2D Material Auto Finder Code Explanation Manual

Jingxu Xie

August 11, 2020

https://github.com/jingxuxie/2D-Material-Auto-Finder

Contents

1 Introduction

This code explanation manual describes almost all the codes and features in the 2D material auto finder program. To keep the original codes clean and tidy, instead of writing comments and explanations in the *.py files, I decided to write a separate manual to explain the usage of variables and methods in the codes. To follow this manual and understand the code for future development, basic knowledge of Python, PyQt5, and object-oriented programming is required. I try to write explanations in detail and I hope this may help to understand the codes.

The top folder contains a "support_file" folder containing some icon images and several *.py files, most of which are used for testing, and only a few of them are used for the main program. The most important files are MainWindow2_0.py, layer_search.py, layer_search_TMD.py, auxiliary_class.py, auxiliary_func.py, Threads.py. The structure is: MainWindow2_0.py will create a GUI and Threads.py control the scanning stage and focus, then layer_search.py will do finding work. A short description of all the files is listed below.

autofocus.py: This is just a test file for autofocus and is not relevant to the main program.

auxiliary_class.py: This file contains a lot of auxiliary classes used in the main program and it will be discussed in detail.

auxiliary_func.py: This file contains a lot of auxiliary functions used in the main program and it will be discussed in detail.

BresenhamAlgorithm.py: This file contains functions to find points on the line, rectangle, and circle. This is used for drawing shapes.

Camera.py: This is a class used to define and call a camera.

canvas.py: This is a test file and has been deprecated.

control_bx2.py: This is a test file for communicating with BX2 software.

control_other.py: This is a test file.

drawing.py: This file defines several shapes and is used for drawing.

edge_detection.py: This is a test file for detecting substrate edges and it has been deprecated.

layer_search.py: This is the core file to locate and identify graphene on SiO_2 substrate.

layer_search_TMD.py: This is the core file to find TMD materials on SiO_2 substrate.

MainWindow2_0.py: This is the main file to create the GUI and call all the other functions.

plane_formula.py: This is used to get plane parameters and height positions when doing autofocus.

setup.py: This is a test file for auto upgrading the software and it has been deprecated.

Sony_view.py: This is an old version file.

Sony_view20.py: This calls MainWindow2_0 and will create two GUI windows, one for live view, one for editing images.

test.py: Just a test file to test anything.

Threads.py: This file contains several threads to control the scanning stage and the focus.

Transparent.py: A temporary file to make the image transparent.

2 MainWindow2_0.py

This is one of the most important files in the program. It uses PyQT5 to create all the features of the GUI. This is a relatively large file containing hundreds of variables and methods. To help understand this file, two important things need to be mentioned.

- In this file, two types of images are used to make the code clear and easy to maintain. One is img_raw, i.e., the raw image read directly from camera or file, the other one is img_show, the image after some processing and shown on screen.

- To make it easier to deal with the mouse position, the absolute position is used. The original position is relative to the GUI window, then it is converted to be relative to the raw image, i.e., the absolute position. And convert it back whenever using the position. The reason to do this is that there are a lot of geometry transformations such as zooming in and using absolute positions will help get rid of being confused by coordinates. Coordinates transformation is usually a troublesome problem and I hope you can figure it out.

All the global variables and methods will be discussed below in the order that they appear in the program.

2.1 Variables

self.openfile: whether the image shown on screen is read from file or camera.
self.current_dir: current working path.
self.current_bk_dir: current background images folder path.
self.gray: color mode, to use grayscale or not.

self.img_raw: image read directly form camera. Initially, it is set to be 'no_camera.png' stored in /support_file, and if it is not found, an error will occur.
self.img_show: image procced from self.img_raw and used to show on the screen. Initially, it is set the same as self.img_raw.
self.img_release: image to be captured and saved.
self.canvas: no use for now.
self.canvas_blank: a blank canvas used to record drawing points. It has the same size as the image shown on screen. Initially, it is set all to zero, and if the user draws a shape somewhere, the number of the shape will be recorded at corresponding pixels on this canvas.
self.custom_contrast_file: file containing customized contrast parameters.
self.bk_filename: background image file name.
self.background_norm: RGB mean values of background, initially, they all set to 0.
self.bk_error: whether the background is correctly loaded from a file.
self.date: current date used to name the released images.
self.release_count: number of released images in a day, used to name released images.
self.selectbk_folder: folder path used to open background image files.
self.showFileDialog_folder: folder path used to open image files.
self.saveFileDialog_folder: folder path used for "save as" image.
self.calibration: scalebar calibration number.
self.camera_num: camera number.
self.camera: an instance of Camera class.
self.font: the font used to show the text on the image.
self.obtained_plane_para: substrate plane parameter, [A, B, C, D] which satisfies $Ax + By + Cz + D = 0$. Initially it is set to be a horizontal plane [0, 0, -1, 0].
self.substrate_thickness: define the SiO_2 substrate thickness, the default value is '285nm'.
openBK: an action connected to self.select_background() method.
openFile: an action connected to self.showFileDialog to open an image from file.
saveFile: an action connected to self.saveFileDialog to 'save as' an image.
show_scale: an action to choose whether to show scalebar on the screen.
self.show_scale: whether to show scalebar on screen.
self.developer_options: an action to choose whether to use some advanced tools.
capture_BK: an action to capture 100 background images and average them as background.
stage: an action to start stage.
autofocus: an action to start autofocus.
find_focus_plane: an action to start finding the focus plane of the substrate.
scan: an action to start scanning and capturing images on one chip.
layer_search: an action to start searching 2D layers. This is just for testing use.
large_scan: an action to start scanning 9 chips on the holder and searching for thin layers.
theme: an action to change the GUI background color.
restart: an action to restart the software.
calibrate_scale: an action to calibrate the scalebar.
calibrate_coordinate: an action to calibrate the x-y spacing distance of chips.
set_cam_num: an action to set the camera number.
help_contact: an action to show the contact information.

help_about: an action to show the about information.
self.menubar: create a menu bar.
fileMenu: create a menu named ‘File’.
toolMenu: create a menu named ‘Tools’.
settingMenu: create a menu named ‘Setting’.
HelpMenu: create a menu named ‘Help’.
self.developerMenu: create a menu named ‘Developer options’.
self.open_file_button: a button to open images from the file.
self.save_file_button: a button to ‘save as’ images.
self.zoom_in_button: a button to zoom in the image.
self.zoom_draw: whether the zoom-in tool button is checked.
self.zoom_draw_start: whether to start drawing the zoom-in rectangle.
self.zoomed: whether the image has been zoomed in.
self.draw_shape_action_list: record the actions of drawing and erasing.
self.draw_shape_list: record the shapes needed to be drawn on the screen.
self.draw_shape_action_list_for_redo: record the actions that have been undone and then for redoing them.
self.draw_shape_count: record the number of drawing actions to label the shape.
self.straight_line_button: a button to start drawing straight lines.
self.draw_shape_line: whether the straight line button is checked.
self.drawing_shape_line: whether to start drawing straight lines.
self.draw_shape: whether one of the draw-shape buttons is checked.
straight_line_tool_menu: a popup menu to add a checkmenu.
show_distance_checkmenu: choose whether to show distance when drawing straight lines.
self.show_distance: a flag indicating whether to show distance when drawing straight lines.
self.rectangle_button: a button to start drawing rectangle shapes.
self.draw_shape_rectangle: whether the rectangle button is checked.
self.drawing_shape_rectangle: whether to start drawing rectangles.
rectangle_tool_menu: a popup menu to add a checkmenu.
show_side_length_checkmenu: choose whether to show side length of the rectangle.
self.show_side_length: a flag indicating whether to show side length of rectangles.
self.circle_button: a button to start drawing circle shapes.
self.draw_shape_circle: whether the circle button is checked.
self.drawing_shape_circle: whether to start drawing circles.
circle_tool_menu: a popup menu to add a checkmenu.
show_radius_checkmenu: choose whether to show the radius of the circle.
self.show_radius: a flag indicating whether to show radius when drawing circles.
self.eraser_button: a button to start drawing eraser.
self.erase: whether the eraser button is checked.
self.drawing_eraser: whether to start drawing eraser.
self.undo_draw_button: a button to undo drawing actions.
self.redo_draw_button: a button to redo drawing actions.
self.clear_draw_button: a button to clear all the drawings.
self.angle_button: a button to start measuring angle.

self.start_angle_measurement: whether the angle button is checked.
self.base_line: a list to store which line as a base when measuring angle.
self.target_button: a button to start scanning 9 chips and searching for thin layers.
angle_tool_menu: a popup menu to add a checkmenu.
choose_baseline: to choose the baseline for measuring angle.
self.toolbar_file: add a toolbar contains file relevant tools.
self.toolbar_zoom: add a toolbar contains zoom in button.
self.toolbar_drawing: add a toolbar contains drawing relevant tools.
self.toolbar_advanced_tools: add a toolbar contains measuring angle button.
self.toolbar_search: add a toolbar contains layer search button.
self.toolbar: add a blank toolbar.
self.choose_base_line: whether to choose the baseline.
self.sld: to create an RGB slider.
self.button_release: to create a to release button
self.button_live: to create a live view button.
self.button_save_as: to create a 'save as' button.
button_reset_contrast: to create a 'reset contrast' button.
self.combo_custom_contrast: to create a combo box containing customized contrast values.
self.button_saveto: to create a 'save to' button to select images saving folder.
self.release_folder_lbl: to create a label to display the current images saving folder.
cb_gray: to create a checkbox to choose whether to use gray mode.
cb_sb: to create a checkbox to choose whether to subtract background.
self.SB: whether to subtract background.
self.combo_mag: to create a combo box contains different objective magnification.
self.magnification: objective magnification, 5, 10, 20, 50 or 100.
self.crop_left_rate: the ratio of the left part of the image when cropping.
self.crop_right_rate: the ratio of the right part of the image when cropping.
self.cb_crop: create a checkbox to choose whether to crop the black frame of the image.
self.CP: whether to crop the image.
self.cb_tool_distance: to create a checkbox to measure distance.
self.DT: whether the distance tool is checked.
self.DT_draw: whether to start drawing a line to measure distance.
self.distance_lbl: to create a label to display currently measured distance.
self.distance: the current distance value.
self.cb_tool_contrast: to create a checkbox to measure contrast.
self.CT: whether the contrast tool is checked.
self.CT_draw: whether to start drawing lines to measure contrast.
self.contrast_lbl: create a label to display currently measured contrast value.
self.contrast: the current contrast value.
self.pixmap: to create a pixmap to display the image in the self.lbl_main.
self.lbl_main: a label widget to display self.img_show.
self.pw_hist: a widget to display histogram.
self.movie_thread: a thread to acquire camera frames consecutively.
self.live_timer: a timer to refresh images reading from the camera. Currently, it is set as 40ms.

self.openfile_timer: a timer to refresh images reading from file.
self.capture_bk_timer: a timer to capture background images
self.screen_width: current monitor width (in pixels)
self.screen_height: the monitor height
self.initial_window_width: GUI width when started
self.initial_window_height: GUI height when started
self.window_normal: whether the GUI window is in normal size.
self.large_scan_thread: a thread used for canning all the chips on the holder
self.search_property_widget: a widget to customize scanning and searching settings
self.layer_search_thread: a thread used for searching graphene on SiO_2 substrate.
self.choosethickness_widget: a widget to customize searching settings
self.scan_thread: a thread used for scanning.
self.find_focus_plane_thread: a thread used for finding the plane parameter of the substrate.
self.autofocus_thread: a thread used for doing autofocus.
self.stage_thread: a thread used for controlling the stage and scanning.
self.calibrate_coordinates_widget: a widget to calibrate the x-y spacings between samples.
self.custom_contrast_widget: a widget to customize contrast values.
self.delete_custom_contrast_widget: a widget to delete customized contrast values.
self.custom_contrast_dic: a dictionary to store customized contrast values.
self.zoom_draw_able: whether the user can draw the zoom-in rectangle.
self.mouse_x1: mouse position x of the first point.
self.mouse_y1: mouse position y of the first point.
self.mouse_x2: mouse position x of the second point.
self.mouse_y2: mouse position y of the second point.
self.DT_Line: a Line instance to measure the distance of this line.
self.line_num: number of contrast lines that have been drawn. It can be 0, 1, or 2.
self.contrast_line: a list storing two lines to measure contrast.
self.mouse_line_x1: mouse position x after correction of the first point.
self.mouse_line_y1: mouse position y after correction of the first point.
self.mouse_line_x2: mouse position x after correction of the second point.
self.mouse_line_y2: mouse position y after correction of the second point.
self.img_bfDT_width: record the original image width before zooming in. Deprecated.
self.img_bfDT_height: record the original image height before zooming in. Deprecated.
self.r_min: red brightness value. The variable name doesn't match the meaning because of some historical reason.
self.r_max: red contrast value.
self.g_min: green brightness value.
self.g_max: green contrast value.
self.b_min: blue brightness value.
self.b_max: blue contrast value.
self.brightness: global brightness value. This is used for grayscale.
self.contrast_coe: global contrast coefficient. This is used for grayscale.
self.img_raw_not_cropped: the original raw image before cropping.

2.2 Methods

self.get_bk_normalization(): to calculate the RGB mean values of background image and store them in self.background_norm. If the background image is missing, an error will occur and self.bk_error will become True

self.mouse_pos_initial(): to initialize the mouse positions. All set zero.

self.select_background(): to open and select background image used to flatten background.

self.rgb_initialize(): to initialize the RGB brightness and contrast. Set brightness be 0 and contrast be 1.

self.sld_connect(): to connect the slider changing signal to functions that set RGB brightness and contrast values.

self.read_previous_custom_contrast(): to read customized contrast parameters from file. If there is no such file, an error will occur.

self.large_scan(): this method is the top method to run the scanning for all chips on the holder and automatically find layers on chips. It first creates a self.large_scan_thread instance and then display the property setting window, which is connected to self.get_search_property() method by 'confirmed' signal.

self.get_search_property(): to get the layer searching properties, and pass the settings to self.large_scan_thread, then start it.

self.layer_search(): this is for searching graphene on SiO_2 substrate. It is for testing use.

self.choose_thickness_search(): to get substrate thickness for searching, just for testing.

self.scan(): to scan the holder, just for testing.

self.recv_scan_stop(): to receive the signal that scanning has been stopped.

self.find_focus_plane(): this method is for focusing on three different points on the plane and then get the plane parameters. It is for testing use.

self.recv_find_focus_plane_stop(): receive the stop signal from self.find_focus_plane_thread.

self.autofocus(): to automatically focus on one point.

self.recv_focus_stop(): to receive the stop signal from self.autofocus_thread.

self.calibrate_coordinates(): to create a widget that helps users to calibrate the x-y spacing of different samples.

self.show_developer_options(): to show developer options in the menubar.

self.change_theme(): to change the background color of the GUI.

self.custom_contrast(): to customize the contrast values and record them for later use.

self.delete_custom_contrast(): to delete customized contrast values.

self.new_custom_contrast(): to save the newly defined customized contrast values.

self.generate_custom_contrast_items(): to convert the dictionary of customized contrast values to a list, then display it in self.combo_custom_contrast.

self.read_previous_custom_contrast(): to read the customized contrast values from file, save it as a dictionary.

self.show_scale_method(): to turn on or off the scale bar on the screen.

self.accept_new_image(): when users dragging an image from file to the region of self.lbl_main, it will read the file and refresh it on the screen.

self.set_calibration(): to create a widget to set scale calibration value. It shows the current calibration value as a reference and the user can decide whether to save the new value just for once or write the value into the file so that to use it later.

self.recv_new_calibration(): once user set a new scale calibration value and click 'save for once', this method will be called to check whether the new value is valid and if so, replace the old value.

self.recv_save_new_calibration(): if user click 'save for later' after setting a new value, this method will be called to check whether the new value is valid and if so, a warning box will show to let the user decide again whether to save the new value to file since this action is irreversible.

self.calibration_warning(): if the new calibration value is invalid, this method will be called to warn the user.

self.set_camera_number(): to create a widget to set the camera number. The current camera number will be shown as a reference. The camera number must be integers.

self.recv_camera_number(): once the user clicks 'save for once', this method will be called to check whether the camera number is valid and if not, a warning box will be shown. If it is valid, it will restart and rebuild a camera class to use the new camera.

self.recv_save_camera_number(): once user click 'save for later', this method will be called to check whether the camera number is valid and if so, a warning box will be shown to let users confirm again whether they want to save the new camera number in the file since the old camera number will be lost and this action is irreversible.

self.undo_redo_setting(): to reset the undo redo button color, gray icon means this button can't be used currently, and clear the variable self.draw_shape_action_list_for_redo.

self.undo_draw(): first check whether the user has drawn something and if so, undo the last drawing action, then add this add to the redo list self.draw_shape_action_list_for_redo. If there is no more action to undo, set the undo button color to gray.

self.redo_draw(): first check whether the user has undone something and if so, redo the last undone action. If there is no more action to redo, set the redo button color to gray.

self.clear_draw(): to clear all the drawings. Initialize all the drawing shape settings and add a Clear_All action to the self.draw_shape_action_list.

self.draw_shape_initial(): to initialize all the draw-shape settings. Uncheck all the drawing shape buttons and set all the drawing variables to False.

self.draw_straight_line(): once user click the self.straight_line_button, this method will be called. If the button is checked, then start drawing straight lines, otherwise stop drawing.

self.show_distance_method(): this method will be called once user click the show_distance_checkMenu. If it is already checked, then uncheck it and set the self.show_distance variable to False, otherwise check it and the distance will show when drawing straight lines.

self.draw_rectangle(): this method is similar to self.draw_straight_line().

self.show_side_length_method(): see self.show_distance_method().

self.draw_circle(): see self.draw_straight_line() method.

self.show_radius_method(): show radius of the circle. self.show_distance_method() method.

self.erase_shape(): to turn on or off drawing eraser. see self.draw_straight_line().

self.choose_base_line(): this method will be called if choose_baseline is clicked and is used for measuring angle. Then the cursor will become a hand shape.

self.clear_base_line(): this method is to clear the chosen baseline but hasn't been realized.

self.angle_measurement(): this method starts to measure angles between straight lines and the baseline. If the baseline is not set, a warning box will show.

self.graphene_hunt(): this method has been deprecated.

self.capture_background(): a warning box will show first since in my testing, in very few cases capturing background may cause some unstable situations. If the user does want to capture the background, The self.capture_bk_timer will start and capture one image per 40ms, a total of 100 background images will be captured and averaged as a background image. There will be a progress bar during this process to show the capturing progress.

self.capture_bk_start(): once self.capture_bk_timer is started, this method will be called every 40ms and capture a background image. If 100 images have been captured, the timer will stop and it will average the images. I use int16 to lower memory usage but there is still a chance for a memory error to occur during this process. If this happens, the user can choose to try again or abort. If everything goes well, a message box will show to let users confirm whether they want to save the image with the shown name. The background image name must match the current magnification and the cropping condition.

self.contact(): this method will show the 'contact' information.

self.about(): this method will show the 'about' information.

self.acknowledgment(): to show the acknowledgment information.

self.auto_restart(): if there is a camera error occur, the best way to solve it is to restart the program. This method has been deprecated.

self.restart_program(): there is a restart tool in the Tools menu and once it is clicked, this method will the called. A warning box will show to warn users and if the user does want to restart, the restart function will be called to restart the program. This method is very useful when something wrong with the camera connection and usually this can solve most of the problems caused by camera connection.

restart(): this function will restart the whole program.

self.file_dir_test(): to test whether some specific folder is existing or not. This method is for automatically save images into folders named by date and it has been currently deprecated.

self.initial_size(): to initialize the GUI window size. It has been deprecated.

self.save_path_setting(): this method will be called once the 'save to' button is clicked. It is for choosing the folder to save released images. And the last 20 characters of the folder path will show in the self.release_folder_lbl.

self.set_magnification(): this method will be called once the user changes magnification in self.combo_mag. Also, different background images will be chosen to use for background flatten according to different magnifications.

self.is_contrast(): this method will be called once self.cb_tool_contrast checkbox is clicked. If the checkbox is checked, then start measuring contrast, otherwise, stop it.

self.is_distance(): this method will be called once self.cb_tool_distance checkbox is clicked. If the checkbox is checked, then start measuring the distance, otherwise, stop it.

self.tool_initial(): to initialize all the tools. Set all the buttons as unchecked and turn off corresponding flags. This method is used for closing all the other tools before starting to use a tool.

self.initial_measurement(): to initialize the distance and contrast values, i.e., set them to zeros.

self.zoom_in(): to start of close the zoom_in tool. If the image has already been zoomed in, then it can't be zoomed in again.

self.mousePressEvent(): this is to override the method in QMainWindow class. This method is called when the mouse is pressed. It first calculates the x-y limit of the image shown on screen, i.e., the position and geometry of self.lbl_main. If the mouse position is not in this region, the event is ignored. Otherwise, set the second point the same as the first point. The positions get from event.pos() is relative to the GUI window, and self.mouse_pos_correct_rec() or self.mouse_pos_corrct_line() methods will convert them to be relative the self.lbl_main, i.e., relative to the image shown on screen.

- If the zoom-in tool is turned on, then start drawing the zoom-in rectangular region.
- If measuring distance tool is turned on, then create a Line instance and start drawing a straight line to measure distance.
- If measuring contrast tool is used, then create a Line instance to measure contrast. If there are already two contrast lines, clear self.contrast_line and then add a new contrast line. Then start drawing the line.
- If to draw straight lines, add a Line instance to self.draw_shape_action_list and start drawing straight lines, also reset the undo-redo settings.
- If to draw rectangle shape, add a Rectangle instance to the drawing shape action list, then start drawing rectangles, also reset the undo-redo setting.
- If to draw circle shape, add a Circle instance to the drawing shape action list, then start drawing circles, also reset the undo-redo setting.
- If to erase shapes, add an Eraser instance to the drawing shape action list, then start drawing the eraser.
- If to choose the baseline to measure angles, create a temporary Point instance to record a

circle region with radius 7 pixels around the mouse, then find line across the region. x_temp and y_temp record the positions around the mouse, self.canvas_blank record the positions of shapes drawn on the image. If there is a line around the mouse, first check whether there is already a baseline chosen before and if so, change it back to a normal line, then change the newly chosen baseline color to yellow and change the width to 2.

- If the mouse right button is clicked, then zoom out the image to the original size.

self.mouseMoveEvent(): this is to override the method in QMainWindow class. This method is called when the mouse is moving. If the mouse position if out of the image, then the event is ignored, otherwise, drawing different shapes according to different situations.

- If to zoom in, then draw a zoom-in rectangle to select the wanted zoom-in region.
- If to measure distance, then refresh the second point of the self.DT_Line.
- If to measure contrast, then refresh the second point of the last contrast line.
- If to draw a shape (straight line, rectangle, circle, eraser), then refresh the second point of the last shape.

self.mouseReleaseEvent(): this is to override the method in QMainWindow class. This method is called when the mouse button is released.

- If to zoom in, first check whether the rectangle is valid, if so, then self.zoomed is set to be True, i.e., successfully zoomed, otherwise, just ignore.
- If finished drawing the eraser, then stop drawing it and delete the Eraser instance.

self.mouse_pos_correct_line(): to correct the mouse position of a line. Originally the mouse position is relative to the GUI window, and this method will convert the position to be relative to self.img_raw, i.e. the raw image. It will first convert the positions to be relative to the self.lbl_main, i.e., the image shown on the screen. Then it checks whether the position is within the image, and if not, change it to the border of the image. Next, it will call self.mouse_pos_show2raw method to convert the position to be relative to the raw image, i.e., the absolute position.

self.mouse_pos_show2raw(): to convert the mouse position from being relative to self.img_show to be relative to self.img_raw by considering the zoom-in and cropping effects.

self.mouse_pos_correct_rec(): this is specially for changing mouse positions of zoom-in rectangle. To draw a zoom-in rectangle, the first point will be at the top left corner and the second point will be at the bottom right. Attention, this method just convert the position to be relative to the self.lbl_main, i.e., the image shown on the screen, don't be confused!

self.mouse_pos_correct(): to convert the mouse position from being relative to the GUI window to be relative to self.lbl_main, i.e., the image shown on screen. The difference of former and later is the toolbar and menu bar height, and it is also caused by centering the self.lbl_main widget.

self.is_Crop(): if the 'crop' checkbox is checked, then start cropping the image. Note that crop can’t be used if the image is already zoomed in, otherwise a warning box will show.

self.is_SB(): if the ‘subtract background’ checkbox is checked, then start flattening the background. If the background image file is missing, a warning box will show.

self.is_gray(): if the ‘gray’ checkbox is checked, then use grayscale to show the image.

self.select_background(): to select a background image from file as reference to flatten background. This method has not been used for a long time and it is almost deprecated.

self.set_r_min_value(): to set the red brightness value. This method is called when the RGB slider is changed. Note the method name doesn’t match its meaning for historical reasons.

self.set_r_max_value(): to set the red contrast coefficient.

self.set_g_min_value(): to set the green brightness value.

self.set_g_max_value(): to set the green contrast coefficient.

self.set_b_min_value(): to set the blue brightness value.

self.set_b_max_value(): to set the blue contrast coefficient.

self.set_brightness(): to set the global brightness value. Set the RGB brightness value equal to the global brightness, then set these values.

self.set_contrast(): to set the global contrast coefficient. Set the RGB contrast coefficients equal to the global contrast coefficient, then set these values.

self.saveFileDialog(): once ‘save as’ is clicked, this method will be called to open a file dialog and users can choose their favorite path to save the image.

self.showFileDialog(): once ‘open file’ is clicked, this method will be called to open a file dialog to open an image file.

self.rgb_initialize(): to initialize the brightness values and contrast coefficients. 0 means no brightness shift and 1 means no contrast expansion or compression.

self.set_rgb_value(): to set the RGB brightness values and contrast coefficients to the RGB slider.

self.sld_connect(): to connect the RGB slider changing signal to different methods.

self.change_contrast(img): to change the brightness and contrast of the input image, then output the changed image. If the input image has RGB three channels, then change the RGB brightness and contrast separately. And if it only has a gray channel, just change the gray. go_fast() is a highly optimized function to do the calculation very fast.

self.release_sound(): to play a shutter sound after capturing an image. Before this software has been developed, there is a shutter sound when release images. And some people feel unsafe if there is no sound. Therefore this method simulates such sound.

self.update_release(): to capture images. It will capture 5 frames and average them as the result to lower the noises. The released image will be saved to self.release_folder.

self.start_live(): to start the self.movie_thread and read from camera continuously. Then start the self.live_timer to refresh the image on screen. Currently, the refresh rate is 40ms, i.e., 25fps, which is good enough for human eyes.

self.update_live(): to update the image shown on screen.

self.refresh_show(): to update self.img_show, i.e., the image used for showing on the screen. If self.open_file is True, then read image from file, otherwise read from camera. First resize the showing image to the size of the GUI window. Other processing depends on other flags.

- If to crop the black frames on the left and right side, crop the showing image together with the raw image.
- If to draw the zoom-in region, draw a rectangle.
- If self.zoomed flag is True, then zoom in the image.
- If to subtract background, first resize the background to match the size of the showing image, then divide the showing image by the background.
- If to use grayscale, then change the color of the showing image from BGR to gray. Then change the brightness and contrast of the image.
- If to measure distance, first change back the mouse position from absolute position to be relative to the image showing on the screen, then draw lines on the screen and calculate the distance of the line, also showing the distance.
- If to measure contrast, first change back the mouse position from absolute position to be relative to the self.img_show, i.e., the image showing on the screen, then calculate the contrast of the two lines and show it. Also, draw the contrast lines.
- If to draw an eraser, first get the eraser parameters from the action list, then change the mouse position and draw the eraser. Find the shapes around the eraser and erase them.
- If to draw shapes, generate a shape list and draw them on the self.img_show and record their positions on self.canvas_blank to locate them.
- If to measure angles between straight lines and the baseline, calculate the angles and show them.
- If to show the scale bar on the image, then draw the scale bar and show it.

self.draw_graduated_scale(): to draw scalebar on self.img_show. The scale mark depends on the size of the image.

self.calculate_distance(): to calculate the distance of a straight line.

self.show_angle_value(): to calculate the angle between baseline and straight lines.

self.generate_draw_shape_list(): to generate a shape list to draw on the screen.

self.find_eraser_num(): to find the shapes around the eraser and record them to erase.

self.draw_shape_canvas(): to draw shapes on the screen and record the shape position for the eraser to find.

self.mouse_pos_ratio_change_rec(): to convert back the mouse position from absolute position to be relative to self.img_show. This method has been deprecated.

self.mouse_pos_ratio_change_line(): to convert back the mouse position from absolute position to be relative to self.img_show.

self.mouse_pos_ratio_change_done(): this method has been deprecated.

self.show_on_screen(): to change the array format image to QPixmap format then display it on screen. Also, draw a histogram of the current image. Then adjust the window size.

self.refresh_raw(): to refresh the raw image. This method is used to save the image file since when capturing images, raw images are used instead of the showing image to improve the saved image quality.

self.draw_shape_canvas_for_raw(): to draw shapes on the canvas, then add the canvas to the raw image.

self.add_canvas_to_raw(): to add the canvas to the raw image.

self.draw_hist(): to draw a histogram of the showing image. Normalize the hist so that the maximum value is 1000.

self.changeEvent(): this is to override the method in the QMainWindow class. When maximize or restore the window size, this method is called to change the GUI window size correspondingly.

self.display_resize(): to resize the showing image to match the GUI window size.

self.np2qimage(): to change the image format from NumPy array to QImage.

class MovieThread(): a thread to acquire frames form camera continuously. This is a separate thread to avoid clog.

3 Threads.py

This is also one of the most important files. It contains several thread classes to control the x-y stage as well as the focus. ‘StageThread’ controls Prior software to control x-y stage, ‘AutoFocusThread’ controls the BX2 software to adjust focus, ‘Set_Stage_Focus’is a bundle of ‘StageThread’ and ‘AutoFocusThread’, ‘FindFocusPlane’ will do autofocus on three different points and calculate the plane parameter, ‘Scan’ will scan one chip and capture images, ‘LayerSearchThread’ will start a thread to find layers, ‘LargeScanThread’ will scan 9 chips on the sample holder and do the searching.

To control other software and then indirectly communicate will the stage and focus, I use win32gui to find the handle of the windows and spy++ would be a helpful tool to do this.

3.1 class StageThread(camera, pos = [0, 0])

This thread will control the Prior Terminal, a software to communicate with the stage, and therefore control the stage. The input arguments ‘camera’ and ‘pos’ are useless and will be deprecated. It is inherited from QThread and will use a thread to run to avoid clog. This class can be replaced by directly communicate with the stage using COM port to make possibly faster communication.

3.1.1 Variables

self.camera: it has been deprecated.
self.pos: the position stage will move next, relative to the current position.
self.absolute_postion: the stage absolute position, relative to a fixed zero point.
self.error: an error flag.
self.hwnd: the handle to ‘PriorTerminal’ window.
self.prior_edit_text_hwnd: the handle to the edit window.
self.prior_edit_text: an instance of EditWrapper to send messages to the stage.
self.prior_list_box_hwnd: the handle to the list box window.
self.prior_list_box: an instance of ListBoxWrappe to read from list box.
self.manubar_hwnd: the handle to the menu bar.

3.1.2 Methods

self.prior_terminal_initial(): initialize the class. It will first find the Prior Terminal and if not, an error will occur. Then find the edit, list box, and menu bar handle. Also, get the current absolute position to test whether the communication with the stage is in good condition.

self.prior_pause(): after sending a command to the stage, there would be some time for the stage to execute it and then return ‘R’ to indicate that the stage has stopped. Therefore this method is called after running the stage and it is for waiting and checking whether the stage is stopped and if so, the next command can be sent. It will wait for at most 4 seconds, otherwise, a time out warning message will show.

self.run(): to move the stage to the self.pos position relative to the current position. It will call self.prior_pause and once the stage has stopped, a 'stop' message will be sent out through self.stop().

self.get_absolute_position(): to get the current absolute position of the stage by sending a 'P' command to the stage.

self.get_abs_pos_from_text(): to change the string format of position into int format.

self.stop(): to send out a 'stop' signal when the stage has moved and stopped.

3.2 class AutoFocusThread(camera, Fine = False)

This thread will control the BX2, a software communicating with the BX60 microscopy, and therefore control the focus. The input argument 'camera' must be a camera class and 'Fine' indicate whether to do autofocus super finely.

3.2.1 Variables

self.camera: a camera class used for capturing images and then calculate the clearness.
self.Fine: a flag indicating whether to do autofocus super finely.
self.gradient_mean: to store the image gradient values.
self.bx2_position: current focus position.
self.focus: no use.
self.error: an error flag.
self.focus_hwnd: the handle to the 'Focus' child window.
self.soft_key_hwnd: the handle to the 'Soft Key' window.
self.soft_key_1_button: a ButtonWrapper instance of button 1 in the Soft Key panel, different buttons are set to different functions.
self.soft_key_2_button: a ButtonWrapper instance of button 2 in the Soft Key panel.
self.soft_key_3_button: a ButtonWrapper instance of button 3 in the Soft Key panel.
self.soft_key_4_button: a ButtonWrapper instance of button 4 in the Soft Key panel.
self.soft_key_5_button: a ButtonWrapper instance of button 5 in the Soft Key panel.
self.bx2_edit_hwnd: the handle to the focus edit window.
self.bx2_edit: the Ediwrapper instance of the focus edit.
self.bx2_position_hwnd: the handle to the current focus position window.
self.bx2_position: current focus position.
self.up_height_hwnd: the handle to the button that raises the focus.
self.up_height_button: a ButtonWrapper instance of the button that raises the focus.
self.down_height_hwnd: the handle to the button that lowers the focus.
self.down_height_button: a ButtonWrapper instance of the button that lowers the focus.

3.2.2 Methods

self.BX2_initial(): to initialize the AutoFocusThread. First try to find the window named 'BX2' and if it is not existing, then an error will occur, otherwise, get the handle of it. Next, find the child window named 'Focus' and get its handle. Then find the 'Soft Key' window. If it is not existing, then open it and find it again. If it is now open, get the handles to the buttons on the 'Soft Key' panel. After that, check whether the 'focus' box is checked and if so, get the handles of the edit, position, raising focus and lowering focus child windows.

self.run(): to do autofocus. First check whether there is an error and if so, abort the thread, otherwise, there are three steps to do autofocus.

- First rising the focus by 100um, then calculate current clearness. Next lower the focus for 10 steps, 20um for each step and calculate clearness for each step. Then move the focus to the clearest position.
- Finely adjust the focus. Rise the focus by 20um and lower the focus for 4 steps, 10um for each step, also calculate the clearness during the process. Then move the focus to the clearest position.
- If to adjust super finely, then rise the focus of 5um and then lower it for 5 steps, 2um for each step. This is for autofocus under 20x magnification.

After autofocus, try to get the current focus position, then send out a stop signal.

self.cal_clearness(): to calculate the clearness by converting the image into grayscale and calculating the Laplacian of the image, i.e., the second derivative which reflects the sharpness of the shape edges on the image.

self.up_coarse(): to coarsely rise the focus. It has been deprecated.

self.down_coarse(): to lower the focus by 20um.

self.down_fine(): to lower the focus by 10um.

self.down_super_fine(): to lower the focus by 2um.

self.go_height(z): to move the focus. The input argument 'z' indicates how much to move, in um units, and it must be an integer.

self.click_go_height_wait(h): to click the button 5 on 'Soft Key' panel and then move the focus by 'h' um. Then wait for the move to be done.

self.stop() to emit a 'stop' signal when autofocus is finished.

3.3 class Set_Stage_Focus(camera)
This thread is a bundle of stage and focus thread. The input argument 'camera' is a Camera class.

3.3.1 Variables
self.camera: a Camera class.
self.plane_points: a useless variable.

self.stage_running: a flag to indicate whether the stage is moving.
self.autofocus_running: a flag to indicate whether the autofocus thread is running.
self.stage: a StageThread instance.
self.autofocus: an AutoFocusThread instance.

3.3.2 Methods

self.initUI(): to initialize the thread by creating stage and autofocus thread instances. Then connect the 'stop' signal to stop functions.

self.stage_stop(s): when receiving the 'stop' signal from StageThread, set the stage running flag to be False.

self.autofocus_stop(s), when receiving the 'stop' signal from AutoFocusThread, set the autofocus running flag to be False.

self.wait_stage(): to wait for the stage to stop. The maximum waiting time is 4 seconds.

self.wait_focus(): to wait for the autofocus thread to top. The maximum waiting time is 30 seconds.

3.4 class FindFocusPlane(camera)

This thread is to find plane parameters of the substrate. It will focus on three points and thus determine the plane.

3.4.1 Variables

self.para: plane parameters A, B, C, D, which satisfy $Ax + By + Cz + D = 0$. The initial value [0, 0, 1, 0] corresponds to a perfectly horizontal plane.
self.plane_points: three focus points positions

3.4.2 Methods

self.run(): to do autofocus on three non-collinear points and then calculate the plane parameters. The relative position of three points are [0, 0], [0, 5000] and [5000, 5000]. For each point, first send a command to move the stage to the point, then wait for the stage to stop. Next do autofocus and wait for it to stop. After focusing on three points, calculate the plane parameters and move back the stage and focus to the starting point. Then emit a 'stop' signal.

self.stop(): to emit a 'stop' signal when self.run() is done.

3.5 class Scan(camera, plane_para = [0, 0, 1, 0], magnification = '5x')

This method is to scan the chip and capture images. The input argument 'plane_para' is the substrate plane parameter and the 'magnification' can be '5x', or '10x' or '20x'. It will automatically change the focus when moving the stage to stay focusing on the sample.

3.5.1 Variables

self.plane_para: the plane parameters of the substrate.
self.magnification: the objective magnification when scanning.
self.rotate_90: a flag indicating whether to rotate 90 degrees when scanning.
self.date: current date, to name the image.
self.month: current month.
self.day: current day.
self.year: current year.
self.save_folder: the path to save the image.
self.save_count_dir: a counter used to make a saving folder.
self.save_count: a counter for the saved images.
self.index: to name the image file.
self.current_dir: current working path.
self.bk_filename: the background image file path.
self.capture_still: to capture a still image since there might be shaking when moving the stage. However, it has been deprecated.
self.threshold: the threshold for the difference of adjacent captured image. If the difference is smaller than this threshold, then the image is regarded as a still image, otherwise wait for the stage to be stable.

3.5.2 Methods

self.get_background(): to read the background image from file.

self.run(): to start scanning. First define the step size and moving steps according to different objective magnification, then read the background image for current magnification. Then calculate the x and y focus difference, and scan along the smaller difference direction to reduce adjusting focus times since moving focus would cost more time and only the height is accumulated to some value the focus would move. Find the correct saving folder and write a black image as a starting point. Then start moving the stage and calculate the height difference. If the total difference is large than 3um, then move the focus. Finally emit a ‘stop’ signal when finished.

self.move(pos = [0, 0]): to move the stage to ‘pos’ position relative to the current position.

self.focus(h = 0): to move the focus.

self.recv_capture_done(): it has been deprecated.

self.get_index(total_x, total_y, x, y): to get the image label index. The input arguments ‘total_x’ and ‘total_y’ are the total x and y direction steps. And ‘x’ and ‘y’ are the current steps.

self.capture_save(): to capture an image and save it into a file. Capturing 2 adjacent images and calculate their difference. It is smaller than the threshold, then save it.

self.int2str(): to convert int number to string.

self.stop(): to emit a 'stop' signal.

class CaptureStill()
The functions of this class have been moved to self.Scan.capture_save() and therefore has been deprecated.

3.6 class LayerSearchThread(thickness, material, filepath, resultpath):
This class will call layer_search.py or layer_search_TMD.py to search 2D thin layers. The input argument 'thickness' indicate the thickness of SiO_2, it can be '285nm' or '90nm', 'material' can be 'graphene' or 'TMD', 'filepath' is the input images path and the 'resultpath' is the path to output the resulting images.

3.6.1 Variables
self.thickness: substrate SiO_2 thickness.
self.material: material to find, graphene, or TMD.
self.filepath: the path from which to read image files
self.pathDir: image files names
self.outpath: temporary files saving path
self.finished_count: finished images number.
self.resultpath: output image saving path.

3.6.2 Methods
self.run(): to start finding 2D layers from image files in the image folder. The while loop will run forever unless all the image in the folder has been searched. The criterion is that there is no new image within 2 minutes, i.e., the thread will stop 2 minutes later after the scanning finished. Only images with target layers will be output to designated folder and others would be abandoned.

self.draw_position(filename, img_out): to draw relative position of the image. The position is the image captured position relative to the substrate. The input argument 'filename' is the image's name, and 'img_out' is the image needed to be drawn position.

3.7 LargeScanThread(thickness, magnification, material)
This thread will call FindFocusPlane, Scan and LayerSearch threads to scan for 9 chips on the sample holder. The input arguments are similar to that of LayerSearch thread.

3.7.1 Variables
self.camera: a Camera class
self.para: plane parameters.
self.find_focus_plane_running: a flag to indicate whether the FindFocusPlane thread is running.
self.scan_running: a flag to indicate whether the Scan thread is running.

self.thickness: the thickness of the SiO_2 substrate.
self.magnification: objective magnification.
self.material: material to find, graphene, or TMD.
self.stage_focus: a Set_Stage_Focus instance.
self.find_focue_plane: a FindFocusPlane instance.
self.scan: a Scan instance
self.layer_search: a LayerSearchThread.
self.current_dir: current working path.
self.xy_step_file: the file containing x-y spacing between different chips.
self.x_step: x spacing between chips.
self.y_step: y spacing between chips.

3.7.2 Methods

self.run(): to start scanning. First set some parameters in LayerSearchThread. Then start finding the focus plane for one chip and scan it. Repeat the action for 9 chips.

self.stop_find_focus_plane(): to receive the FindFocusPlane stop signal and set the running flag to be False

self.stop_scan(): to receive the Scan stop signal and set the running flag to be False.

self.wait_for_find_focus_plane(): to wait for the FindFocusPlane thread to stop by checking whether the running flag is False.

self.wait_for_scan(): to wait for Scan thread to stop

4 layer_search.py

This is the core file to do the searching. This file is mainly based on OpenCV-python. It will be explained line by line in detail.

The input arguments are 'filename', i.e., from where to read the image file and 'thickness', the SiO_2 substrate thickness. First read the image file to img0, then make two copies of it for later use. isLayer is a flag to indicate whether there are target layers in this image.

Split the image into BGR channels then calculate the histograms separately. Convert the raw image into grayscale and calculate the histogram of the gray image. Then get the highest peak positions of BGRY channels separately as background colors. The gray channel will be used to calculate the contrast value later.

The threshold is set to be 5 color values smaller than the gray peak position. Then make a binary image by setting all the pixels that have gray values higher than the threshold be black and that have gray values lower than then threshold be white, since the graphene on the SiO_2 substrate will absorb light and will be darker than the background. Then create a kernel to do the opening operation to erase a lot of small noises pixels. And create another kernel to do the closing operation to connect the white region. After this, find the contours of all the white regions.

Create a list to store contours with large areas. Calibration of the pixel with the real size is necessary to set the area threshold. Currently, it is set to be 300 pixel^2. Then draw all the contours to a temporary image to show.

For every large contour, create a mask to take it out from the original image and do the following processing. Calculate the histogram or BGR channels separately. The check if the color is similar to graphene and if not, abandon this contour and go for next.

Convert the color of the picked out region into grayscale. Then subdivide the region since it might contain different numbers of layers. Subdivide it into small homogeneous region by using find_real_peak_pos() function.

To subdivide the large region, first increase the contrast to make the color value reach the highest 255. Then calculate its histogram. Next, Abstract all the maxima and their positions, and calculate the average height. Change the peaks and positions into list format and if there is no peak, just return [0]. Otherwise, find the maxima of the maxima. This is for trying to reduce the noises and find real maxima. If the maximum is higher than the average height, then it is viewed as a real peak, it is abandoned if not. Then return the maxima positions and the image with increased contrast.

After obtaining the small peaks in the histogram of the large region, find the boundaries of these peaks by calling find_peak_boundary(). If the two peaks are separated more than 10 color values, then the boundaries are set to be positive/minus 5.5 of the peak. If the distance between the two peaks is smaller than 10 but larger than 5, then the boundary is set to be the middle of two peaks. And if the two peaks are too close, the second peak is abandoned.

After finding the peaks and peak boundaries of the histogram of the large region, divide the large region into small segments and calculate the contrast of them separately by calling the find_contours() function. For every color boundaries pairs, pick out the corresponding small segments and then do the opening and closing operations to reduce some noise. Then find small contours corresponding to such color range and abandon some very small segments by setting an area threshold, also abandon some segments with color being quite different from graphene. Calculate the total area of segments in such color range and if it is larger than another area threshold, keep it and calculate the contrast. To calculate the contrast precisely, the local contrast is used. First create a mask to pick out all the small segments within the color range then calculate the mean color value of these segments. Then calculate the local contrast by calling calculate_local_contrast() function.

To calculate the local contrast, first obtain the bounding rectangle and enlarge it a little bit to include more background. Then Calculate the histogram of the region within the rectangle.

Then find the background color by the following method. First calculate the maximum value of the histogram, then pick out the histogram from the sample color value to the global background value plus some error-tolerant value. Find the local maxima within this region and find the maximum maxima as the local background color value. Then calculate the local contrast by dividing the difference of local background color value and sample color value by the local background color value.

If the local contrast value is within the range of [0.028, 0.30], then these segments are regarded as thin graphene layers, otherwise they are abandoned.

After acquiring homogeneous segments and their local contrast, use mask to pick them out and calculate the bounding rectangle. Then draw a slightly enlarged bounding rectangle to indicate the position and put the text of contrast near the rectangle. If the program finds thin layers, the isLayer flag is set to be True.

5 layer_search_TMD.py

This is a specialized version of layer_search.py to find TMD materials. Most of this file is the same as layer_search.py but some parameters are changed to find TMD, also there are some specialized optimizations for TMD to reduce false-positive results.

6 auxiliary_func.py

This file contains several auxiliary functions used in other files.

go_fast(a, b, c): to change the image color. a is the input image, b is the shift value and c is the multiplier. This function use numba.jit to accelerate.

background_divide(a, b, c): to divide the background. a is the input image, b is the background image and c is the mean value of the background image. This function also uses numba.jit to accelerate.

matrix_divide(a, b): to divide the matrix a by b. This function use numba.jit to accelerate.

get_folder_from_file(filename): to get the parent directory of the input file.

float2uint8(img_aver): to convert the input matrix format from float to np.uint8, i.e., the image format used in OpenCV.

calculate_contrast(matrix, x1_1, y1_1, x1_2, y1_2, x2_1, y2_1, x2_2, y2_2): to calculate the contrast of the color values on two lines. The input argument 'matrix' is the image. The first line is defined by x1_1, y1_1, x1_2 and y1_2, and the second line is defined by x2_1, y2_1, x2_2 and y2_2. First find the pixel positions on the two lines, then get the color values of these pixels. The gray color value is calculated by Y = R*0.299 + G*0.587 + B*0.114.

record_draw_shape(blank, x_pos, y_pos, num): to record the drawings on the blank canvas. The input argument 'blank' is a canvas as a recording medium. 'x_pos' and 'y_pos' are shape positions. And 'num' is the shape number to be recorded.

calculate_angle(pos11, pos12, pos21, pos22): to calculate the angle between two lines.

np2qimage(img): to convert the array format image into QImage format.

7 auxiliary_class.py

This file contains several auxiliary widgets class used in other files.

DropLabel: it defines a drop label to accept dragging and dropping images from local files.

RGB_Slider: it creates 8 sliders to adjust the global brightness, contrast coefficient and RGB brightness and contrast coefficient separately.

ProgressBar: it creates a progress bar widget. It is used when capturing 100 background images.

CameraNumEdit: it creates an editing widget to edit the camera number. It also shows the current camera number for reference.

CalibrationEdit: it creates an editing widget to edit the scale calibration value. It also shows the current scale calibration value for reference.

CalibrateCoordinate: it creates a special widget to calibrate the x-y spacing between different chips. It contains step by step instructions.

CustomContrast: it creates a widget to customize the contrast. It is mainly used for setting the name of the customized contrast.

DeleteCustomContrast: it creates a widget to delete the customized contrast.

ThicknessChoose: it creates a widget to choose the material and the thickness of the substrate. It has been deprecated.

SearchingProperty: it creates a widget to choose the material, the thickness of the substrate and the objective magnification used for searching layers.

8 Camera.py

This file defines a Camera class to call the camera and read frames from the camera.

8.1 Variables

self.cam_num: camera num, must non-negative integer.
self.cap: an instance of cv2.VideoCapture().
self.current_dir: the current working path.
self.initial_img_error: a flag to indicate whether reading the initial image is properly done.
self.camera_error: a flag to indicate whether there is a connection error with the camera.
self.last_frame: current camera frame.

8.2 Methods

self.initial_last_frame(): to initialize self.last_frame. Read the image from file an if there is no such file, an error will occur and the self.last_frame will be a black image.

self.initialize(): to initialize self.cap.

self.get_frame(): to read frames from camera. If there is something wrong when communicating with the camera, an error will occur and the current camera frame will be initialized.

self.acquire_movie(): to acquire camera frames continuously.

self.set_brightness(value): to set the camera brightness. This is useful for a USB camera. This is just a testing method has been deprecated. Be cautious when using it.

self.get_brightness(): to get the current camera brightness value.

self.close_camera(): to close the camera properly.

self.get_folder_from_file(filename): to get the parent directory of the input file.

9 drawing.py

This file contains several shape classes with similar structures.

Line(): to define a line class. x1, y1, x2, y2 record the absolute positions of the starting and ending points of the lie, x1_show, y1_show, x2_show, y2_show record the relative positions for showing on the screen. The variable 'self.num' records the number label of the line, 'self.prop' record the property of this class, and 'self.show_distance' is a flag indicating whether to show the distance of the line. The method self.pos_refresh() is to refresh the current position.

Rectangle(): to define a rectangle class.

Circle(): to define a circle class.

Eraser(): to define an eraser class.

Point(): to define a single point class.

Creal_All(): to define a ‘clear all’ class.

10 BresenhamAlgorithm.py

This file mainly uses the Bresenham algorithm to calculate the points on the line, circle and rectangle, or points in the circle. It can be easily understood if you are familiar with the Bresenham algorithm and part of the codes in this file are just copied from the internet. Therefore it is unnecessary to explain too much for this file.

11 support_file

In the support_file folder, this is a folder named background, which contains background images captured under different magnifications and cropping conditions. Also, the support_file folder contains icon images used in the GUI and several txt files containing user-customized data.

calibration.txt: scalebar calibration value

camera.txt: camera number, must be a non-negative integer, usually it is set to be 0 or 1, depending on the number of cameras connected to the computer.

coordinate_calibration.txt: x-y spacing of different samples.

custom_contrast.txt: user-customized contrast parameter, global brightness, global contrast, red brightness, red contrast, green brightness, green contrast, blue brightness, blue contrast.

saveto.txt: containing customized folder path to save released image.