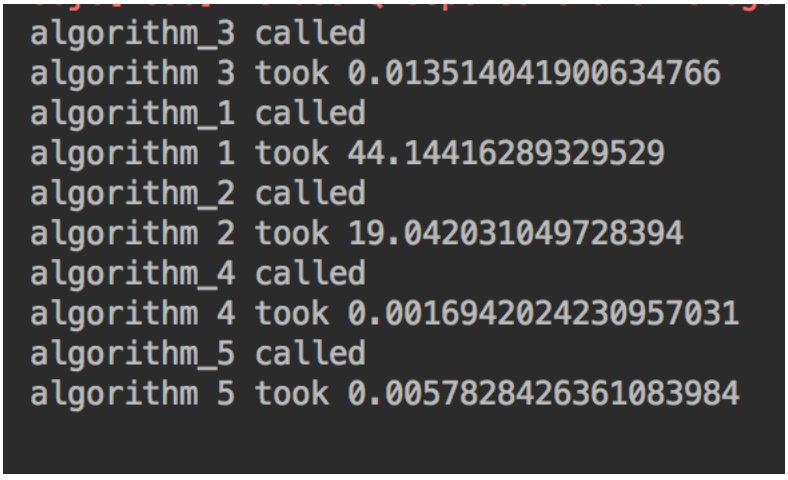
Outcome and Deviations:

Here I will give a presentation of the project outcome, the dataset used and collected are some sample image files from the Internet. There are total 12 images with JPG and PNG format. For more information about the presentation of the project outcome, please visit: https://youtu.be/1F-B5_QwaEg

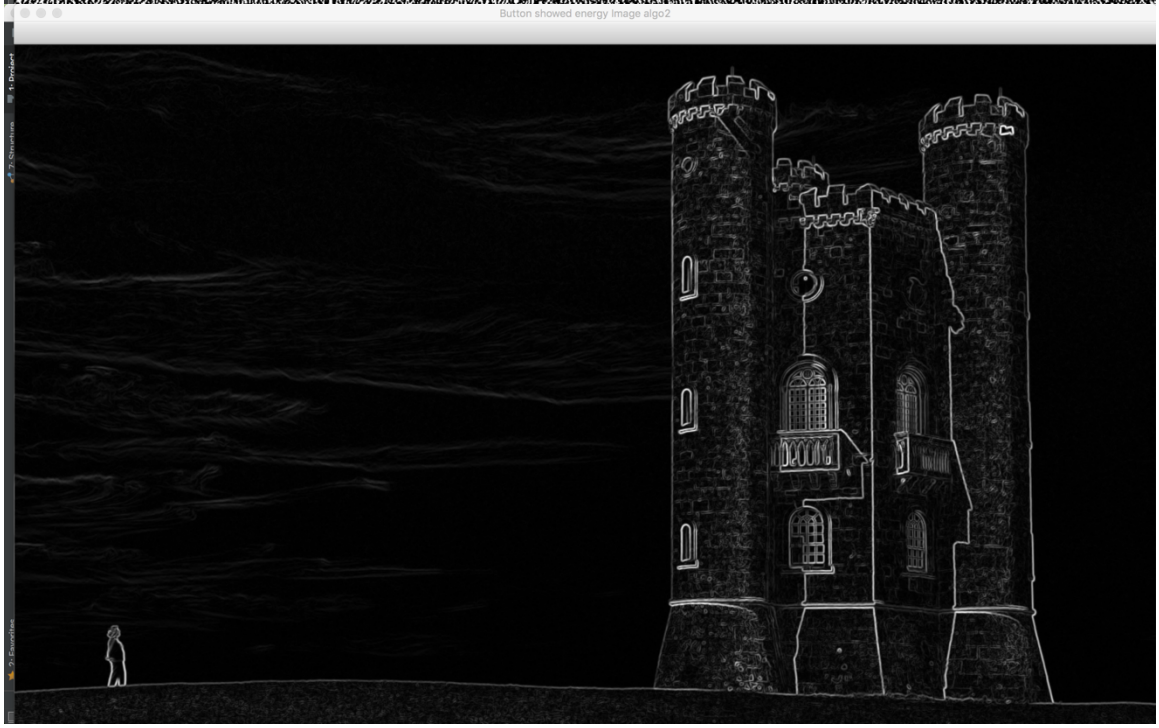
The evaluation metrics are the runtime of the algorithm and how the energy map was generated by the algorithm. For this, I implemented two buttons to compare the result. In Button Compare, the program will compare the run time of the five algorithms and display their individual runtime to the console. In the Button energy map, the program will compare the energy map calculation and display the result of the energy map for each algorithm.

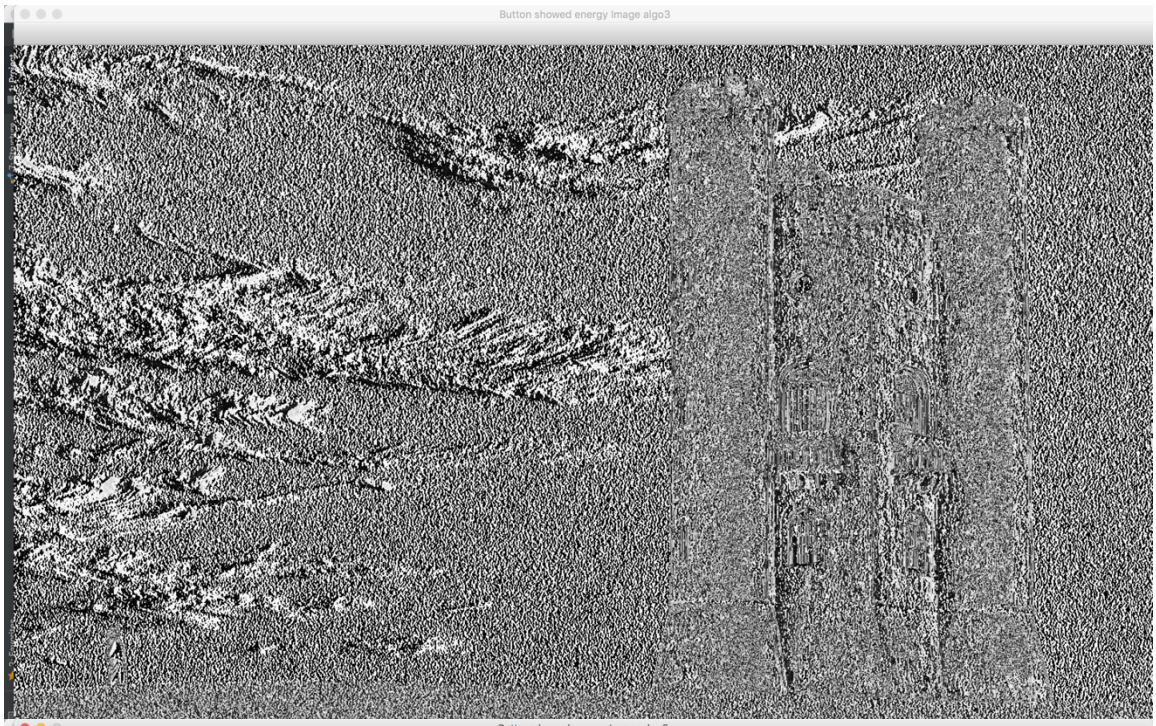
The experimental results are that my algorithm 1 and 2 (type 1, 2) perform very well. They can find the edges very well and removed noise correctly, and therefore give a perfect energy map.
But the other 3 algorithms (type 3,4,5) performance is also acceptable. They can find the energy map, although, with some noise, it is still an acceptable energy map.
Although the run time of my algorithm 1 and 2 are high, it gives better energy map. Hence, they are better than the other 3 algorithms in performance.

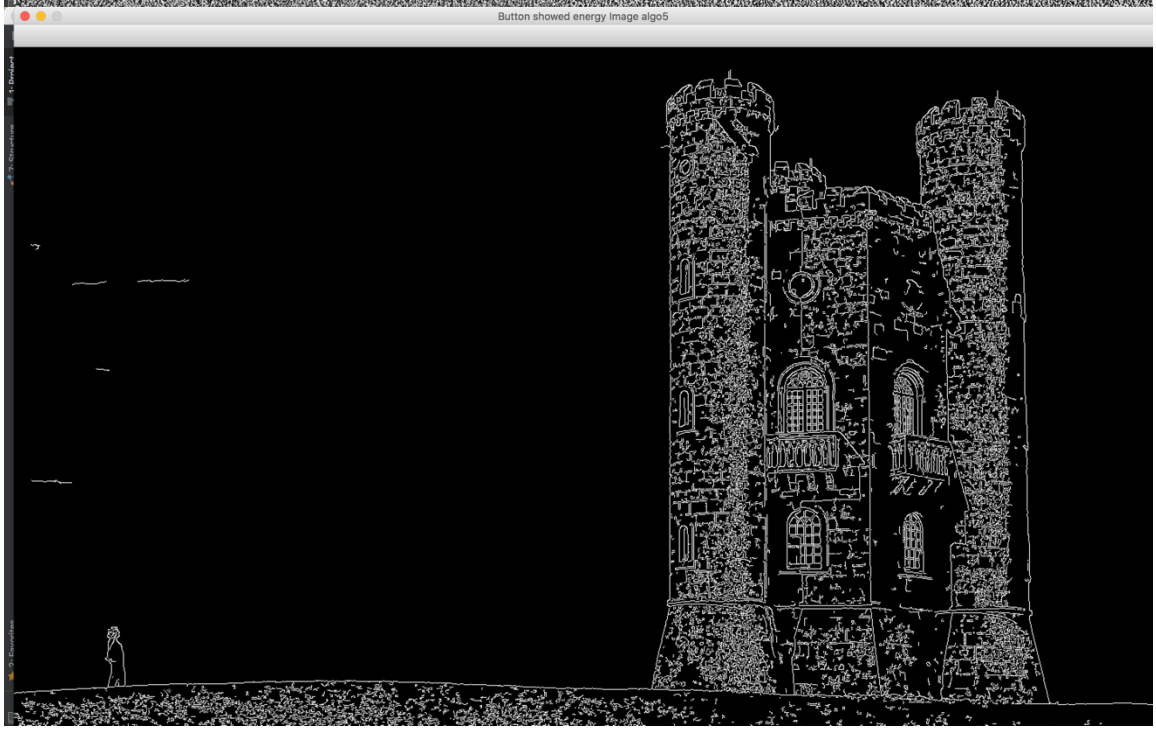
Here I show the comparisons with relevant approaches. The following image is the run time of the algorithm from type 1 – 5.

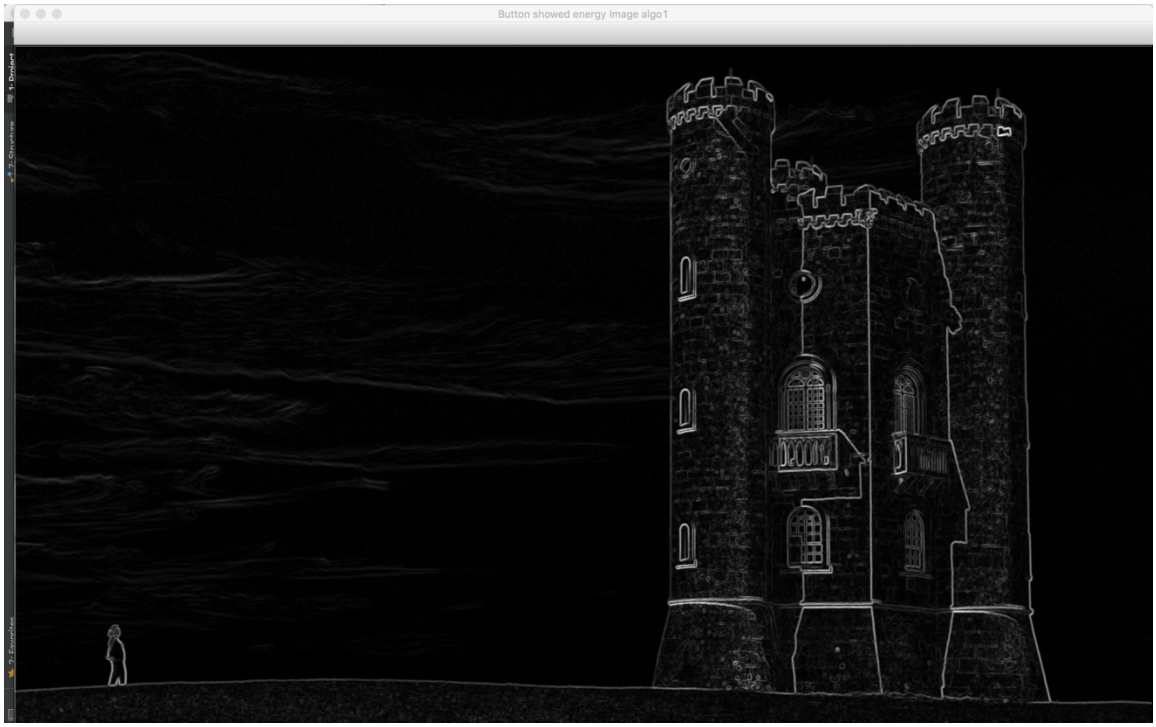



Button showed energy Image algo4

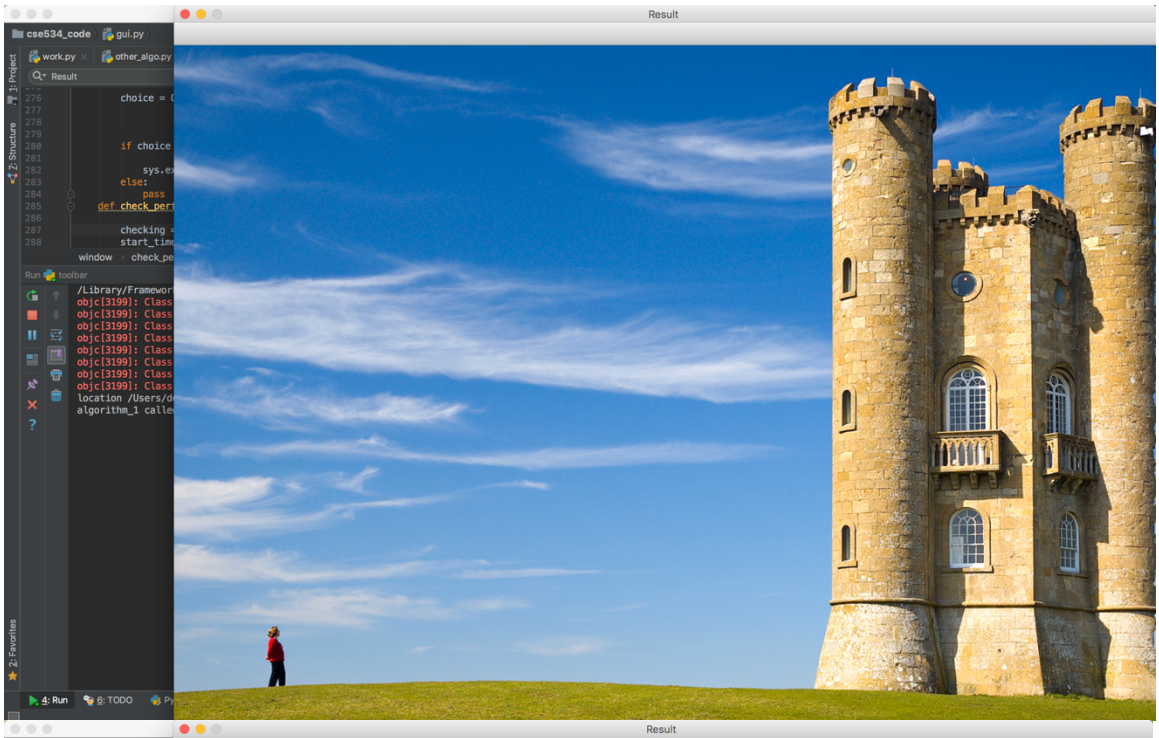

Button showed energy Image algo2

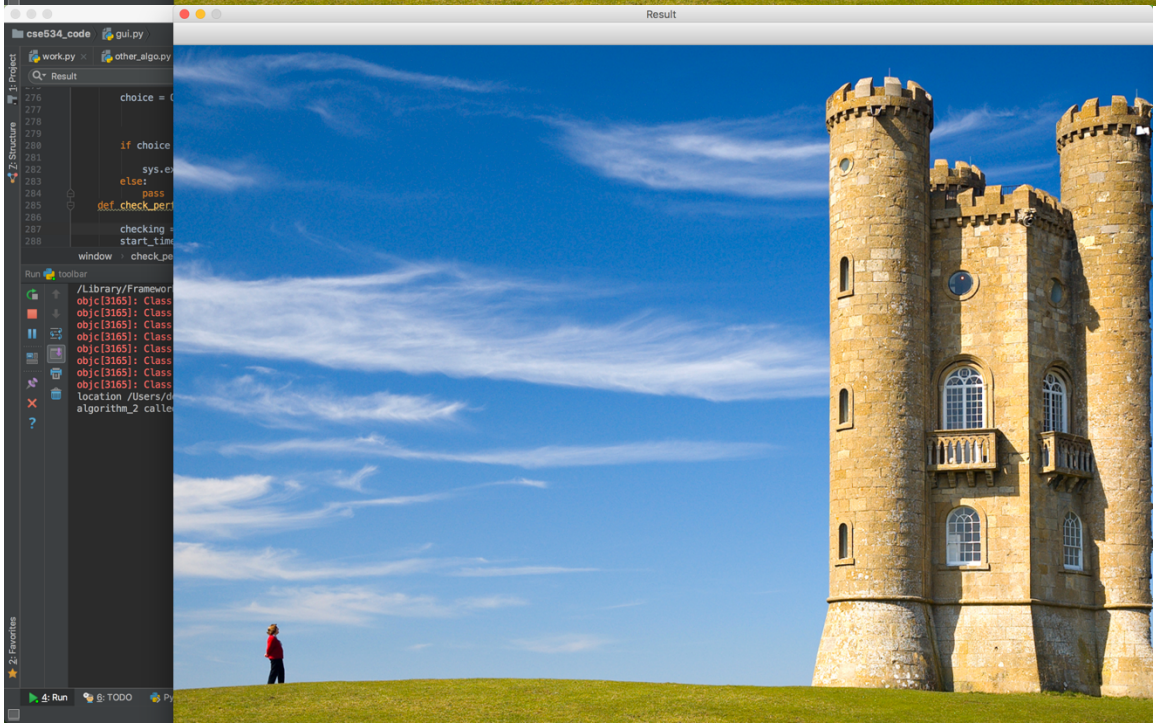

Button showed energy Image algo3

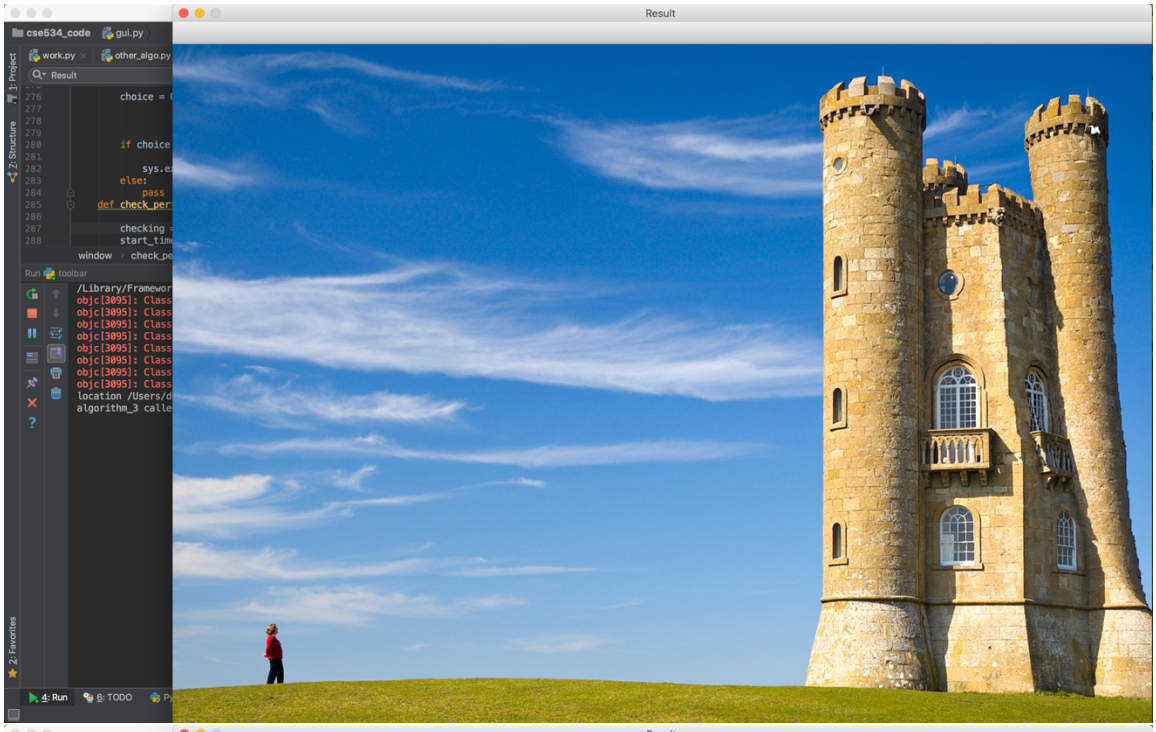

Button showed energy Image algo5

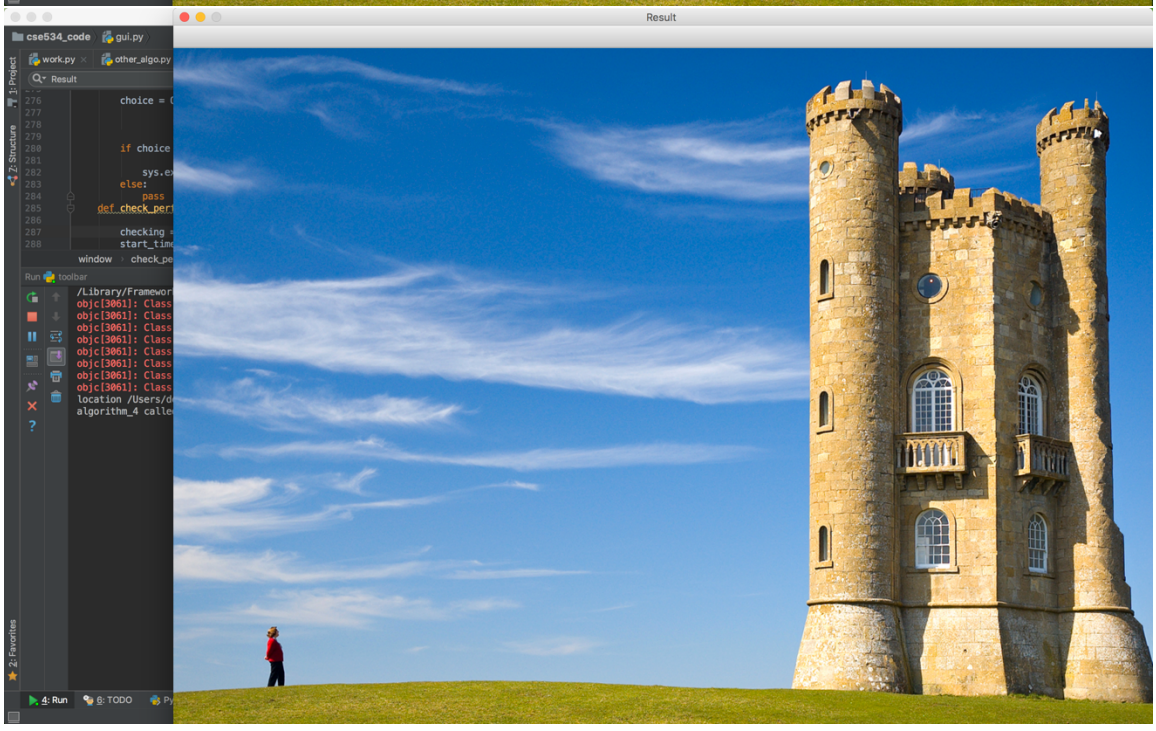


The 5 images above are the energy map generated by the algorithms, and the following 5 images are the outcome of the seam carving algorithms is finished.


cse534_code
gui.py
work.py
other_algo.py
Result
choice =
if choice
sys.ex
else:
pass
def check_perf
checking
start_tim
window
check_pe
Run
toolbar
/Library/Framewor
objc[3199]: Class
objc[3199]: Class
objc[3199]: Class
objc[3199]: Class
objc[3199]: Class
objc[3199]: Class
objc[3199]: Class
objc[3199]: Class
location /Users/d
algorithm_1 calle
4: Run
6: TODO
1: Project
7: Structure
2: Favorites
Result


cse534_code
gui.py
work.py
other_algo.py
Result
choice =
if choice
sys.ex
else:
pass
def check_perf
checking
start_tim
window
check_pe
Run
toolbar
/Library/Framewor
objc[3165]: Class
objc[3165]: Class
objc[3165]: Class
objc[3165]: Class
objc[3165]: Class
objc[3165]: Class
objc[3165]: Class
objc[3165]: Class
location /Users/d
algorithm_2 calle
4: Run
6: TODO
1: Project
7: Structure
2: Favorites
Result


Result
cse534_code
gui.py
work.py
other_algo.py
Result
choice =
if choice
sys.ex
else:
pass
def check_perf
checking
start_tim
window
check_pe
Run
toolbar
/Library/Framewor
objc[3095]: Class
objc[3095]: Class
objc[3095]: Class
objc[3095]: Class
objc[3095]: Class
objc[3095]: Class
objc[3095]: Class
location /Users/d
algorithm_3 calle
4: Run
6: TODO
1: Project
7: Structure
2: Favorites


Result
cse534_code
gui.py
work.py
other_algo.py
Result
choice =
if choice
sys.ex
else:
pass
def check_perf
checking
start_tim
window
check_pe
Run
toolbar
/Library/Framewor
objc[3061]: Class
objc[3061]: Class
objc[3061]: Class
objc[3061]: Class
objc[3061]: Class
objc[3061]: Class
objc[3061]: Class
location /Users/d
algorithm_4 calle
4: Run
6: TODO
1: Project
7: Structure
2: Favorites

Here, I will discuss the outcome. As we can see, the algorithm 1 has the slowest run time, and the algorithm 4 has the fastest run time. But the performance for algorithm 4 is worse than algorithm 4 because the image contains a lot of noise points in the energy map. algorithm 2 and algorithm 1 have the similar performance. (Both images removed noise, and detect edge well). But the runtime of algorithm 2 is slightly faster because it has only one nested for loop, and algorithm 1 has two nested for loop. Algorithm 3 ‘s runtime is the better than algorithm 1 and 2, but it contains a lot of noise, so its performance will be worse when the image has a lot of details. Algorithm 5’s runtime is also very fast, actually the second fastest, but it did not capture every detail in the image, and it is very sensitive the noise.

There are no deviations here. Because the result makes sense. In algorithm 3 and 4, there is a lot of noise in the energy map, and so the final result image will suffer from the noise and the shape of the main object is twisted. Algorithm 5 perform better than algorithm 3 and 4 but the edge is still twisted in the middle, and there is still some useless information being kept by the image. Also, because algorithm 5 is sensitive to parameter and noise. The energy map of algorithm 5 will be varied depends on the image and the parameter. In this particular image, it performs worse than algorithm 1 and 2. Finally, algorithm 1 and 2 performance is very close, but if I consider the runtime for these two algorithms, the algorithm 2 will be the winner because it has fewer loops in the code and runs faster than algorithm 1.

What I learned from this experiment is the calculation of the energy map play a dominating role in the algorithm and preprocessing is important. In my implementation of algorithm 1 and 2, I first convert the image into grayscale image and then apply the Gaussian blur to reduce the noise and some useless details. Although the preprocessing steps take sometimes to finish, the performance of the seam carving algorithm and the energy map are improved by doing the preprocessing. This gives me a better result in my algorithm 1 and 2. And I also learned that if I want the runtime to be faster, it need to scarify some performance. For example, algorithm 3 gave a worse result compare to algorithm 1 and 2. But it beat algorithm 1 and 2 in the runtime.

Summary and Discussion:

In summary, this project implements a seam carving algorithm that resizes images while preserving the important objects in the image. The algorithm can be adjusted by the user with different parameters to achieve the user's goal. It has various way to calculate the energy map of the image and a wide range of input parameters for the user to choose from. The user can try out different combination of parameters and obtain the optimal result thanks to the interactive user interface. The program is responsive and user-friendly with recommended parameters. The user can have no knowledge about the algorithm and still get a decent result with the default setting.

I learned a lot in this course. In the classroom, I found the slides are very informative and contain many practical examples. Although the material in the slides is a lot and complicated for me, it is well organized and the instructor is very knowledgeable and explains the material really well. This really helps me to understand the material in class and prepare me for the quiz, homework, and project.

Moreover, In the homework and quizzes, it contains a mix of practical and theoretical questions. It is very challenging to me. So, I study a lot about it homework and quizzes, and in this learning process, I find myself learning a lot from it, for example, I learned how to manage my time when the deadline of assignments approaching. Moreover, the instructor is very helpful and caring and most of my doubt is clear up during the office hours and after the lecture.

Furthermore, I also learned a lot from the project. At first, I was not sure what project topic to pick, and the first idea of my project topic is image search engines because I was passionate about computer vision. With the guidance and support from the instructor, I finally discover the project topic I really want to work on, that is the seam carving algorithm. I implemented this algorithm to help solidify my understanding of the concepts that were cover in the class. And since this topic is closely related to computer vision, I feel very fulfilled when I finally finished implementing it. Along the way, I pick up the Python programming language and learned how to build a user interface from scratch. Additionally, I also gained communication and organization skills because I was working in a team of two and we work together using GitHub. I really enjoy taking this course and learned a lot from the instructor.

Reference List:

[1] “Optimized image resizing using seam carving and scaling” – ACM SIGGRAPH Asia, 2009

[2] “Seam carving for content-aware image resizing” – ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH, 2007

[3] “Depth-Aware Image Seam Carving” – IEEE Transactions on Cybernetics, 2013

[4] “Stream carving: An adaptive seam carving algorithm” – IEEE International Conference on Image Processing, 2010

[5] “Improved seam carving for image resizing” – IEEE Workshop on Signal Processing Systems, 2010