Documentation

For the image processing library created
By group

“Linear Regression”
(Chenyu Ren, Gustavo Machado, Vidushee Geetam, Yifan Wang, Yulin Zhuo)

For the course
Advanced Programming

Imperial College London
24th March, 2023

Image Class

Image Class: Provides various constructors to create image object. Main functionalities are to load images into 1D arrays, extract image information(height, width, pixels and channels) and save 1D array back into an image.

1. int Image::**load**(): loads the given image into a 1D array
2. int Image::**save**(std::string output_path): saves the 1D array as an image at the specified file path
3. std::string Image::**get_name_in_path**(std::string file_path):helper function, creates a string of desired file path

Filter Class

Filter Class: It is used as a tool, it contains all the 2D and 3D filters in it.

1. Image Filter::**greyScale**(Image& im): This function converts an RGB image to grayscale.
2. Image Filter::**autoColorBalance**(Image& im): This function balances the color of an image.
3. image Filter::**brightness**(Image& im, int bright_value, bool is_auto): This function generates a new image after adjusting the brightness of the input image.
4. Image Filter::**histogramEqualization**(Image& im): This function generates a new image after applying histogram equalization to the input image.
5. image Filter::**median_blur**(Image& im, int kernel_size): This function applies median blur to the input image.
6. Image Filter::**box_blur**(Image& im, const int kernel_size): This function applies box blur to the input image.
7. Image Filter::**gaussian_blur**(Image& im, const int kernel_size, const double sigma): This function apply gaussian blur to the input image
8. Image Filter::**robert_cross_ed**(Image& im, const char* blur_name, const int blur_kernel_size, const double num): This function applies Roberts' cross edge detection to the input image.
9. Image Filter::**scharr_ed**(Image& im, const char* blur_name, const int blur_kernel_size, const double num): This function applies Scharr edge detection to the input image.
10. Image Filter::**sobel_ed**(Image& im, const char* blur_name, const int blur_kernel_size, const double num): This function applies Sobel edge detection to the input image.
11. Image Filter::**prewitt_ed**(Image& im, const char* blur_name, const int blur_kernel_size, const double num): This function applies Prewitt edge detection to the input image.
12. unsigned char* Filter::**get_edge_image**(Image& im, pair<vector<vector<double>>, vector<vector<double>>> kers, const char* blur_name, const int blur_kernel_size, const double num) : This function prepares the image for doing edge detection(convert to grayscale, then apply blur).

13. Image Filter::**blur**(Image& im, const char* blur_name=NULL, int kernel_size=0, double num=0): This function applies blur to the input image, making a union of all the blur functions.
14. Image Filter::**edge_detection**(Image& im, const char* edge_name=NULL, const char* blur_name=NULL, int blur_kernel_size=0, double num=0): This function apply edge detection to the input image, make a union of all the edge detection functions
15. void Filter::**bubble_sort**(vector<unsigned char>& vec, bool (*cmp)(unsigned char, unsigned char)): This function sorts the input vector in ascending order.
16. unsigned char* Filter::**median_blur_loop_3d**(const unsigned char* vol, int kernel_size, int chanel_size, int vol_w, int vol_h, int vol_l, bool is_alpha) : This function applies median blur to the input volume.

Conv_tool Class

Conv_tool Class: Provides functions required for doing 2D and 3D convolutions. It also provides two casting functions to convert data between unsigned char array and double array.

1. double* **ConvTool::conv_2d**(const double* im, std::vector<std::vector<double>> kernel, int chanel_size, int im_w, int im_h, int k_w, int k_h, bool is_alpha): This function performs 2D image convolution.
2. double* **ConvTool::conv_3d**(const double* vol, std::vector<std::vector<std::vector<dou--ble>>> kernel, int chanel_size, int vol_w, int vol_h, int vol_d, int k_w, int k_h, int k_d, bool is_alpha): This function performs 3D image convolution.
3. unsigned char* **ConvTool::cast_double2unsChar**(double* input, int size): This function casts a 1D array of double to unsigned char.
4. double* **ConvTool::cast_unsChar2double**(unsigned char* input, int size): This function casts a 1D array of unsigned char to double.

Kernels Class

Kernels Class: This tool class generates different types of kernels, such as box and Gaussian kernels. These kernels are used by the Filter class methods to apply the corresponding filters.

1. const double Kernels::**pi_fun**(): This function returns the value of pi, it will be used for calculating the Gaussian kernels.
2. vector<vector<double>> Kernels::**box_kernel**(int size): This function generates a 2D array of a box kernel by taking a kernel size as input, for example, a 3x3 kernel should take an input 3.
3. vector<vector<double>> Kernels::**gaussian_kernel** (int size, double sigma): This function generates a 2D array of a gaussian kernel by taking a kernel size as input, for example, a 3x3 kernel should take an input 3.
4. vector<vector<vector<double>>> Kernels::**gaussian_kernel_3d**(int size, double sigma): This function generates a 3D array of a gaussian kernel by taking a kernel size and sigma (standard deviation) as input.

5. pair<vector<vector<double>>,vector<vector<double>>>Kernels::**robert_cross_kernel**() : This function generates a Roberts' Cross kernel
6. pair<vector<vector<double>>,vector<vector<double>>>Kernels::**scharr_kernel**(): This function generates a Scharr kernel.
7. pair<vector<vector<double>>,vector<vector<double>>>Kernels::**sobel_kernel**(): This function generates a Sobel kernel.
8. pair<vector<vector<double>>,vector<vector<double>>>Kernels::**prewitt_kernel**(): This function generates a Prewitt kernel.

Volume Class

Volume Class: Provides two different constructor to create Volume objects, one with filepath as argument, and second with all the required volume parameters: height, width, number of images, channels in each image and overall pixel data for the volume. Main functionalities are loading a 3D volume of image into a 1D array and writing out a 1D array back into a volume, i.e. a folder containing a set of 2D images.

1. *Getter functions:* These functions provide access to the properties and data of a Volume object, i.e. width, height, channels in each image, number of images, pixel data and file name of each image
   a. int Volume::**getWidth**()
   b. int Volume::**getHeight**()
   c. int Volume::**getChannels**()
   d. int Volume::**getTotalImages**()
   e. unsigned char *Volume::**getData**()
   f. std::vector<std::string> Volume::**getFileNames**()
2. void Volume::**info**(): This method updates the width, height, and channel parameters of a Volume object by retrieving this information from a single image in the volume using the STBI library.
3. void Volume::**load**(): This method loads a volume of images into a Volume object by reading each image individually and storing it in the final volume data according to the 1D representation used by the STBI library.
4. int Volume::**save**(std::string output_path) : This method saves the volume of images to a specified output path by generating and saving each image individually using the STBI library.
5. std::vector<std::string> Volume::**get_output_file_paths**(std::string output_path): This method method generates a vector of output file paths for each image in the volume by extracting the file name from each image path and concatenating it with the specified output path. It returns the vector of output file paths.
6. void Volume::**getImageFiles**(): This method adds the names of all image files in the directory specified by the path member variable to the images vector of a Volume object, and then sorts the images vector.

7. void Volume::**tokenize**(std::string const &string, std::vector<std::string> &output, const char delimiter): This method splits a given string into a vector of strings using a specified delimiter. It utilizes a stringstream to extract the individual strings, which are then pushed onto the output vector.
8. void Volume::**sortImageFiles**(): This method sorts the images vector of a Volume object in ascending order using the bubble sort algorithm.
9. void Volume::**allocate**(int channels): This method allocates memory for the image data of a Volume object. The size of the data is based on the width, height, and number of channels of the volume, and is stored in the data member variable as an unsigned char pointer.
10. char *Volume::**generateSingleImagePath**(int index): This method generates the file path of a single image in a Volume object by concatenating the path member variable, a delimiter string ("/"), and the name of the image file at the specified index. It returns the generated file path as a char pointer.

Projection Class

<u>Projection Class</u>: It provides the tool to do projection on a volume object. It can choose to project on different methods and can choose part of the image sets to projecting.

1. Projection::**Projection**(Volume &volume): Constructor of a projection tool object.
2. void Projection::**allocate**(int size): method to allocate memory for producing projected images.
3. Image Projection::**chosen_projection**(std::string choice, int start_img, int end_img): methods to output the chosen projection. It can choose a method among “max”, “min” and “average”. It can also specify the starting image index and the end image index (Note: it is the real index not its file name).

Slice Class

<u>Slice Class</u>: It provides a function to slice the image in a certain plane. It turns a volume object to an image object.

1. Image Slice::**slice**(Volume& volume, const char* plane, int& position): It takes a volume object (a set of images), the plane the tool is going to cut and the location of the plane in its direction. The user can choose to cut the volume in plane “XZ” and “YZ”.