UNIVERSITY OF LONDON

BSc Computer Science

Module: CM2030 – Graphics Programming

Introduction

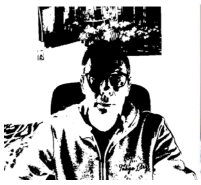
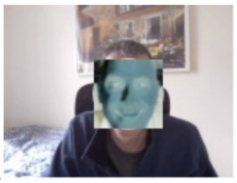
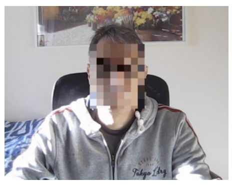
For this assignment you will develop an image processing application. You will utilise the webcam to perform several image processing manipulations such as image thresholding and colour space conversions. You will also use face detection to perform some privacy enhancing filters such as face blur and pixelate. You will earn individual points for each requirement and points for the overall performance i.e. is the logic of the application simple or complex and confusing, are there any bugs during execution etc. You will need to justify your answer for a few of these tasks – use a commentary to discuss that. You can only use the libs shown in this course and no external code will be allowed. All the code should be your own work. The following grid of images should be expected to appear at your browser when executing the app. See steps below for completing the final assignment.

Webcam image	Grayscale and brightness + 20%	
Red channel	Green channel	Blue channel
Threshold image	Threshold image	Threshold image
Webcam image (repeat)	Colour space 1	Colour space 2
Face detection and replaced face images	Threshold image from colour space 1	Threshold image from colour space 2

Task/steps

1. Load an image using the webcam: you can either take a snapshot, save it to disk and load it back or you can use this snapshot as a buffer image and use it without saving it to disk. Use key/mouse interaction to take the snapshot or save the image.
2. Scale that image to 160 x 120 pixels (i.e minimum resolution).
3. Display the webcam image in the grid at the position titled “Webcam image”. Then using this webcam image complete all tasks 4 – 11.
4. Convert image to grayscale and increase the brightness by 20%. This should happen within the same nested for loop that you use to convert the image to greyscale. Display the image at the appropriate position as seen in the grid.
5. Increasing the brightness can cause the pixel intensity to get beyond the 255 levels. Write code to prevent this from happening.
6. Using the original webcam image: split into three colour channels (R, G, B) and show each channel. Again, use the appropriate positions as seen in the grid.
7. Perform image thresholding with a slider for each channel separately. Similarly, display the results using the appropriate positions as seen in the grid.
8. What can you say about the result of thresholding for each channel – is it different and why? Explain this in the commentary section.
9. Using the original webcam image again and assuming your camera’s colour model is using an RGB colour model, perform colour space conversion. Select two algorithms from the resource attached in the submission page. See examples of how the converted images should look like at the bottom of this file. Display the images at the correct position in the grid again.
10. Using the colour converted images perform image thresholding using either a static threshold or a slider to control the threshold. If using a static threshold value, make sure the thresholded image shows an expected result (see 1st image at the bottom of this page). There’s no need to split into channels here. Display the images at the correct position in the grid again.
11. What can you say about the thresholding results above when compared to step 7 i.e. is the thresholded image noisier etc? Can you use a different colour space to improve results? Discuss the above in the commentary.
12. Perform face detection as described in the lecture videos. Here you have two options: you could use the approach shown in the course (see Week 19) or the FaceMesh api from here https://docs.ml5js.org/#/reference/facemesh to provide the bounding box.
13. Display the detected face at the correct position in the grid - then by using keystrokes (i.e. 1, 2, 3, 4 etc.) - replace the detected face image (see 2nd and 3rd images below) with:
    a. A greyscale image
    b. A blurred image – adjust blurring so your face is not recognisable
    c. A colour converted image – reuse code from task 9
    d. A pixelate image. To perform pixelation use the following approach
        i. Run step a – so that your image is greyscale
        ii. *Split* the detected face image into 5x5 pixel blocks
        iii. Calculate the average pixel intensity of each block using **image.get(x, y); or use the pixel array** (to access each pixel’s intensity)
        iv. Paint the entire block using the average pixel intensity. Utilise this command: **outimage.set(x, y, avePixInt); or use the pixel array**
        v. Loop through all blocks repeating steps iii and iv.

Display the images at the correct position in the grid again.

Coding style

1. Code presentation: use appropriate syntax, comments, consistent indentation and leftover code blocks
2. Code competency: use of object orientation, code reusability, use of functions, variables global vs local

Extension

Since this is a creative module, we would like to give more marks for implementing further ideas. Also, you should write some words about it in the commentary (see next paragraph). Please note that we will award marks for the uniqueness of your extension and how technically challenging it appears to have been. The extension is worth 20% of your mark.

Commentary

- Discuss your findings e.g. image thresholding using each colour channel
- What problems have you faced and were you able to solve them?
- Were you on target to successfully complete your project? If not, how would you address the issue/s and do things differently?
- Also discuss your extension and why it is a unique idea

The report quality (i.e. language) will also be assessed. Be precise and deliver the information within 500 words. Include this in your main .js file.

Video demo

Demonstrate your application using a video. While you run your app - verbally go through all functionalities (i.e. go through tasks 3 - 13) and demonstrate how they work. Also talk about your extension. Always have the console window open so any behaviour is recorded. Use OBS Studio or similar software and not your mobile phone and make sure you talk through your app during the demo. You can upload the video in **.mp4** format or use the alternative link – see submission page. We will give zero points to this question if console is not shown. The video should be up to 5 minutes long. We will only watch the first 5 minutes of the video.

General information

You should complete this work using the libs shown in the module. Do not use external code for this assignment. All your code and commentary will be checked for plagiarism/AI generation.

Submission requirements

1. Compress all your code in .ZIP format and upload it in the first prompt
2. Upload the video demo in .mp4 format in the second prompt
3. Use alternative video submission link to upload the video demo to YouTube or similar, then submit the URL only in the third prompt. **Make sure your video remains unlisted.**
4. Merge all your .js code in a single file, then upload it in the last prompt. Use the JavaScript Bundler Tool (see previous learning item) to merge all your .js files into a single .txt file. Exclude any libraries you used such as p5.js, matter.js etc. **This is a submission requirement as we need this to run your code. You will receive zero points for the entire project if you fail to submit your code in this way.**

Rubric

[3 points]: Program runs and without errors? (check console)

[3 points]: Program is usable e.g. easy to use without confusing behaviour

[1 point]: Image loaded from the webcam

[1 point]: greyscale conversion

[1 point]: brightness increase to 20%

[1 point]: added brightness not to exceed 255 pixel intensity

[1 point]: split image into three channels

[3 points]: perform image thresholding for each channel with a use of a slider to adjust values

[2 points]: colour space conversion #1

[2 points]: colour space conversion #2

[2 points]: perform image thresholding for colour space #1 [1 point if you use static threshold] or [2 points with a use of a slider to adjust values]

[2 points]: perform image thresholding for colour space #2 [1 point if you use static threshold] or [2 points with a use of a slider to adjust values]

[2 points]: Face detection works using a bounding box

[1 point]: Replace the detected face image with a greyscale converted image

[1 point]: Replace the detected face image with a blurred image

[4 points]: Replace the detected face image with a colour converted image (check if task 9 is revisited)

[6 points]: Pixelate filter (check for nested loops, correct use of average)

[4 points]: Code presentation: indentation, white space, comments, variable naming

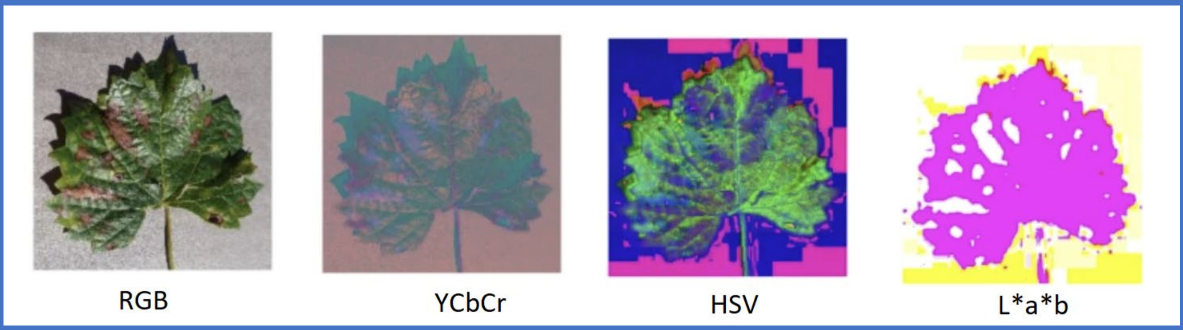
[4 points]: Code competency: code reusability (some functionality is repeated – did student use functions/OO to organise the code better?)

[8 points]: Commentary included?

[4 points]: Video included?

[14 points]: Has learner implemented any unique ideas for further development? How technically complex, original and challenging was the idea and its implementation?

Example of colour space conversion



Khan, M.A., AlGhamdi, M.A. An intelligent and fast system for detection of grape diseases in RGB, grayscale, YCbCr, HSV and L*a*b* color spaces. Multimed Tools Appl (2023). https://doi.org/10.1007/s11042-023-17446-8