Bridge Gap Detection With OpenCV-Python and PyRealsense2

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 Main Namespace Reference

Variables

- v = Viewer.Viewer()

  Initialisation of viewer object.
- e1 = cv2.getTickCount()

  Tick count 1 for calculating framerate of system.
- e2 = cv2.getTickCount()

  Tick count 2 for calculating framerate of system.
- tuple time1 = (e2 - e1) / cv2.getTickFrequency()

  Total time taken for function to execute.
- control = v.get_drone_control()

  Control command to be sent to drone.
- phase = v.deployment_phase

  Flight phase of the drone.

4.1.1 Variable Documentation

4.1.1.1 control

```
Main.control = v.get_drone_control()
```

Control command to be sent to drone.

4.1.1.2 e1

```
Main.e1 = cv2.getTickCount()
```

Tick count 1 for calculating framerate of system.

4.1.1.3 e2

```
Main.e2 = cv2.getTickCount()
```

Tick count 2 for calculating framerate of system.

4.1.1.4 phase

```
Main.phase = v.deployment_phase
```

Flight phase of the drone.

4.1.1.5 time1

```
tuple Main.time1 = (e2 - e1) / cv2.getTickFrequency()
```

Total time taken for function to execute.

4.1.1.6 v

```
Main.v = Viewer.Viewer()
```

Initialisation of viewer object.

4.2 Viewer Namespace Reference

Classes

- class Viewer

  Class for the Realsense Viewe.

Chapter 5

Class Documentation

5.1 Viewer.Viewer Class Reference

Class for the Realsense Viewe.

Public Member Functions

- def __init__ (self)

  Contructor Function.
- def adapt_depth_clipping (self, depth_image)

  Function to implement adaptive depth clipping.
- def get_move_dir (self, lr_dir)

  Function to get the movement direction in the drone's flight phase.
- def get_drone_control (self)

  Function to get the drone control commands, dependent on the flight phase - either flying, or deploying.
- def canny_edge (self, image)

  Function to process the colour image.
- def check_z_orientation (self, centre)

  Function to check the orientation about the Z axis of the object being viewed.
- def get_dist_to_pillar (self, rect)

  Get the distance between the Realsense camera and the 'bridge support pillar'.
- def find_contours (self, edges)

  Function used to find the contours of the object being viewed.
- def get_deployment_dir_to_move (self, c_x, c_y)

  Gets the direction for the drone to move in the deployment phase.
- def get_deployment_coords (self, tl, tr, bl, br, gap_rect, full_rect)

  Gets the coordinates of the internal contour in the deployment phase.
- def scale_internal_contour (self, centre, dims)

  Function used to verify the points of the internal contour for deployment.
- def verify_deployment_area (self, shrunk_points)

  Function to verify whether the edges of the internal contour cross any point of the surrounding support structure.
- def separate_contours (self, contours, hierarchy)

  Takes in the full contour list, alongside heirarchy and separates these into internal and external contours.
- def draw_roi (self, box, theta)

  Draws a region of interest (ROI) above the pillar if no 'gap' is detected.

- def shift_line_y_t (self, point1, point2, avg_ty)

  Shifts the top line of the internal gap contour in the nevative y direction.
- def shift_line_y_b (self, point1, point2, avg_by)

  Shifts the bottom line of the internal gap contour in the positive y direction.
- def shift_line_x_r (self, point1, point2, avg_rx)

  Shifts the right hand line of the internal gap contour in the positive x direction.
- def shift_line_x_l (self, point1, point2, avg_lx)

  Shifts the left hand line of the internal gap contour in the nevative x direction.
- def deproj_to_pt (self, point1, point2)

  Deprojects the pixel data into 3D point data with reference to the camera position.
- def update (self)

  Update function which runs all of the required functionality of the script and draws all required objects to the screen.
- def get_x_dist (self, mid, wh, theta)

  Function to get the direction that the drone is required to move in, only in the x-direction.
- def get_colour (self, x, y)

  Function to get the BGR colour values at specified pixel.
- def get_line_eq (self, ln1, ln2)

  Function to calculate the equations of two lines.
- def intersection (self, ln1, ln2)

  Uses Cramer's Rule to find the intersection point of two lines.
- def show_window (self)

  Function used to show the 'Realsense' window and the 'Edge' window 'Realsense' - Shows processed image with all drawn contours and eliminated background, next to the colourised depth stream.

Public Attributes

- pipeline

  PyRealsense Object Initialiser.
- screen_width

  Global integer for screen width (default=848 pixels)
- screen_height

  Global integer for screen height (default=480 pixels)
- profile

  Initialises the profile for the depth and colour streams.
- align

  Alignment object to align the depth stream with colour stream.
- colouriser

  Initialise colouriser for depth stream.
- decimation

  Initialise decimation filter object.
- spatial

  Initialise spatial filter object.
- temporal

  Initialise temporal filter object.
- hole_filling

  Initialise hole filling filter object.
- depth_to_disparity

  Transform depth stream to disparity map.
- disparity_to_depth

  Transform disparity map to depth stream.

- depth_scale

  Initialise depth scale object for converting raw distances into different units (default=millimetres.)
- active_contours

  List containing all of the active contours, used to verify whether or not the contours are usable.
- clipping_dist

  Initialise clipping distance with use of depth scale.
- verified_deployment_area

  Boolean - used to identify whether or not the deployment area is safe and deployable.
- deployment_phase

  Boolean - used to identify whether or not the drone is in the deployment phase.
- pre_deploy_dir

  Dict containing key: movement direction, value: movement distance (pixels).
- direction_to_move

  String containing directions to move.
- distance_to_move

  Float value containing distance to move.
- distance_from_pillar

  Float value containing distance from 'pillar'.
- deployment_area_coordinates

  List containing tuples of deployment area coordinates.
- deployment_dir_to_move

  Dict containing key: movement direction, value: movement distance (millimetres).
- internal_contour

  Boolean - identifies if the internal contour is active.
- external_contour

  Boolean - identifies if the external contour is active.
- contour_shrink_sf

  Float value containing scale factor to shrink deployment area by.
- deploy_tl

  Tuple containing coordinates of top left corner of deployment area.
- deploy_tr

  Tuple containing coordinates of top right corner of deployment area.
- deploy_bl

  Tuple containing coordinates of bottom left corner of deployment area.
- deploy_br

  Tuple containing coordinates of bottom right corner of deployment area.
- deployment_area_distances_mm

  List containing the width and height of deployment area in millimetres.
- aligned_depth_frame

  Object containing depth frame data of the aligned depth frame (aligned with colour frame).
- colour_frame

  Object containing colour frame data.
- colourised_filtered_depth

  Array containing the depth frame data, after being filtered and colourised for viewability.
- colour_image

  NumPy array containing colour image data.
- depth_intrin

  Object containing intrinsic depth data used to calculate pixel-to-pixel distances in real-world units.
- colour_intrin

  Object containing intrinsic colour data used to calculate pixel-to-pixel distances in real-world units.
- depth_colour_extrin

Object containing data with respect to extrinsics of colour data cast to depth data.

- bg_removed

  Image object that all objects are drawn on.

- edges

  Array object containing thresholded edge frame.

- images

  Stack of the self.bg_removed image and self.colour_image.

5.1.1 Detailed Description

Class for the Realsense Viewe.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 __init__()

```
def Viewer.Viewer.__init__ (
              self )
```

Contructor Function.

5.1.3 Member Function Documentation

5.1.3.1 adapt_depth_clipping()

```
def Viewer.Viewer.adapt_depth_clipping (
              self,
              depth_image )
```

Function to implement adaptive depth clipping.

Takes in the nearest object's depth and eliminated background behind this with a 20% tolerance.

5.1.3.2 Parameters

depth_image: The image data of the depth stream from the Realsense camera.

5.1.3.3 Returns

closest_object_tol: The distance of the closes object (+20%) tolerance in metres.

5.1.3.4 canny_edge()

```
def Viewer.Viewer.canny_edge (
              self,
              image )
```

Function to process the colour image.

Takes in colour image data and outputs image edge data, in a BW thresholded format.

5.1.3.5 Parameters

image: The colour image data to be processed.

5.1.3.6 Returns

edges: The edge data to be used in subsequent processing functions.

5.1.3.7 check_z_orientation()

```
def Viewer.Viewer.check_z_orientation (
              self,
              centre )
```

Function to check the orientation about the Z axis of the object being viewed.

Outputs a visual of the orientation of the object relative to the horizontal.

5.1.3.8 Parameters

centre: Centre point of the contour of the object.

5.1.3.9 deproj_to_pt()

```
def Viewer.Viewer.deproj_to_pt (
              self,
              point1,
              point2 )
```

Deprojects the pixel data into 3D point data with reference to the camera position.

Used to find the distances between pixels in millimetres.

5.1.3.10 Parameters

point1: The first point of the line for which the distance is required to be measured. point2: The second point of the line for which the distance is required to be measured.

5.1.3.11 Returns

[dx, dy, dz]: A list of the x, y and z distances between the two points in millimetres.

5.1.3.12 draw_roi()

```
def Viewer.Viewer.draw_roi (
              self,
              box,
              theta )
```

Draws a region of interest (ROI) above the pillar if no 'gap' is detected.

Essentially detects the top edge of the pillar.

5.1.3.13 Parameters

box: The external contour box coordinates theta: The angle of rotation of the box through the y-axis.

5.1.3.14 Returns

[t_l_, t_r_, p_l, p_r, t_l, t_r, b_l, b_r]: A list of the deployable coordinates, given that the ROI is a verified deployment area.

5.1.3.15 find_contours()

```
def Viewer.Viewer.find_contours (
              self,
              edges )
```

Function used to find the contours of the object being viewed.

Contains operations to calculate the outer contour and inner contour, and also draws the contours on the screen, and processes contour data to output multiple variables, including the deployment phase and some movement directions.

5.1.3.16 Parameters

edges: The edge data from the canny_edge() function.

5.1.3.17 Returns

box: Box object containing coordinates of the four corners of the bounding contour box. rect: List containing the centre point of the contour, the height and width of the contour, and the angle of rotation through the y-axis (into/out of the screen). contours_int: Contours object containing all internal contour data.

5.1.3.18 get_colour()

```
def Viewer.Viewer.get_colour (
              self,
              x,
              y )
```

Function to get the BGR colour values at specified pixel.

(0, 0, 0) corresponds to an inactive pixel.

5.1.3.19 Parameters

x: X-coordinate of pixel. y: Y-coordinate of pixel.

5.1.3.20 Returns

(B, G, R): Tuple containing (B, G, R) colour data (Blue, Green, Red), between 0 and 255.

5.1.3.21 get_deployment_coords()

```
def Viewer.Viewer.get_deployment_coords (
              self,
              tl,
              tr,
              bl,
              br,
              gap_rect,
              full_rect )
```

Gets the coordinates of the internal contour in the deployment phase.

Shrinks the contour to output verified coordinates so that the drone does not attempt to deploy directly against one of the dap's edges.

5.1.3.22 Parameters

tl: Original top-left coordinate of contour. tr: Original top-right coordinate of contour. bl: Original bottom-left coordinate of contour. br: Original bottom-right coordinate of contour. gap_rect: The rect data (midpoint, width, height and theta) of the gap contour. full_rect: The rect data (midpoint, width, height and theta) of the main object contour.

5.1.3.23 Returns

self.deployment_area_coordinates: List of verified coordinates for the drone to deploy the ground robot within.

5.1.3.24 get_deployment_dir_to_move()

```
def Viewer.Viewer.get_deployment_dir_to_move (
              self,
              c_x,
              c_y )
```

Gets the direction for the drone to move in the deployment phase.

Outputs a Dict object with {direction to move : coordinates of movement}

5.1.3.25 Parameters

c_x: Centre point x coordinate of the internal (gap) contour. c_y: Centre point y coordinate of the internal (gap) contour.

5.1.3.26 get_dist_to_pillar()

```
def Viewer.Viewer.get_dist_to_pillar (
              self,
              rect )
```

Get the distance between the Realsense camera and the 'bridge support pillar'.

Outputs the distance to the target in millimetres.

5.1.3.27 Parameters

rect: List containing the centre point of the contour, the height and width of the contour, and the angle of rotation through the y-axis (into/out of the screen).

5.1.3.28 Returns

dist: Distance to the viewed object in millimetres.

5.1.3.29 get_drone_control()

```
def Viewer.Viewer.get_drone_control (
              self )
```

Function to get the drone control commands, dependent on the flight phase - either flying, or deploying.

Outputs the direction in which the drone needs to move, dependent on the phase of flight.

5.1.3.30 Returns

self.pre_deploy_dir: Dictionary containing key: direction for the drone to move, value: number of pixels away from centre.

self.deployment_dir_to_move: Deployment direction if in deployment phase.

5.1.3.31 get_line_eq()

```
def Viewer.Viewer.get_line_eq (
              self,
              ln1,
              ln2 )
```

Function to calculate the equations of two lines.

In form a0x + b0y = c0 (1) & a1x + b1y = c1 (2).

5.1.3.32 Parameters

ln1: Line one start and end coordinates. ln2: Line two start and end coorinates.

5.1.3.33 Returns

[a0, b0, c0, a1, b1, c1]: List of the two lines, to be used in the above format of equations (1) & (2).

5.1.3.34 get_move_dir()

```
def Viewer.Viewer.get_move_dir (
              self,
              lr_dir )
```

Function to get the movement direction in the drone's flight phase.

Takes input directions and outputs readable control commands.

5.1.3.35 Parameters

lr_dir: Move left, right or centred command received from the get_x_dist() function.

5.1.3.36 Returns

self.pre_deploy_dir: Dictionary containing key: direction for the drone to move, value: number of pixels away from centre.

5.1.3.37 get_x_dist()

```
def Viewer.Viewer.get_x_dist (
              self,
              mid,
              wh,
              theta )
```

Function to get the direction that the drone is required to move in, only in the x-direction.

Outputs the direction and distance in pixels to move.

5.1.3.38 Parameters

mid: Midpoint of the input contour. wh: Width and height of the input contour. theta: Angle of rotation through the object y-axis of the object.

5.1.3.39 Returns

dir: Direction of required movement of the drone. dx: Number of pixels required to be moved.

5.1.3.40 intersection()

```
def Viewer.Viewer.intersection (
              self,
              ln1,
              ln2 )
```

Uses Cramer's Rule to find the intersection point of two lines.

Outputs the intersection point. Generally used in conjunction with get_line_eq().

5.1.3.41 Parameters

ln1: Line one input. ln2: Line two input.

5.1.3.42 Returns

intercept: Intersection point of line one and line two.

5.1.3.43 scale_internal_contour()

```
def Viewer.Viewer.scale_internal_contour (
              self,
              centre,
              dims )
```

Function used to verify the points of the internal contour for deployment.

Scales contour with respect to whether or not it is verified as a proper area of deployment.

5.1.3.44 Parameters

centre: Centre point of internal contour. dims: Width and height of internal contour.

5.1.3.45 Returns

tl: Scaled top-left coordinate. tr: Scaled top-right coordinate. bl: Scaled bottom-left coordinate. br: Scaled bottom-right coordinate

5.1.3.46 separate_contours()

```
def Viewer.Viewer.separate_contours (
              self,
              contours,
              hierarchy )
```

Takes in the full contour list, alongside heirarchy and separates these into internal and external contours.

Outputs independedt internal and external contour lists.

5.1.3.47 Parameters

contours: The full list of all detected contours in the colour image. hierarchy: The hierarchy of the contours: i.e. If they are internal (having parent contours) or external (having no parent contours and being the initial of the group).

5.1.3.48 Returns

int_cnt: Internal contour list. ext_cnt: External contour list.

5.1.3.49 shift_line_x_l()

```
def Viewer.Viewer.shift_line_x_l (
              self,
              point1,
              point2,
              avg_lx )
```

Shifts the left hand line of the internal gap contour in the nevative x direction.

Outputs a set of shifted coordinates.

5.1.3.50 Parameters

point1: The first point of the internal gap contour left hand line. point2: The second point of the internal gap contour left hand line.

5.1.3.51 Returns

(pt1_x, pt1_y): Tuple containing x and y coordinates of shifted point 1. (pt2_x, pt2_y): Tuple containing x and y coordinates of shifted point 2.

5.1.3.52 shift_line_x_r()

```
def Viewer.Viewer.shift_line_x_r (
              self,
              point1,
              point2,
              avg_rx )
```

Shifts the right hand line of the internal gap contour in the positive x direction.

Outputs a set of shifted coordinates.

5.1.3.53 Parameters

point1: The first point of the internal gap contour right hand line. point2: The second point of the internal gap contour right hand line.

5.1.3.54 Returns

(pt1_x, pt1_y): Tuple containing x and y coordinates of shifted point 1. (pt2_x, pt2_y): Tuple containing x and y coordinates of shifted point 2.

5.1.3.55 shift_line_y_b()

```
def Viewer.Viewer.shift_line_y_b (
              self,
              point1,
              point2,
              avg_by )
```

Shifts the bottom line of the internal gap contour in the positive y direction.

Outputs a set of shifted coordinates.

5.1.3.56 Parameters

point1: The first point of the internal gap contour bottom line. point2: The second point of the internal gap contour bottom line.

5.1.3.57 Returns

(pt1_x, pt1_y): Tuple containing x and y coordinates of shifted point 1. (pt2_x, pt2_y): Tuple containing x and y coordinates of shifted point 2.

5.1.3.58 shift_line_y_t()

```
def Viewer.Viewer.shift_line_y_t (
              self,
              point1,
              point2,
              avg_ty )
```

Shifts the top line of the internal gap contour in the nevative y direction.

Outputs a set of shifted coordinates.

5.1.3.59 Parameters

point1: The first point of the internal gap contour top line. point2: The second point of the internal gap contour top line.

5.1.3.60 Returns

(pt1_x, pt1_y): Tuple containing x and y coordinates of shifted point 1. (pt2_x, pt2_y): Tuple containing x and y coordinates of shifted point 2.

5.1.3.61 show_window()

```
def Viewer.Viewer.show_window (
              self )
```

Function used to show the 'Realsense' window and the 'Edge' window 'Realsense' - Shows processed image with all drawn contours and eliminated background, next to the colourised depth stream.

'Edge' - Shows thresholded edge detection image.

5.1.3.62 update()

```
def Viewer.Viewer.update (
              self )
```

Update function which runs all of the required functionality of the script and draws all required objects to the screen.

Called every time the screen updates.

5.1.3.63 verify_deployment_area()

```
def Viewer.Viewer.verify_deployment_area (
              self,
              shrunk_points )
```

Function to verify whether the edges of the internal contour cross any point of the surrounding support structure.

If the edges do cross the structure, the drone cannot deploy as it will collide with pillar. If they do not, then there is a verified empty space for the drone to deploy the ground robot within.

5.1.3.64 Parameters

shrunk_points: The shrunken inner contour of the gap.

5.1.4 Member Data Documentation

5.1.4.1 active_contours

`Viewer.Viewer.active_contours`

List containing all of the active contours, used to verify whether or not the contours are usable.

5.1.4.2 align

`Viewer.Viewer.align`

Alignment object to align the depth stream with colour stream.

5.1.4.3 aligned_depth_frame

`Viewer.Viewer.aligned_depth_frame`

Object containing depth frame data of the aligned depth frame (aligned with colour frame).

5.1.4.4 bg_removed

`Viewer.Viewer.bg_removed`

Image object that all objects are drawn on.

Background is removed using self.clipping_dist and aligned depth/

5.1.4.5 clipping_dist

`Viewer.Viewer.clipping_dist`

Initialise clipping distance with use of depth scale.

Used to eliminate background from image.

5.1.4.6 colour_frame

`Viewer.Viewer.colour_frame`

Object containing colour frame data.

5.1.4.7 colour_image

`Viewer.Viewer.colour_image`

NumPy array containing colour image data.

5.1.4.8 colour_intrin

`Viewer.Viewer.colour_intrin`

Object containing intrinsic colour data used to calculate pixel-to-pixel distances in real-world units.

5.1.4.9 colourised_filtered_depth

`Viewer.Viewer.colourised_filtered_depth`

Array containing the depth frame data, after being filtered and colourised for viewability.

5.1.4.10 colouriser

`Viewer.Viewer.colouriser`

Initialise colouriser for depth stream.

5.1.4.11 contour_shrink_sf

`Viewer.Viewer.contour_shrink_sf`

Float value containing scale factor to shrink deployment area by.

5.1.4.12 decimation

`Viewer.Viewer.decimation`

Initialise decimation filter object.

5.1.4.13 deploy_bl

`Viewer.Viewer.deploy_bl`

Tuple containing coordinates of bottom left corner of deployment area.

5.1.4.14 deploy_br

`Viewer.Viewer.deploy_br`

Tuple containing coordinates of bottom right corner of deployment area.

5.1.4.15 deploy_tl

`Viewer.Viewer.deploy_tl`

Tuple containing coordinates of top left corner of deployment area.

5.1.4.16 deploy_tr

`Viewer.Viewer.deploy_tr`

Tuple containing coordinates of top right corner of deployment area.

5.1.4.17 deployment_area_coordinates

`Viewer.Viewer.deployment_area_coordinates`

List containing tuples of deployment area coordinates.

5.1.4.18 deployment_area_distances_mm

`Viewer.Viewer.deployment_area_distances_mm`

List containing the width and height of deployment area in millimetres.

5.1.4.19 deployment_dir_to_move

`Viewer.Viewer.deployment_dir_to_move`

Dict containing key: movement direction, value: movement distance (millimetres).

For deployment phase.

5.1.4.20 deployment_phase

`Viewer.Viewer.deployment_phase`

Boolean - used to identify whether or not the drone is in the deployment phase.

5.1.4.21 depth_colour_extrin

`Viewer.Viewer.depth_colour_extrin`

Object containing data with respect to extrinsics of colour data cast to depth data.

5.1.4.22 depth_intrin

`Viewer.Viewer.depth_intrin`

Object containing intrinsic depth data used to calculate pixel-to-pixel distances in real-world units.

5.1.4.23 depth_scale

`Viewer.Viewer.depth_scale`

Initialise depth scale object for converting raw distances into different units (default=millimetres.)

5.1.4.24 depth_to_disparity

`Viewer.Viewer.depth_to_disparity`

Transform depth stream to disparity map.

5.1.4.25 direction_to_move

`Viewer.Viewer.direction_to_move`

String containing directions to move.

5.1.4.26 disparity_to_depth

`Viewer.Viewer.disparity_to_depth`

Transform disparity map to depth stream.

5.1.4.27 distance_from_pillar

`Viewer.Viewer.distance_from_pillar`

Float value containing distance from 'pillar'.

5.1.4.28 distance_to_move

`Viewer.Viewer.distance_to_move`

Float value containing distance to move.

5.1.4.29 edges

`Viewer.Viewer.edges`

Array object containing thresholded edge frame.

5.1.4.30 external_contour

`Viewer.Viewer.external_contour`

Boolean - identifies if the external contour is active.

5.1.4.31 hole_filling

`Viewer.Viewer.hole_filling`

Initialise hole filling filter object.

5.1.4.32 images

`Viewer.Viewer.images`

Stack of the self.bg_removed image and self.colour_image.

5.1.4.33 internal_contour

`Viewer.Viewer.internal_contour`

Boolean - identifies if the internal contour is active.

5.1.4.34 pipeline

`Viewer.Viewer.pipeline`

PyRealsense Object Initialiser.

5.1.4.35 pre_deploy_dir

`Viewer.Viewer.pre_deploy_dir`

Dict containing key: movement direction, value: movement distance (pixels).

For flight phase.

5.1.4.36 profile

`Viewer.Viewer.profile`

Initialises the profile for the depth and colour streams.

5.1.4.37 screen_height

`Viewer.Viewer.screen_height`

Global integer for screen height (default=480 pixels)

5.1.4.38 screen_width

`Viewer.Viewer.screen_width`

Global integer for screen width (default=848 pixels)

5.1.4.39 spatial

`Viewer.Viewer.spatial`

Initialise spatial filter object.

5.1.4.40 temporal

`Viewer.Viewer.temporal`

Initialise temporal filter object.

5.1.4.41 verified_deployment_area

`Viewer.Viewer.verified_deployment_area`

Boolean - used to identify whether or not the deployment area is safe and deployable.

The documentation for this class was generated from the following file:

- C:/Users/harry/OneDrive/Documents/Aerial_Source_Code/Viewer.py

Chapter 6

File Documentation

6.1 C:/Users/harry/OneDrive/Documents/Aerial_Source_Code/Main.py File Reference

Namespaces

- Main

Variables

- Main.v = Viewer.Viewer()

  Initialisation of viewer object.
- Main.e1 = cv2.getTickCount()

  Tick count 1 for calculating framerate of system.
- Main.e2 = cv2.getTickCount()

  Tick count 2 for calculating framerate of system.
- tuple Main.time1 = (e2 - e1) / cv2.getTickFrequency()

  Total time taken for function to execute.
- Main.control = v.get_drone_control()

  Control command to be sent to drone.
- Main.phase = v.deployment_phase

  Flight phase of the drone.

6.2 C:/Users/harry/OneDrive/Documents/Aerial_Source_Code/Viewer.py File Reference

Classes

- class Viewer.Viewer

  Class for the Realsense Viewe.

Namespaces

- Viewer