Introduction to Computer Vision

using Python

Assignment 1: Image Processing, Color, Edges, Lines

Deadline: 31/12/21, 12 pm

1. OpenCV - Basic Operations

Image Resizing: We've been talking a lot about resizing and interpolation in class, now's your time to do it! To resize we'll need to implement an interpolation method and fill the new image with the pixel values using the input image (*example.jpg*) and return it as output. Try to both upscale and downscale the image. What can you say about the results?

Create a function **nn_resize(image, w, h)**. It should:

- Create a new image that is **w** x **h** and the same number of channels as the image.
- Loop over the pixels and map back to the old coordinates
- Use **nearest-neighbour** interpolation to fill in the image. Remember to use the closest integer coordinates.
- Display the result(s) on screen or save the image to disk.
- You can use OpenCV methods

2. OpenCV – Filtering and Lines

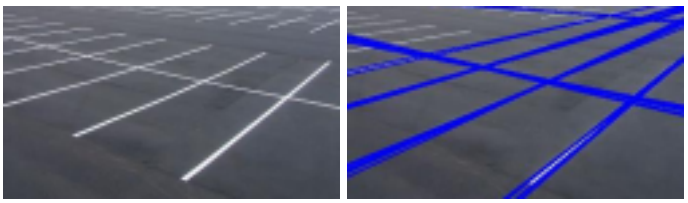
In this assignment you will also get the chance to play around with OpenCV and learn some of its capabilities. It provides many ready-to-use functions so that you don't have to write low level code. Also note that OpenCV reads and displays images assuming the BGR ordering instead of RGB. You can convert back and forth if needed. In some cases, you will need to post-process the output of a built-in function so read the documentation to familiarize with the outputs of each function and expected input format.

a) **Skin Detector:** In this exercise you will process color images of faces (*f1.jpg* and *f2.jpg*) to detect the skin regions in the image. Use the **cv2.cvtColor()** function along with the **cv2.inRange()** function to write the skin detection algorithm. The output of your code should be a binary image where the pixel values corresponding to skin should be *255* and *0* otherwise. You can choose whether to convert the RGB image to a different color space or not and state your reasoning in either case. Then find the best threshold discriminating between skin and non-skin pixels. Try it with images of your own. How good are the results? In your submissions include the images that you have tested and comment on the results.

b) Parking Lines detection: In this exercise you will process some color images (*p1.jpg*, *p2.jpg*, and *p3.jpg*) of parking lots to detect the parking lines. Try to use a combination of **cv2.sobel(),**

cv2.canny() and colour thresholding with **cv2.threshold(), cv2.inRange()** to extract relevant edges and regions. Use the **cv2.houghlines()**, to find the possible lines in the image. The output of your code should be a color image where the detected lines are visible as below.

- You will need to convert the output of the *houghlines* function from (ρ, θ) to points (x_1,y_1) and (x_2,y_2) that you can then use to draw the lines using **cv2.line()**.



Helper Functions:

- Use **cv2.gaussianblur()** to smooth an image.
- Use **cv2.imshow()** to display an image. You will need to run **cv2.waitKey()** to update the UI to actually display the results.
- Use **cv2. resize()** to resize an image
- Use **cv2.imread()** to read an image
- Use **cv2.imwrite()** to write an image to disk

How to submit your assignments?

After you have completed each assignment, you need to submit the following deliverables:

- The code!
- All additional data used.
- Assignment_1.pdf – A write up that must Include all answers to questions and expected results with the relevant discussion if needed.
- Alternatively, if you are using a jupyter notebook (e.g., google colab) export it as a PDF while including all the above in the notebook (results, etc.).
- Zip all files and email to thuraaung.ai.mdy@gmail.com OR send it to Discord Channel