Submission Deadline: @12:59 30 November 2022

Answer sheet for 1/2565 Final Exam for
2110431 Introduction to Digital Imaging
2147329 Digital Image Processing and Vision Systems

Read the instructions carefully

Rules of Conduct for Students during Examinations

Unacceptable examination conduct includes

Share your answers or code with your friends or looking at exam materials of other students or anyone or allowing others to look at their exam materials.

When a student cheats or is in suspicion, the supervisory officer has the power to investigate the matter. There are serious consequences to academic misconduct, including receiving an “F” on the work or examination, an “F” in the course.

Name_Naphat_Darunaitorn_____ID__6230136021_____

You must submit

(1) Answer sheet (in pdf)

(2) .ipynb file for problem 1

In MyCourseVille before 12:59, November 30, 2022. Late submission is not accepted.

THE EXAM IS **DUE** ON 12:59, NOVEMBER 30, 2022

Problem 1. Liquid Content and Particles (40 points)

Answer the following questions in the **answer sheet** provided and **.ipynb** in mycourseville only

1.1) Describe steps of your implementation and key parameters for

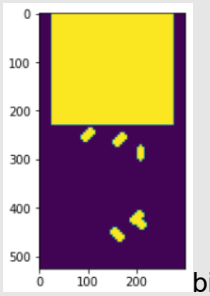
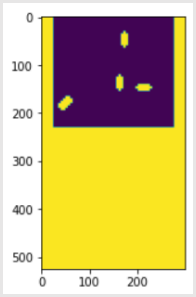
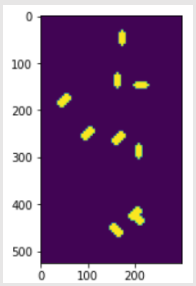
- Calculating proportions of each content in percentage
- Counting the total number of particles
- Counting the number particles in each content

(20 points)

Steps	Details and reasons

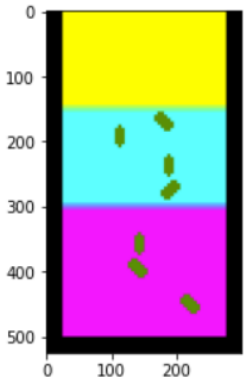
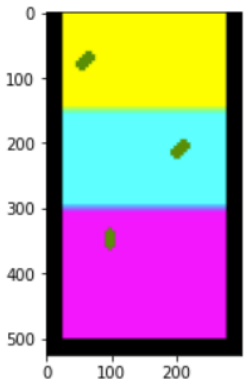
1	For calculating proportions of each content in percentage I start with sampling image vertically at y=width//4 y=width//2 y=3width//4
	`height,width,dim=image.shape #collect height and width from image`
	`first_sampling=image[:,width//4,:]`
	`second_sampling=image[:,width//2,:]`
	`third_sampling=image[:,3*width//4,:]`

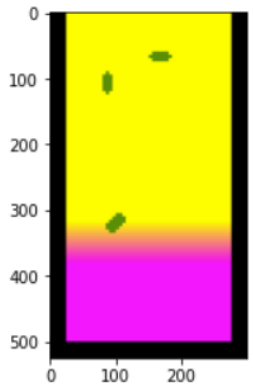
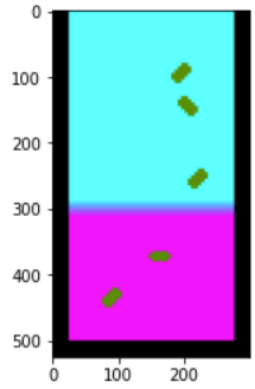
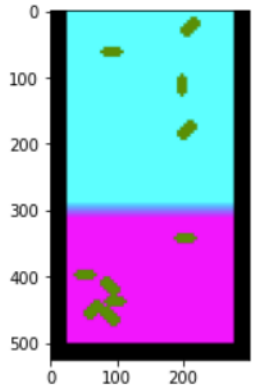
2	then use call function find_region to 3 sampling this function call function thresholding_color for each pixel this function will return nearest color in the set (decide by nearest euclidean distance) so our set included 3 color of contentA,B,C ,color of particle and beaker color then find_region will find begin and end region of A,B,C content
3	Next we will get begin and end region of A,B,C for 3 sampling then we calculate these value to get actual value of these value in case of there are some error in some sample then we will get these actual value And count pixel of each content.
4	Before final step I decide to error correction again. I this case I want to correct gathered value in case that particle lying between boundary of our three samples
5	Finally I calculate each proportion to percent from each content count divided by sum of all content count. This is the end of calculating proportions of each content in percentage part.
6	For Counting the total number of particles I convert image to grayscale then apply thresholding and thresholding_inverse each of them threshold by each

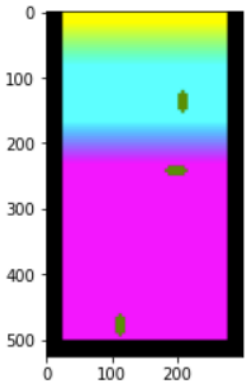
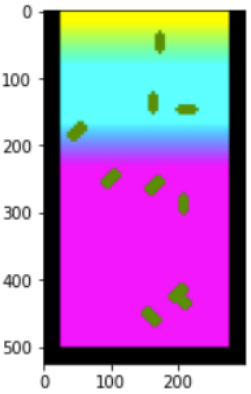
	value for lighter part and darker part because we know luminosity of particle in grayscale (by Y = 0.299 R + 0.587 G + 0.114 B) but our image could contain lighter and darker part so I separate it to apply thresholding and then apply it together again by bitwise_xor and inverse its value  bitwise_xor with  then inverse= 
7	Then I count it by connected component technique by cv2.connectedComponentsWithStats to count it but in particle can be overlapped so I count number of particle from area of each connected component floor function with 425 (from observation minimum area of particle is 425 and each particle will be same size in criteria but this area value is its frame so its can be larger than 425 so I use floor function). From this process we can count total number of particle this is ending of Counting the total number of particles part.
8	For Counting the number particles in each content I decide to get all stat from Counting the total number of particles part (using cv2.connectedComponentsWithStats) then split result to each content using centroid of each particle from cv2.connectedComponentsWithStats and get value of boundary of each region by calculate begin and ending region like calculating proportions of each content in percentage part then apply find_boundary function that will calculate coordinate each content boundary and count it and assign to each boundary using same technique from Counting the total number of particles part. This is ending of Counting the number particles in each content part.

1.2) Write your code in ipynb file which will be mainly tested (10 points) and also provide the results captured for the images below (in case your code is not running, will check some result here)

**** you may have slightly different results from the result examples you may analyze your selected techniques in 1.3)**

Image	Results (Capture your results of your implemented functions here – proportion, number of particles in each content and total particles)
01_01.png	 Proportion: 29.95991983967936%, 30.060120240480963%, 39.97995991983968% Number of Particles in each content: 0, 4, 3 Total number of particles: 7
01_02.png	 Proportion: 29.95991983967936%, 30.060120240480963%, 39.97995991983968% Number of Particles in each content: 1, 1, 1 Total number of particles: 3

02_01.png	 Proportion: 70.04008016032064%, 0.0%, 29.95991983967936% Number of Particles in each content: 3, 0, 0 Total number of particles: 3
03_01.png	 Proportion: 0.0%, 60.02004008016032%, 39.97995991983968% Number of Particles in each content: 0, 3, 2 Total number of particles: 5
03_02.png	 Proportion: 0.0%, 60.02004008016032%, 39.97995991983968% Number of Particles in each content: 0, 4, 6 Total number of particles: 10

04_01.png	 Proportion: 9.919839679358718%, 30.060120240480963%, 60.02004008016032% Number of Particles in each content: 0, 1, 2 Total number of particles: 3
04_02.png	 Proportion: 9.919839679358718%, 30.060120240480963%, 60.02004008016032% Number of Particles in each content: 1, 3, 6 Total number of particles: 10

1.3) Do you get the desired results? Why you get the desired results or why not? please analyze (4 points)

Yes, in proportion part I think my method is slightly low error because I have 2 error correction function (3 sampling and fix boundary gap). In total number of particle part this may contain error if there are a lot of overlap particle in image. In number of particles in each content this can contain error from earlier part (total number of particle part) because I use the same method, and this may contain error in assign counting to each part if there are overlap particle lying between 2 or more content my method will assign all of them to content that area contain group of overlap particles centroid. But in test set condition my code work well result is pretty accurate to truth answer because of those images aren't contain earlier mentioned condition.

1.4) How do you evaluate your results? Give some examples of the evaluation. (6 points)

As I mentioned earlier, the counting and assigning part doesn't work well in mentioned conditions. It's because of limitation of connected component technique. So, if I want to improve my code, I may use sliding window algorithm with fixed size of window to detect particle instead of connected component algorithm. I think the Yolo algorithm and anchor box method would work well due to different alignment of particles. But if ignore those conditions this model's accuracy is high. So, I assume that case of huge amount overlapping particle and alignment that much enough to make my estimation error is less than 5% (case1 error) and in case of pack of particle lying between content boundary and there are some of them lying in different content (case2 error) is less than 5% of all data. So prob of overall error (case1 union case2 error) = prob of case1 error + prob of case 2 error – prob of case 1 intersection with case 2 error. In worst case (intersection part=0) this model would have overall error less than 10% (model accuracy is more than 90%). Note that this calculation varies according to the proportion of data that contain case1 and case2 error. This model will perform perfectly if there is no data that contains those 2 cases.

1.5) References for your code / ideas

My first idea for proportioning part is I want to deal with less data much as possible due to this code will work with image. Calculating all of pixels in the image won't be effective so I think about sampling some part of the image. From aligning if content in the image, content align horizontally at equal height value, so I sample vertically for effective and I deal with variable color at boundary with Euclidean distance from hint. I choose to assign each pixel to the nearest color in the provided set (shortest Euclidean distance). According to the sampling method this may be less accurate than computing all the images and contain some errors. So, I chose to pick 3 sampling and correction boundary code to prevent error from particle is lying between content boundaries.

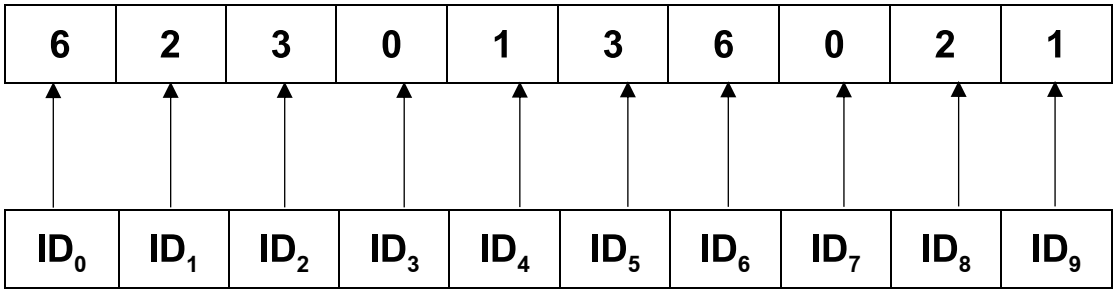
For the counting part I was quite impressed by connected boundary technique and cv2.connectedComponentsWithStats library from classroom because of this library provided coordinate of centroid and area of connected component so I choose to use this method. Then I was faced with overlapping particle problem. So, I decided to calculate the number of particles in overlap component by floor function each area of connected component by 425 this number is collected from smallest size of particle.

For counting and assigning to each content I use same method with proportioning part but provide coordinate of each boundary content so I can assign each particle to count in each content from centroid coordinate provide from library.

Problem 2 Object Recognition

Preliminaries

Fill in your ID here



2.1) Histogram of Oriented Gradients (5 points)

Calculate HoG feature descriptor by setting

- Orientations $= \begin{cases} 9 & ID_5 = 0 \\ ID_5 & otherwise \end{cases}$
- Pixels per cell $= \begin{cases} 16 \times 16 & ID_6 = 0 \\ ID_6 \times ID_6 & otherwise \end{cases}$
- Cells per block $= \begin{cases} 2 \times 2 & ID_7 < 5 \\ 3 \times 3 & otherwise \end{cases}$

2.1.1) Fill in your setting here

$ID_5 =$3	Orientations = 3
$ID_6 =$6	Pixels per cell = 6*6
$ID_7 =$0	Cells per block =2*2

2.1.2) Fill in your setting in python code below

```
fd, hog_image = hog(image, orientations=3, pixels_per_cell=(6, 6),
cells_per_block=(2, 2), visualize=True)
```

2.1.3) What is the size of HoG feature descriptor? How you get this size?

```
Size = 84672

Calculate by
cell per image=512//6*512//6= 85 * 85
from sliding 2*2 window through cell to construct 2*2 cell per block
across the image
block per image=(85-1)*(85-1)= 84 * 84
one cell contain 3 content there are 4 cell in one block
So, content per image = 84*84*4*3 = 84672
```

2.2) Convolutional Neural Networks (6 points)

In lecture 10, object recognition, you have configured the convolutional neural networks for MNIST dataset using pytorch. You have tried to improve the model with suggested techniques. Please suggest 3 primary techniques that you will use for improving the model in the future and state the reason for the selected techniques.

1. Add dropout this technique is effective with overfitting problem because this technique will randomly selected neurons are ignored during training.
2. Batch normalization this method will re-centering and re-scaling the input. So, the model can work faster and more stable. Example for classification model due to batch normalization your model easier to decide boundary to classified object so your model may work faster and more stable.
3. Change number of epochs. Due to underfitting and overfitting problems, changing number of epochs trend affect this problem. If your model is overfitting reduce epochs number may improve your model but underfitting you may want to decide to increase epochs or collect more data if data is too small increasing number of epochs may won't solve underfitting effectively.

Problem 3 General questions (explain in your own words)

3.1) You have learned about Hough line transform and Hough circle. Explain the ideas of finding an ellipse in an image based on Hough transform. How can we find an ellipse in an image? (4 points)

Same method with Hough line and circle this algorithm will try to fit ellipse to image by try to draw ellipse to fit pixel of an image. This method is used after edge detection like line and circle. Drawing method is quite common line and circle transform, in this case we draw ellipse varied by center coordinate, major axes, minor axes and orientation to find result ellipse.

3.2) Otsu's thresholding originally aims for segmenting an image into two groups (binary image). However, we might need to segment an image pixel into more than two groups. You have learned about how to compute optimal threshold using Otsu's thresholding based on the intensity distribution. Can you suggest the idea how you can extend Otsu's thresholding for multiple intensity thresholding? (4 points)

In binary thresholding method we assign value 0 or 1 base on intensity level of each pixel in the image base on 1 value that call thresholding value. It's like we split image in 2 region by 1 intensity level. So, if we want to do multiple thresholding we need to construct more than 2 region and split it to 3 or more intensity value that we assign.
For example if we have [0 , 1 , 2 , 8 , 12 ,15]
If we do binary thresholding with intensity value = 3 we will get [0 , 0 , 0 , 1 , 1 , 1]
But if we do multiple thresholding with value = 3 , 13 we can get [0, 0 , 0 , 1 , 1 , 2]
(value 0,1 or 2 can be adjusted)

3.3) You start working in a company which just recently have an autonomous driving project. The company want to start using computer vision for detecting vehicles using front camera on

the vehicle. The company as for your opinion for this task. What would you suggest them particularly about the data and model? (4 points)

I would suggest this problem can be divided into 2 parts. The first part is vehicle recognition that we have to train our model that can detect which image vehicle or not. The second part is object detection where we must select some part of the image to pass through our first trained model to decide where the vehicles are lying. vehicle recognition part can be done by preprocessing and training machine learning model. I think hog feature method would be effective in this part for more effective. Applying hog feature in each color channel may lead to better result. In object detection part I think sliding window method will be effective. For more accuracy the anchor box method can be applied. This method would work well if there are some objects overlapped in the applied image.

3.4) Give your option that which of the morphology techniques you found the most useful and why? Provide several examples of how and when (you) use it? (4 points)

I think connected component is the most useful method because this method can be used to detect and count the connected component in the image. This method will be more effective when image is preprocessed ex. Noise reduction, thresholding method.
When I use it: when I want to count or detect connected component that not overlapped for example counting bottle in cage from top view by apply thresholding to extract bottle cap and counting number of bottles by connected component method and counting goods in production conveyor by apply thresholding to separate goods from background then apply connected component for counting goods.