CSE 398/498 - Assignment 2

Simple Object Detection & Classifiction

Logistics

The submission should be made on Course site in the form of a zip file named as **<your_full_name>_cv_21.zip** by **September 24, 2021 5:00pm (EST).**

Contents of the compressed folder should be organized in the following manner:

- **data/** - folder containing input images
- **output/** - folder containing images of preprocessing or other intermediary stages
- **src/** - containing your python scripts, one file for each task, python scripts for tasks 1 and 2 are provided.
- **readme.md/readme.txt** - containing verbose description of the steps required to run your code and other observations/comments you may have for your submission. Treat this as a short report. Include any links you may have referred to in this document.
- **environment.yml** - exported conda environment file. Contains information about libraries and their versions.

Credit: 5% of course total

Due date: September 24, 2021, 5:00 pm

Pre-assignment

Prepare your laptop. Virtual environments are a great way to ensure stability of your code execution and also improve its reproducibility. I suggest everyone to use the **Anaconda** package manager for your projects and assignments. If you have used it before, you already know. If you haven't used it before, it will make installations easier and project management better. Also, a lot of other people's code uses it so it's good to learn.

Quick tutorial for conda: https://conda.io/projects/conda/en/latest/user-guide/getting-started.html

This practical will again use Python and the packages you used in the previous assignment. In addition, we need to install a package called scikit-image. You can use the following command to install this package easily:

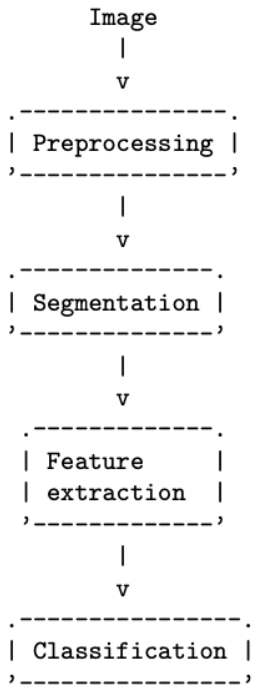
• If you are using Anaconda: `conda install -c anaconda scikit-image`

• Or you can choose to use pip: `pip install scikit-image`

You can find the tutorial and documents for this package from their website:
https://scikit-image.org

Simple Object Detection & Classification

Our goal in this assignment is to classify different simple objects in the image provided in class. The procedure for object classification in this example will follow the general pipeline in Computer Vision:



Preprocessing

The first step involves thresholding and converting the image to a binary image. The result will be a binary image with each object of interest expressed as a connected component of white pixels. Ideally, each connected component is an object to classify. However, in practice, objects may be only partially binarized, or they may overlap. Intensity of images that you will process in class may have some linear trend (read up on detrending), thus its removal should help in getting a better binary image. You may also see the need for histogram equalization before binarization.

Segmentation

Assuming that we already have a good binary image, your next task will be to find all the white blobs representing objects of interest. `scikit-image` package provides a very nice command to find all the connected components:

```
from skimage import measure

labels = measure.label(bw_img, n)
```

in which:

- `bw_img`: binary image
- `n`: connectivity

`n` can take values of 4 or 8, indicating either 4-connected or 8-connected.

Feature extraction

In this assignment we will use simple geometrical features to classify several classes of objects. For instance, looking for elliptic shapes we can model each localized object as an ellipse. The ratio of major axis to minor axis of the localized objects should be greater than 1 for ellipses, while for, say, round objects the ratio should be roughly equal to 1.

This example illustrates that even trivial features such as major-to-minor axes can help to classify simple objects. In this part, you will use a very useful scikit-image function regionprops to extract basic features from the connected components:

```
features = measure.regionprops(labels)
```

where:

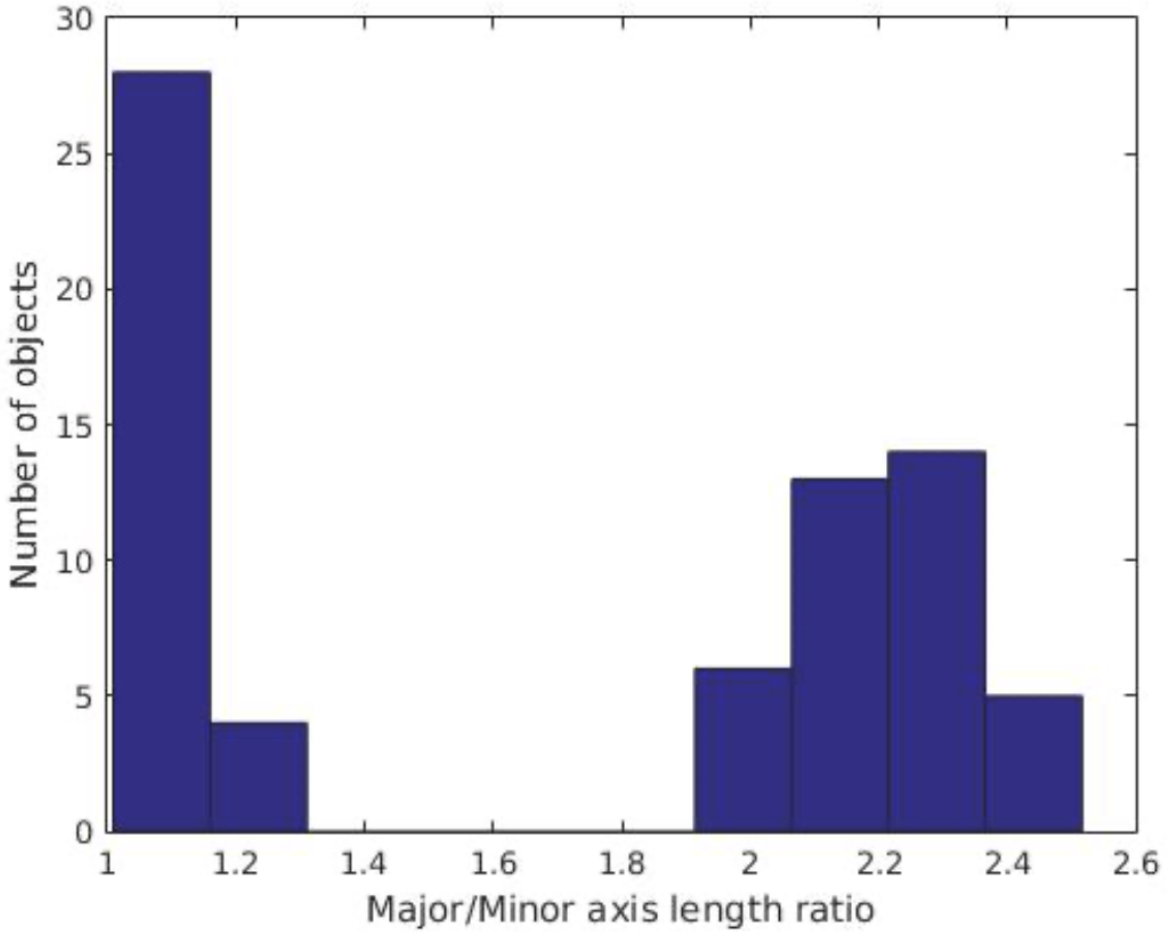
- `labels` = connected components,
- `features` = comma-separated list of properties we want to extract, for example: 'area', 'bbox', 'centroid', . . .

  For a complete list of properties check https://scikit-image.org/docs/dev/api/skimage.measure.html?highlight=regionprops#skimage.measure.regionprops

Classification

For illustration, assume we want to distinguish oval pills from the round ones in this picture:

If we use `regionprops` to get major and minor axis length for each connected component, and plot the histogram of their ratio, we see that this single feature is sufficient to classify these shapes perfectly:

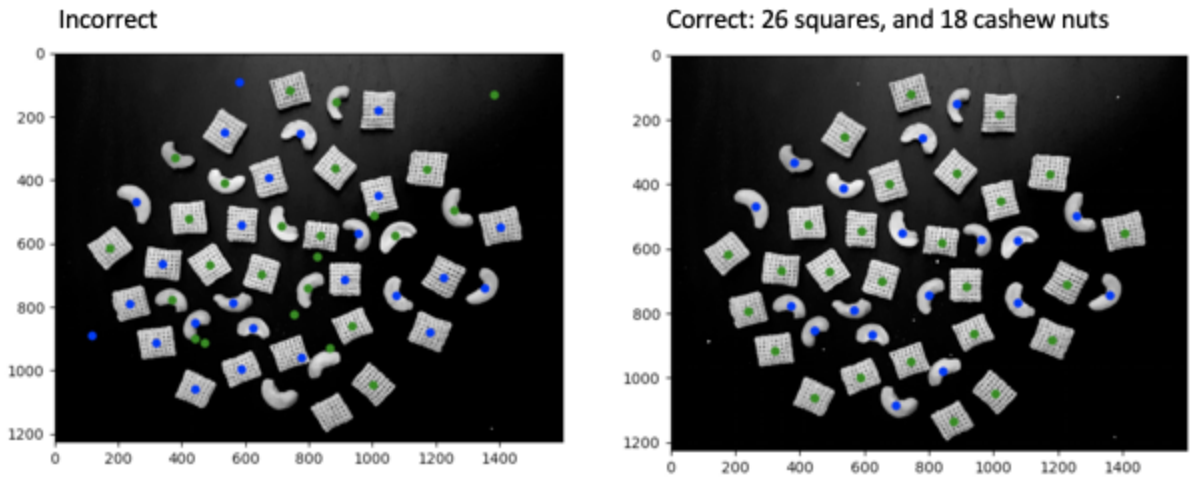


Certainly, one feature might not be enough to separate the classes. Using more features might lift up the data points to a higher-dimension feature space in which they are separated (see this nice illustration).

Now, when we have the features determined, we can build a simple classifier to classify the objects: just check if the ratio is above, say, 1.5. If it is, the object is *elliptical*. If it is not, the object is *round*.

Task 1 (1 point):

Run "task1.py", analyze the code and modify it
(by adding morphological operations) to get the correct object detection. Uses **breakfast1** image. Part of the code annotated as *** needs your attention. Finish the missing parts and present the correct output as shown below to obtain 1 point for this task.



These lines may help you
(experiment with different combinations of these operations and/or kernel size):

```
# definition of a kernel (a.k.a. structuring element):
kernel = np.ones((10, 10),np.uint8)
# one iteration of morphological erosion:
sample_res = cv2.erode(binary_image, kernel, iterations = 1)
# one iteration of morphological dilation:
sample_res = cv2.dilate(binary_image, kernel, iterations = 1)
# morphological closing:
sample_res = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)
# morphological opening:
sample_res = cv2.morphologyEx(binary_image, cv2.MORPH_OPEN, kernel)
```

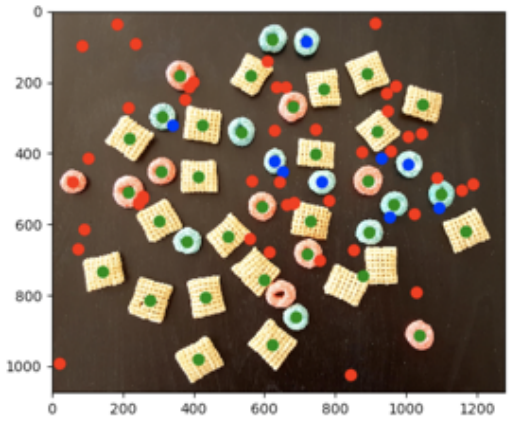
Task 2 (2 points):

Run "task2.py", analyze the code and modify it (e.g., by adding the second feature based on color/intensity) to get the correct object detection. Parts of the code annotated as *** need your attention. Finish the missing parts and present the correct output as shown below to obtain 2 points for this task.

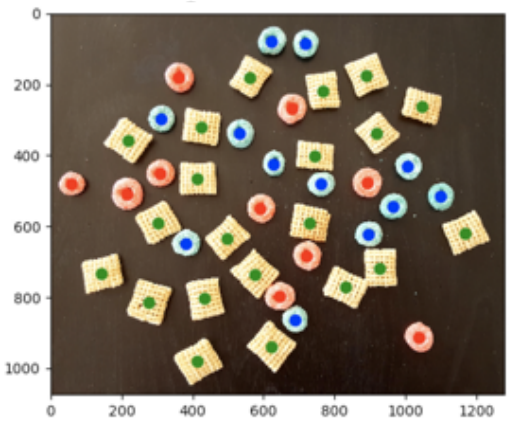
These lines may help you (filling the holes):

```
im_floodfill = binary_image.copy()
h, w = binary_image.shape[ : 2]
mask = np.zeros((h + 2, w + 2), np.uint8)
cv2.floodFill(im_floodfill, mask, (0, 0), 255)
im_floodfill_inv = cv2.bitwise_not(im_floodfill)
binary_image = binary_image | im_floodfill_inv
```

Incorrect

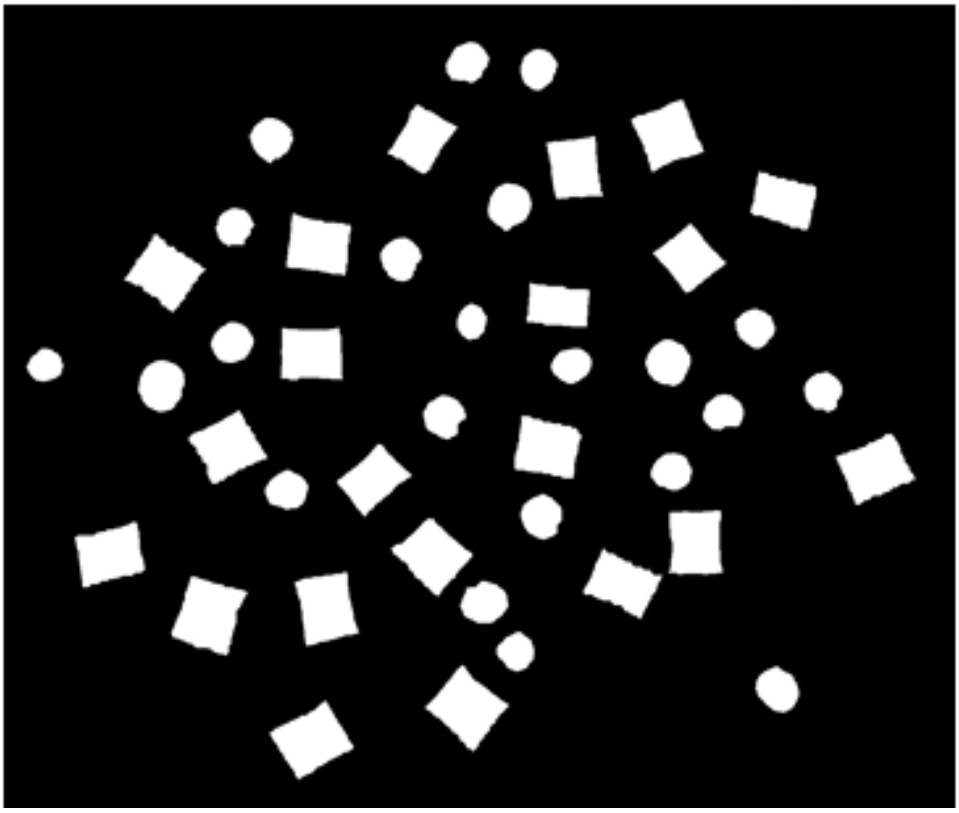


Correct: 21 squares, 12 blue donuts, and 10 red donuts

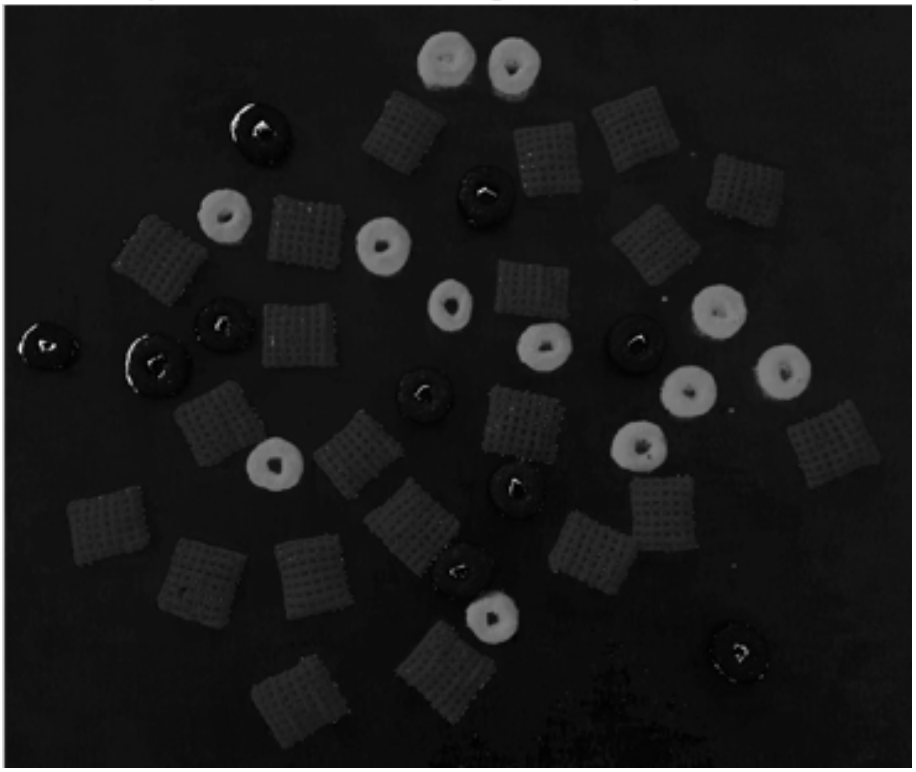


Tips for task 2:

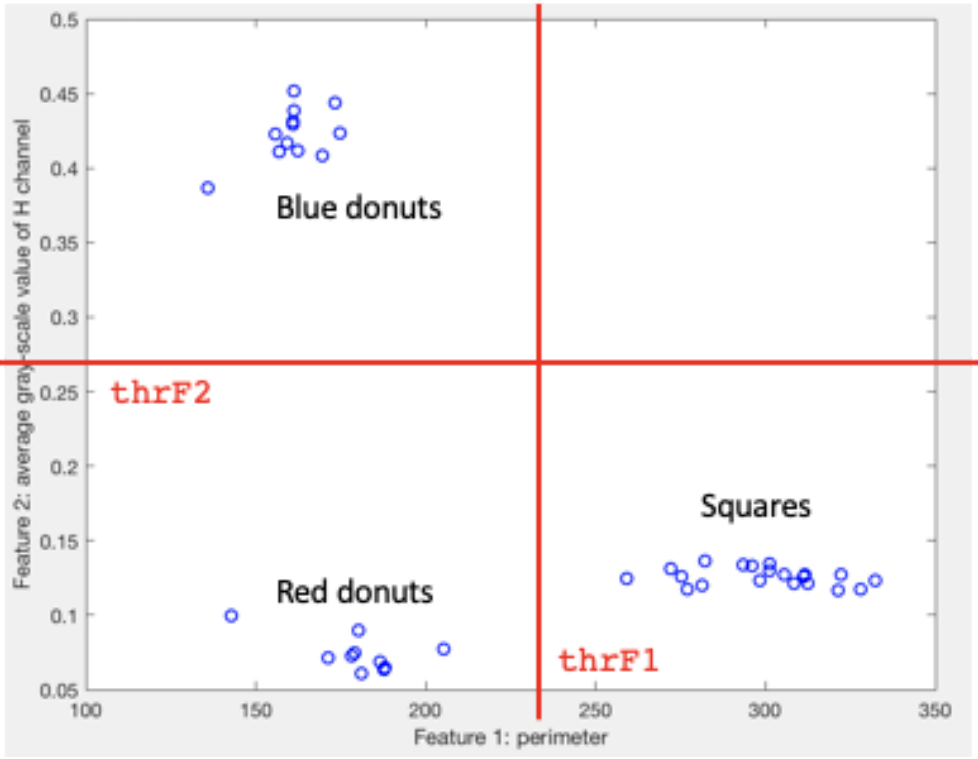
Example feature 1: Perimeter (after application of morphological operations)



Example feature 2: local, average intensity of H channel



Detected objects in our 2D feature space



Task 3 (2 points):

Write a Python program (task3.py) that incorporates all steps described of simple object and classification and count how many round and oval pills are in the example image (pills.png) attached to this document. The steps will be very similar to task 2, but you might have to select different features. Use appropriate comments in your code and description in the README to explain what you used and why. If the code works well, and the number of pills automatically reported by your program is correct, you will get 2 points for this task.

Deadline Extensions and Late Submissions:

- First deadline extension request will be granted with a 3-day extension automatically and you won't be penalized. ***Applicable to Assignments and Projects***
- For subsequent late submissions, you will lose 10% for each day late for the programming projects. This means anytime within the first 24 hours after the due date count as 1 full day, up to 48 hours is two and 72 for the third late day. Beyond that, your submission will not be graded. ***Applicable to only Assignments***

Please clarify this with the instructor if the policy is not clear before you submit late.

Statement on Academic Integrity

University -We, the Lehigh University Student Senate, as the standing representative body of all undergraduates, reaffirm the duty and obligation of students to meet and uphold the highest principles and values of personal, moral and ethical conduct. As partners in our educational community, both students and faculty share the responsibility for promoting and helping to ensure an environment of academic integrity. As such, each student is expected to complete all academic course work in accordance to the standards set forth by the faculty and in compliance with the University's Code of Conduct.

Course - The work you do in this course must be your own. This means that you must be aware when you are building on someone else's ideas—including the ideas of your classmates, your professor, and the authors you read—and explicitly acknowledge when you are doing so. Feel free to build on, react to, criticize, and analyze the ideas of others but, when you do, make it known whose ideas you are working with. If you ever have questions about drawing the line between others' work and your own, ask me and I will give you clear guidance or you may visit Lehigh Library's 'Proper Use of Information' page at http://libraryguides.lehigh.edu/plagiarism

Grade Specific - Zero assigned to the Quiz/Assignment for first offense and the student will Fail the class on second offense.

For assignments, you can discuss with peers but the code should be your own. For quizzes, no consultation with a living or non-living entity is allowed.

University COVID Policy:

To meet the challenge of teaching and learning during the COVID-19 pandemic, Lehigh instructors and students will be adopting new forms of instruction and interaction; following new guidelines around classroom behaviors; enhancing communications; and doing our best to be patient, flexible, and accommodating with each other. In remote synchronous meetings, students are expected to attend just as they would any other Lehigh class. Zoom classes work best when all students come to class ready to participate and follow the instructor's guidelines regarding use of web-cameras. You may be asked to turn your camera on during active learning sessions in Zoom. If you have a strong preference not to do so, please contact your instructor to let them know. Students should respect the in-classroom privacy of their instructors and fellow students by not taking screenshots or recording class sessions. Some instructors will record Zoom sessions; however, any recorded live sessions will be shared only with students in the class and will be deleted at the end of the semester.

In our physical classrooms, Lehigh has established a policy requiring everyone to wear face coverings when in public spaces inside buildings on our campus and to maintain social distance. This policy applies to our physical classroom. Thank you in advance for following this rule. Students who do not wear a face covering during in-class meetings will be reminded to put their face covering on. If they do not do so, they will be asked once again to do so or leave the classroom.