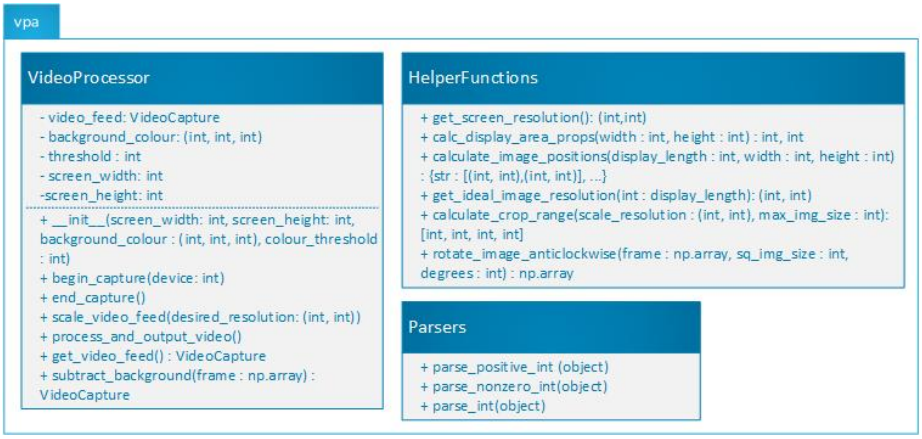
Video Processor Class Diagram

elo9@aber.ac.uk

March 2, 2017

Contents

1 Video Processor Class Diagram



2 Description of Class Diagram

The vpa (Video Processor Application) package contains three modules. The first, the **VideoProcessor** module, contains the **VideoProcessor** class which is the core of the system that will handle video capture and display. The second module will contain helper functions for the **VideoProcessor** class that do not require access to the class variables. The final module, **Parsers**, contains the object parsers for checking integer input is as expected.

2.1 VideoProcessor

2.1.1 Variables

- **video_feed**: The video_feed variable is of type VideoCapture. This class type holds a video stream and can be imported from OpenCV. To maximise the performance of the system the video object will be a global to the class object to avoid passing into and out of functions. However for test access there will be a get method for the video_feed.
- **background_colour**: A tuple that holds the (Green, Blue, Red) of the background colour.
- **threshold**: The threshold for distinguishing if a pixel is part of the background colour.
- **screen_width**: variable of type int and stores the physical width of the screen (not resolution).
- **screen_height**: variable of type int and stores the physical height of the screen (not resolution).

2.1.2 Functions

- **__init__**: Python constructor for initialising the video_feed object to None. Constructor also takes optional screen_width and screen_height for the physical screen dimensions. If these parameters are not given then the screen resolution will be used. The constructor also takes a tuple of Green Red Blue values that represent the background colour of the screen that should be removed and changed to black. In addition the threshold variable is used to state the colours to check that are outside of that range to remove.
- **begin_capture**: Function to start the video_feed with specified device. This function takes an integer (referring to the device number to use - 0 is default) and sets the video_feed to capture the video from that device. The function has a check to ensure that the parameter is an integer and raises a ValueError is it is not.
- **end_capture**: Function to release the camera feed handle and set the video_feed variable to None.
- **scale_video_feed**: This function will scale the video_feed variable to the most ideal resolution. The most ideal resolution is calculated by the get_ideal_image_resolution function.
- **process_and_output_video**: The output_video function will add 4 copies of the video_feed to a window in the correct locations. This function will also handle calls to subtract_background and rotate_image.
- **get_video_feed**: returns the video_feed object.
- **subtract_background**: The subtract_background function will take the input video_feed frame by frame and return it after removing the background (set the non-foreground pixels to black). The background will be removed by checking each pixel to see if it is within the threshold of the background_colour. If it is, it will be changed to black.

2.2 HelperFunctions

2.2.1 Functions

- **get_screen_resolution**: Function that returns the screen resolution of the main monitor which the application is running on.
- **calc_display_area_props**: This function take the specified maximum width and height of the display area and creates the largest possible square that will fit. The function then returns the length of a side of this square display area and the displacement of this display area from the corner of the screen. This function makes the assumption that the screen is wider than it is tall.

- **calculate_image_positions**: Calculates the coordinates that each image (video frame) should be placed in. The images should be in the topmiddle, bottommiddle, leftmiddle and rightmiddle of the screen. The screen length (one require one value for this as the display area is square), image width and image height are parameters of the function. The function will calculate to co-ordinates by finding using half the screen length + or - half the height or width of the image. The function will return a map of lists of tuples containing the co-ordinates. The map will be structured such that:
  {"position" : [origin tuple, bottom right tuple],
  ... }
  An example of this is:
  {"top" : [(100,0), (200,50)],
  ... }

- **get_ideal_image_resolution**: Given the length of one side of the square display area, this functions determines the maximum resolution an image should be. Images will be squares and can, at a maximum, be a third of the length of a side of the display area. This function will traverse a list of possible video resolutions and return that resolution as a tuple.

- **calculate_crop_range**: This function calculates the range to crop to image, post scale, to ensure it fits correctly in the display area. Given the resolution and maximum allowed image size, the function returns a list of the values to crop by in the form: [height_start, height_end, width_start, width_end].

- **rotate_image_anticlockwise**: Function that rotates an image (frame of the video - numpy array representation) anticlockwise. The function take the maximum image size and the rotation desired for the image in degrees. The function returns a numpy array that represents the image.

2.3 Parsers

2.3.1 Functions

- **parse_int**: Takes an object, python variable, and raises an error if it is not a integer.

- **parse_positive_int**: Takes an object, python variable, and raises an error if it is not a positive integer

- **parse_nonzero_int**: Takes an object, python variable, and raises an error if it a zero.