**Digital Image Processing**

**Group Assignment**

**CSC2014**

**Report**

**Group members:**

Gan Caqin (21035720)

Liza Ann Christian (22076095)

Ng Phey Qee (21041744)

Simran Kaur Chahal (21071899)

**Report Introduction**

This report provides a comprehensive overview of the tasks assigned in the CSC2014 Digital Image Processing group assignment, specifically focusing on **Task A – Video Processing and Task B – Paragraphs Extraction**. The objective of this report is to elucidate the methodologies, approaches, and results associated with the programming endeavors undertaken by our team, consisting of up to four members.

For **Task A**, our mission is to process and edit random videos by automating face blurring, resizing, and overlaying a friend's face in the top-left corner, ensuring seamless integration. We'll determine day/night status and apply two watermarks strategically to prevent unauthorized use. The entire process will be implemented in Python, utilizing standard libraries exclusively.

For **Task B**, our mission is to extract paragraphs perfectly from images of single-column, double-column and triple-column scientific papers while ignoring tables and figures that exist throughout the papers.

**Task A: YouTube Video Processing**

**A screenshot of a computer code

Description automatically generatedMethodology/ Proposed Approach**

***DEFINING FACE BLURRING:*** The blur\_faces function automates face blurring in a video frame. It identifies faces, extracts regions of interest (ROIs), and applies Gaussian blur (75x75 kernel, standard dev 0) to obscure facial features. This ensures privacy preservation by seamlessly blurring faces in the entire video.

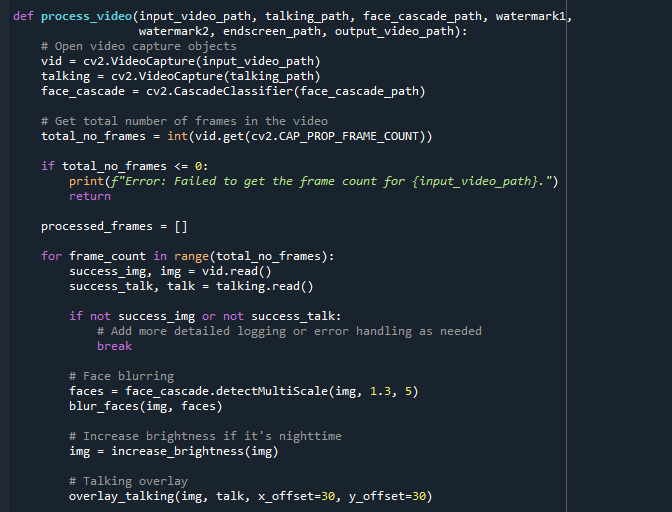
***A computer screen shot of a code

Description automatically generatedDEFINING BRIGHTNESS ADJUSTMENT:*** The increase\_brightness function adjusts a video frame's brightness based on its average, calculated by grayscale conversion. If below a threshold (100), indicating low light, it enhances brightness. It transforms to HSV, adjusts the Value channel by adding 30, ensuring dynamic adaptation or better visibility in varying lighting conditions.

*A screen shot of a computer code

Description automatically generated****DEFINING TALKING VIDEO OVERLAY:*** The **overlay\_talking** function adds a smaller video (**talking**) onto a larger frame (**frame**) in a picture-in-picture style. It resizes **talking** to a quarter of its size, adds a 5-pixel black border for separation, and seamlessly integrates it into the larger frame at specified offsets (**x\_offset** and **y\_offset**).

*A screenshot of a computer program

Description automatically generatedA screenshot of a computer code

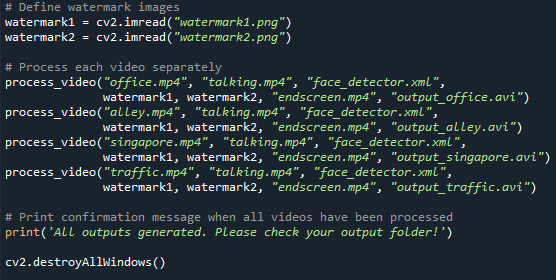
Description automatically generatedTop of Form*

***DEFINING WATERMARKING:*** The watermark function alternates between two watermarks (watermark1 and watermark2) based on the frame count. If the count modulo 150 is less than 100,   
it applies watermark1 with specified transparency; otherwise, it applies watermark2. This dynamic approach enhances video  
security and uniqueness, safeguarding against unauthorized use and contributing to visual appeal.

***MAIN VIDEO PROCESSING:*** The process\_video code pictures above, function manages the entire coding output: reads videos, checks for errors, iterates frames, applies blurring, brightness adjustment, overlay, and dynamic watermarking. Processed frames are collected, and the output video is written. An end screen video is appended, and after releasing resources, the processed video is saved with a completion message. This function combines face blurring, brightness adjustment, overlay, watermarking, and end screen integration.

***A screen shot of a computer program

Description automatically generatedENDSCREEN APPENDMENT:***In the code, **process\_video** integrates the end screen video seamlessly. It initializes a VideoWriter, iterates through frames, and appends them to the output. The end screen video is added using a separate VideoCapture, ensuring smooth integration. After appending, resources are released, guaranteeing the end screen's inclusion for enhanced visual appeal.

*****EXECUTION:*** Initialize watermarks ("watermark1.png" and "watermark2.png"). Iterate process\_video on videos, applying face detection, watermarking, and end screen integration. Save processed videos with unique filenames. Confirm processing success and close windows.

**Results and Discussion**

**A group of people walking in a hallway

Description automatically generated**Our application successfully accomplished key tasks—brightness adjustment, face blurring, overlaying a face camera, watermarking, and dynamic watermark switching every 200 frames. In the following section, we share insights gained from testing on diverse video samples.

**Observations on "office.mp4":** Top of FormFace blurring successfully applied to background faces, focusing exclusively on the main video's faces. No impact on the overlayed talking video. Smooth playback observed, even with simultaneous face blurring and watermark incorporation, showcasing seamless execution without noticeable delays.

A person walking on a street with a blurry image of a person's face

Description automatically generated**"alley.mp4" Face Recognition Analysis:** Haar cascade face recognition effectively blurs three faces simultaneously, highlighting robust multi-face recognition capabilities. However, limitations are observed when faces are turned to the side, indicating potential challenges in side-facing detection, inherent to the Haar cascade face recognition code.

A group of people standing next to a bus

Description automatically generatedA group of people waiting at a bus stop

Description automatically generated

A group of people on a street

Description automatically generated**"traffic.mp4" Brightness Adjustment:** The 2 images above are “traffic.mp4” before and after processing, showcasing successful brightness adjustment. The feature effectively increases brightness in response to a low average pixel value, enhancing visibility in low-light conditions.

A group of people walking on a street

Description automatically generated

**“singapore.mp4” analysis:** the 2 images above showcase 2 successful aspects of our code and its objectives; it was able to detect 3 faces and blur them effectively despite the lack of clarity of the video and how small the faces were. Additionally, the brightening function efficiently enhanced the video's brightness, demonstrating the adaptability and effectiveness of our code in overcoming challenging scenarios, as seen by the stark difference in the 2 images.

In summary, Task A resulted video processing application that effectively accomplished face blurring, brightness adjustment, overlay, watermarking, and end screen integration, all without hardcoding. Although successful, there's room for optimization in execution time and face profile handling. Future improvements could focus on advanced techniques for enhanced speed and recognition capabilities.

**Task B: Paragraph Extraction**Top of Form

**Methodology/Proposed Approach**

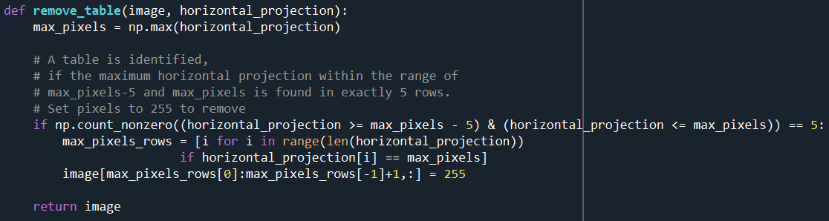
*‘Part\_B\_Main.py’* is the main code while *‘hist\_proj\_display.py’* is the code for producing histogram projection of relevant images.

***Defining histogram projection***

A screen shot of a computer code

Description automatically generatedThe code sets a threshold of 64 for image intensity values, classifying pixels with values below this as black. By summing the values generated from the comparison ***‘img < threshold’***along each row and column, the function calculates the count of black pixels in the corresponding rows and columns of the image. This process yields the horizontal and vertical projections of black pixels, providing insights into the distribution of black pixel intensities in the image.

A graph showing a number of columns

Description automatically generated with medium confidence***Defining removing table***

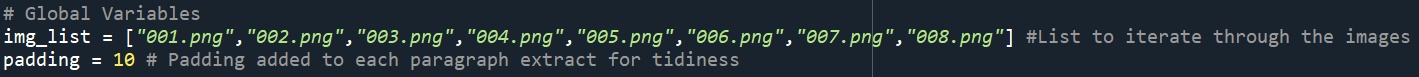
The function aims to identify and eliminate the table by analyzing the horizontal projection of the image containing the table as shown on the left. It checks if there are exactly 5 rows where the horizontal projection falls within the specified range of max\_pixels-5 to max\_pixels. If this condition is met, it signifies the presence of a table, and the corresponding rows in the image are set to the value 255, effectively removing the identified table. However, this method is usually only effective in removing tables that do not share rows with paragraphs.

A screenshot of a computer code

Description automatically generated***Defining find columns***

The vertical projection of an image is analysed to identify the boundaries of the text columns. ***‘col\_boundaries’*** is initialized to store the left and right boundaries of the columns. Initialization of flags and variables are made to control the iteration and track the gap size between histogram peaks. ***‘col\_gap\_threshold’*** represents the minimum gap size required to identify a column. The code then records the left and right boundaries in ***'col\_boundaries'***

***Global variables***



***Main loop***

A screen shot of a computer program

Description automatically generated***Removing table***

The function ***‘hist\_projection’*** is applied here to identify and remove the table. The image is then reprocessed to update its horizontal and vertical projections.

***A screen shot of a computer code

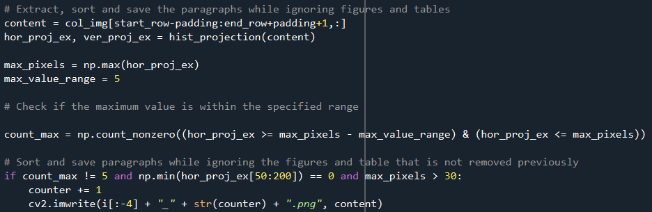
Description automatically generatedIdentifying column boundaries***

Column boundaries are found by using the *‘****find\_columns’***function based on the image's vertical projection. It iterates through each identified column, extracting and processing it.

A computer screen shot of a program

Description automatically generated***Extraction of content***

Key variables such as *‘condition’, ‘start\_row’, ‘end\_row’,* and ‘*row\_gap’*, are initialized for paragraph extraction within the column, considering factors like gap size between text lines. Extraction of the content starts when identification of upper boundary of the content is identified, meaning a nonzero value is encountered and extraction condition is not active. Extraction continues until there’s no new upper boundary identified and the row gap is larger than threshold set. The lower boundary is then set, resets extraction conditions, and proceeds to identify subsequent paragraphs in the column.

***Ignoring figures and tables when saving paragraphs as output image***

To determine whether to extract, sort, and save a paragraph, three conditions must be satisfied:

Horizontal Projection 
600 
500 
— 400 
300 
200 
100 
100 
Pixel Position 
150 
200 ***Condition 1:*** *‘count\_max != 5’*

The content (table) is ignored if the range within the specified range of ‘max\_pixels - max\_value\_range *(max\_value\_range=5)*’ to max\_pixels is detected 5 times across the rows. In other words, the content will only be extracted as paragraph if it does not have 5 rows that are within the maximum value range of 5. The horizontal projection of the table is shown on the right.

***Condition 2:*** *‘np.min(hor\_proj\_ex[50:200])==0’*

A graph showing a graph of a graph

Description automatically generated with medium confidenceThe figure is ignored if there are no projection values equal to 0 within the range of 50 to 200 in the horizontal projection of extracted content (hor\_proj\_ex). This condition is in place to filter out figures where the projection values are continuously non-zero as shown in the histogram graph below.

A tree with flowers in the foreground

Description automatically generated

***A graph of a graph

Description automatically generatedCondition 3****: ‘**np.max(hor\_proj\_ex)>30’*

A white background with black and white clouds

Description automatically generated with medium confidenceThe figure is also disregarded if the maximum projection value in the horizontal projection of extracted content (hor\_proj\_ex) is less than or equal to 30.

**Results & Discussion**

|  |  |
| --- | --- |
| **Input** | **Output** (Extracted Paragraphs) |
|  |  |
|  |
|  |
|  |

As observed in the example output the paragraphs are successfully extracted without extracting the figure. Four images were extracted from image 005 and saved as 005\_1, 005\_2, 005\_3, and 005\_4 respectively. The “Sakura” image existing between paragraphs 1 and 2 has a maximum horizontal projection value is less than 30, not fulfilling one of the conditions to be saved as an image output. Hence, it is not saved as an output image. All the paragraphs were accurately extracted, and none of these paragraphs exhibited any missing text lines or characters.

Top of Form