# ACTIVITY 7. FEATURE EXTRACTION FROM LABELED BLOBS

Julian Christopher L. Maypa
2020-07587
App Physics 157 WFY-FX-1

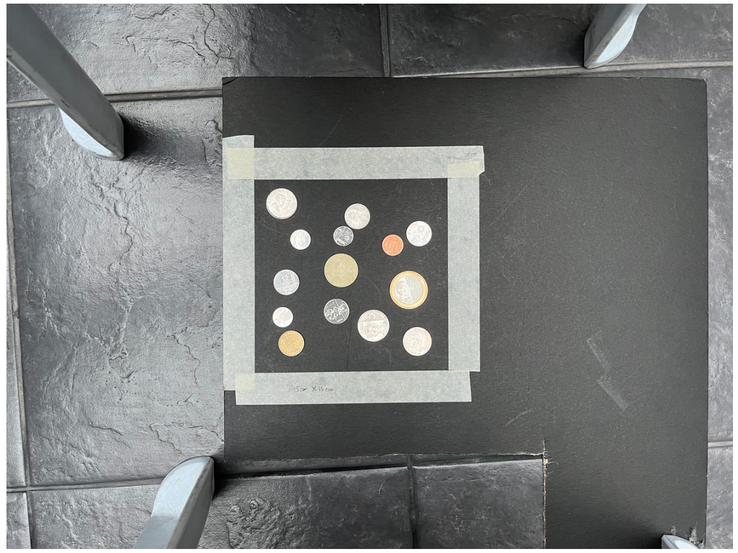
# Objectives

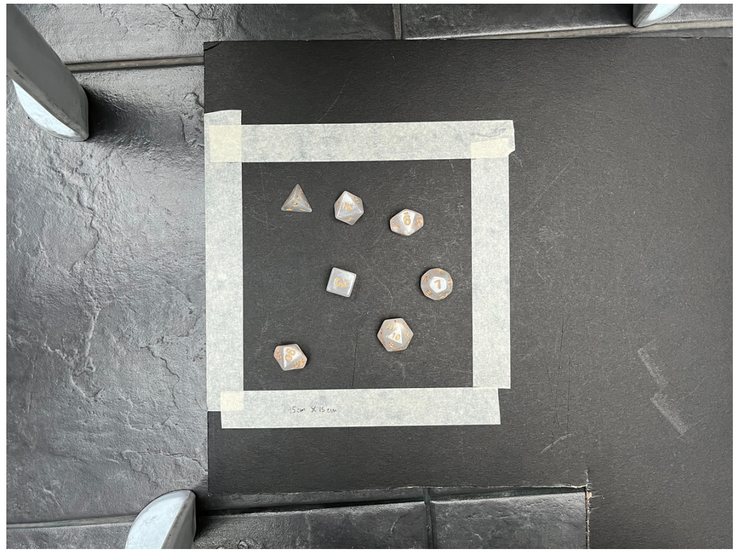
- Take a picture of objects with different shapes and sizes.
- Apply several morphological operations to clean and segment the images.
- Extract features from the objects and apply statistical analysis.

# Preparation

The objects that I used in this activity were coins of varying radii, rice grains, and dice with different shapes.

The objects were placed in a 15 cm by 15 cm square on the black side of an illustration board. Before processing the image, I cropped the images so that the image would only include the region inside the square. Also. I took the pictures with the help of my labmates from IPL-VIP, namely Jonabel Baldres and Johnenn Manalang.





# Coins with varying radii

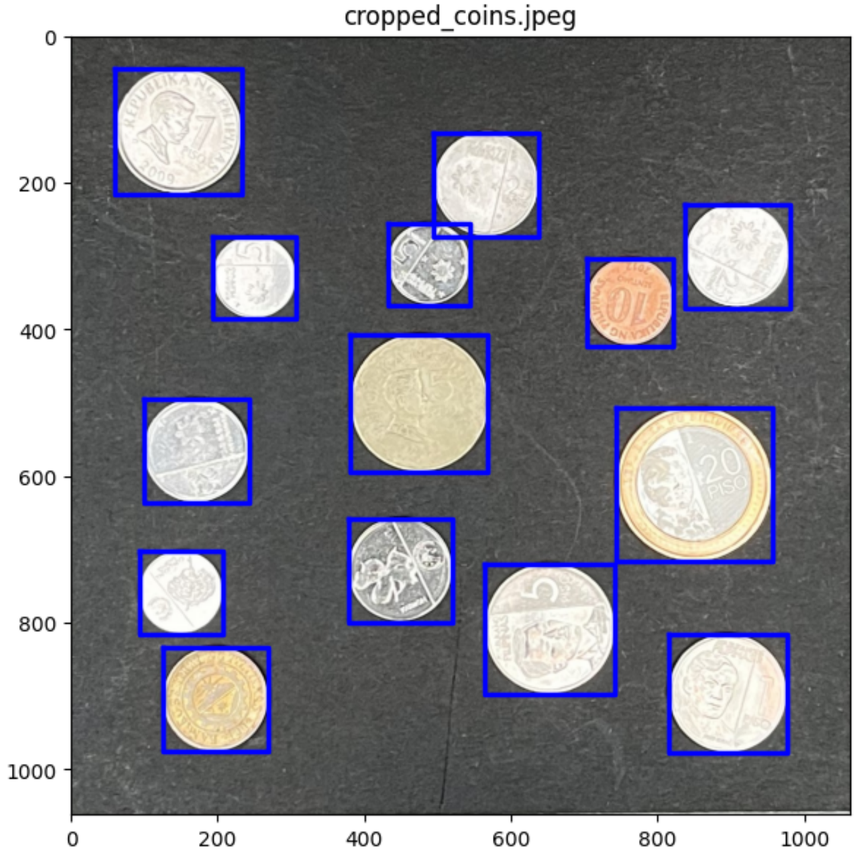
The coins that were used for this part in the activity are a new 5 centavo coin, old 10 centavo coin, new and old 25 centavo coins, old and new 1 peso coins, old and new 5 peso coins, new 10 peso coin, and a 20 peso coin.

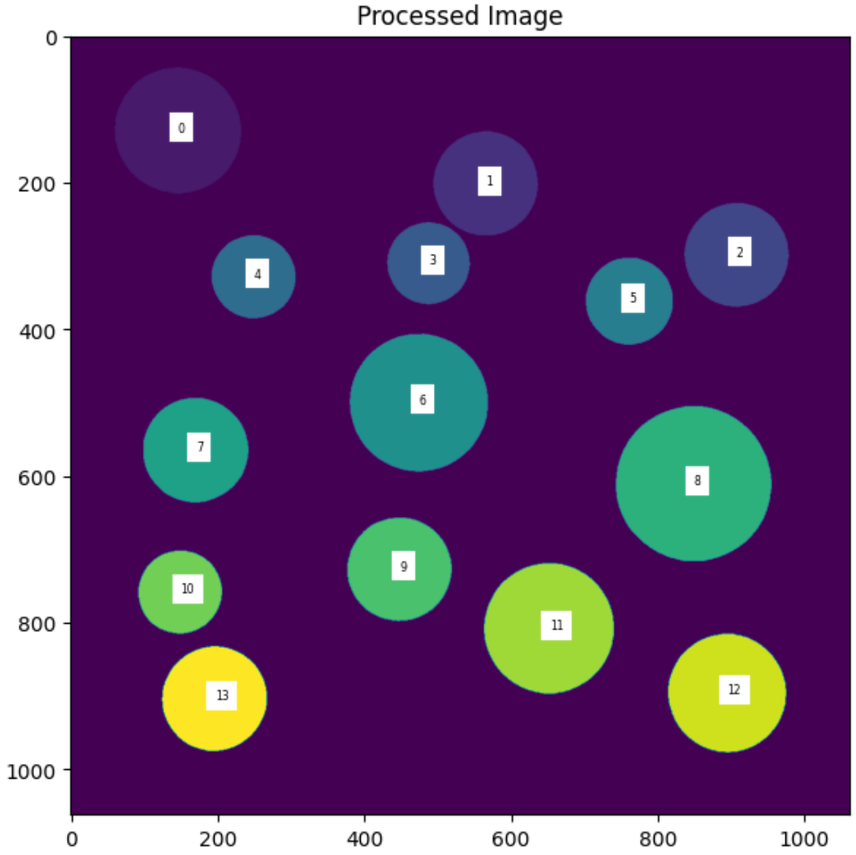
The processes that I did to detect the coins are:

- Converted the image to grayscale
- Median blur with a 3 x 3 kernel. I used median blur because it preserves the edges of the objects [1].
- Thresholded the image by setting all gray values less than the 77.9 th percentile to be 0 and the rest are set to be 1.
- Closing with a 7 by 7 circular structuring element.
- Opening with a 9 by 9 circular structuring element.

I then used scikit-image's *regionprops* and *regionprops_table* to extract the following features: area, centroid, bounding box, perimeter, eccentricity, minor axis, and major axis.

With these measurements, I compared the theoretical area, perimeter, and diameters of the coins with my measurements to validate my results. Also, the measurements were initially in terms of pixels. But since I know that my image is 15 cm by 15 cm, I converted the units by getting the ratio of the total area of the image in cm by the total number of pixels of the image.



| | Coin Type | Measured_diameter [cm] | Measured_area [cm^2] | Measure_perimeter [cm] | Theo_diameter [cm] | Theo_area [cm^2] | Theo_perimeter [cm] | error_diameter [%] | error_area [%] | error_perimeter [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | old 1 peso | 2.427004 | 4.625906 | 8.054476 | 2.4 | 4.523893 | 7.539822 | 1.125160 | 2.254967 | 6.825811 |
| 1 | new 25 centavo | 2.006447 | 3.161807 | 6.620896 | 2.0 | 3.141593 | 6.283185 | 0.322347 | 0.643441 | 5.374836 |
| 2 | new 25 centavo | 1.997001 | 3.132082 | 6.612622 | 2.0 | 3.141593 | 6.283185 | 0.149969 | 0.302730 | 5.243154 |
| 3 | new 5 centavo | 1.576168 | 1.951068 | 5.207291 | 1.6 | 2.010619 | 5.026548 | 1.489495 | 2.961853 | 3.595759 |
| 4 | new 5 centavo | 1.601361 | 2.013909 | 5.272062 | 1.6 | 2.010619 | 5.026548 | 0.085060 | 0.163608 | 4.884337 |
| 5 | old 10 centavo | 1.671337 | 2.193854 | 5.484930 | 1.7 | 2.269801 | 5.340708 | 1.686070 | 3.345966 | 2.700438 |
| 6 | old 5 peso | 2.649494 | 5.513262 | 8.753005 | 2.7 | 5.725553 | 8.482300 | 1.870609 | 3.707767 | 3.191411 |
| 7 | new 25 centavo | 2.014503 | 3.187143 | 6.629170 | 2.0 | 3.141593 | 6.283185 | 0.725141 | 1.449909 | 5.506518 |
| 8 | 20 peso | 2.986705 | 7.005889 | 9.857296 | 3.0 | 7.068583 | 9.424778 | 0.443168 | 0.886944 | 4.589156 |
| 9 | new 25 centavo | 1.995417 | 3.127095 | 6.580947 | 2.0 | 3.141593 | 6.283185 | 0.229133 | 0.461484 | 4.739019 |
| 10 | new 5 centavo | 1.600965 | 2.012911 | 5.272062 | 1.6 | 2.010619 | 5.026548 | 0.060339 | 0.113998 | 4.884337 |
| 11 | new 5 peso | 2.504144 | 4.924950 | 8.267345 | 2.5 | 4.908739 | 7.853982 | 0.165769 | 0.330252 | 5.263100 |
| 12 | new 1 peso | 2.271968 | 4.053952 | 7.495771 | 2.3 | 4.154756 | 7.225663 | 1.218799 | 2.426248 | 3.738173 |
| 13 | old 25 centavo | 2.012621 | 3.181158 | 6.640871 | 2.0 | 3.141593 | 6.283185 | 0.631071 | 1.259405 | 5.692744 |

I tabulated the measured diamater, area, and perimeters of the coins and their theoretical counteraparts. The measured diameter was obtained from getting the average of the major and minor asix of the coins, I did this because I know that the coins are circular so the major and minor axes should be approximately equal to each other. I then got the theoretical diameters of the coins from a list provided by the Bangko Sentral ng Pilipinas [2] and Philippine Numismatics [3]. With the theoretical diameters, I calculated the theoretical perimeter and area of the coins. I then compared the theoretical values with my measured values.

The three right-most columns show the percent errors between the measured and theoretical values of the coins. All of them are below 10%, therefore all the measurements are within an acceptable percent error. I can identify three main sources of error that affected my measurements. The first source is the illustration board. I noticed that the illustration board did not lay completely flat as we took the picture. This distrortion in the illustration board introduces an error when converting the pixel measurements to cm measurements. The second source of error is when applying morphological opperations [4]. My choice of structuring elements for the thresholding, filtering, and other morphological operations could have removed some features of the coins, which consequently affected the measurements. The third source of error is due to the distortion of the image caused by the camera. We minimized this error by putting the objects at the center of the image where the distortion is at its minimum [5].

But even with all of these sources of error, I have still successfully extracted the features of the coins within an acceptable percent error.

# Rice Grains

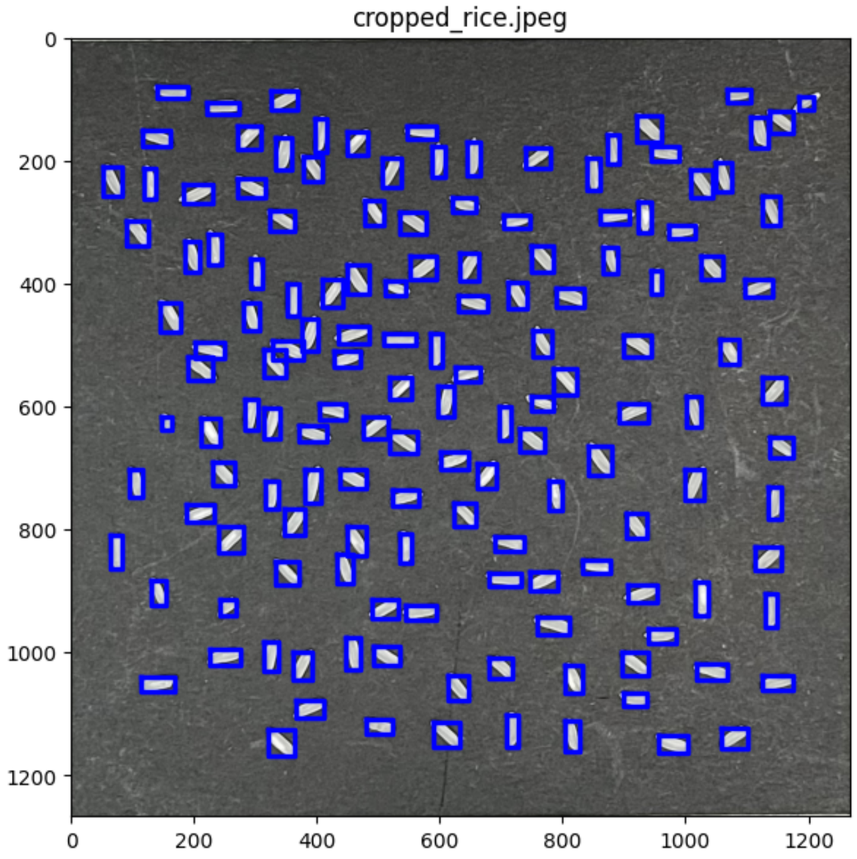
Now that I have validated the measurements extracted by my program fall within an acceptable percent error, I will now try to apply it on rice grains. I'm using rice grains because I recall that an experiment in Physics 106.1 required us to measure the length and width of about 100 rice grains just to get the average values. Although I did not personally measure it because of the online setup during the pandemic, I can imagine that it would be very tedious to manually do it. Therefore automating it would be very efficient and convenient.

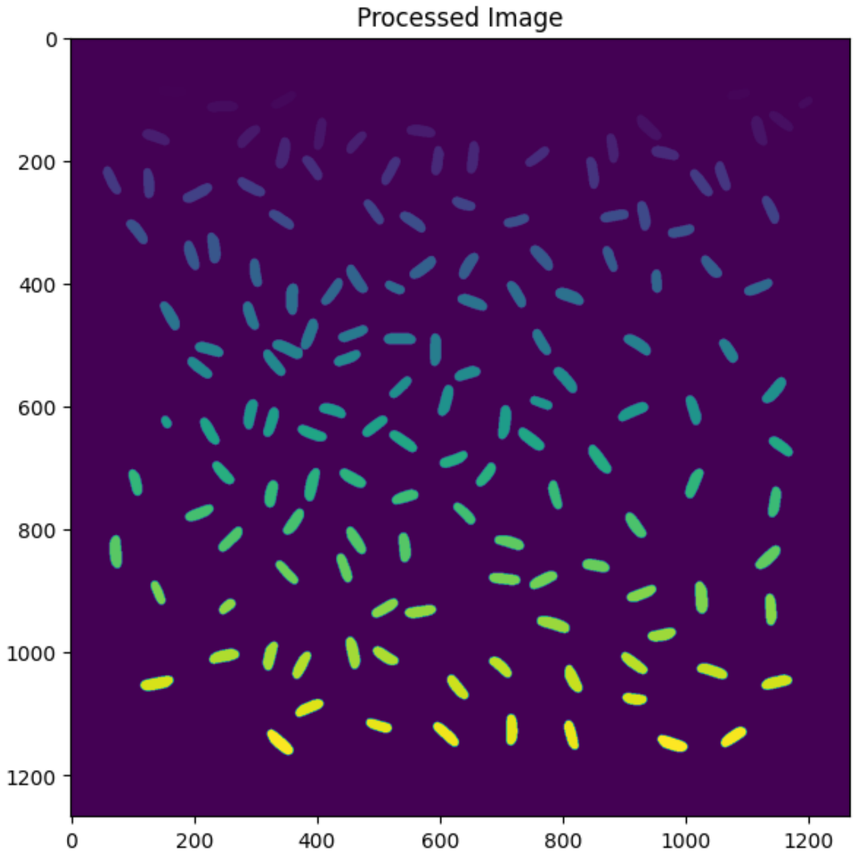
The processes that I did to detect the rice grains are:

- Converted the image to grayscale
- Median blur with a 13 x 13 kernel. I used median blur because it preserves the edges of the objects [1].
- Thresholded the image by setting all gray values less than the 92.65 th percentile to be 0 and the rest are set to be 1.
- Opening with a 13 by 15 elliptical structuring element.

I then used scikit-image's *regionprops* and *regionprops_table* to extract the following features: area, centroid, bounding box, perimeter, eccentricity, minor axis, and major axis.

With these measurements, I converted the units of the area, perimeter, major axis (length), and minor axis (width) of the rice grains from pixels to centimeters using the ratio of the area of the square to the total pixels of the image. I then graphed the distribution of the rice grains' area, perimeter, length, and width.



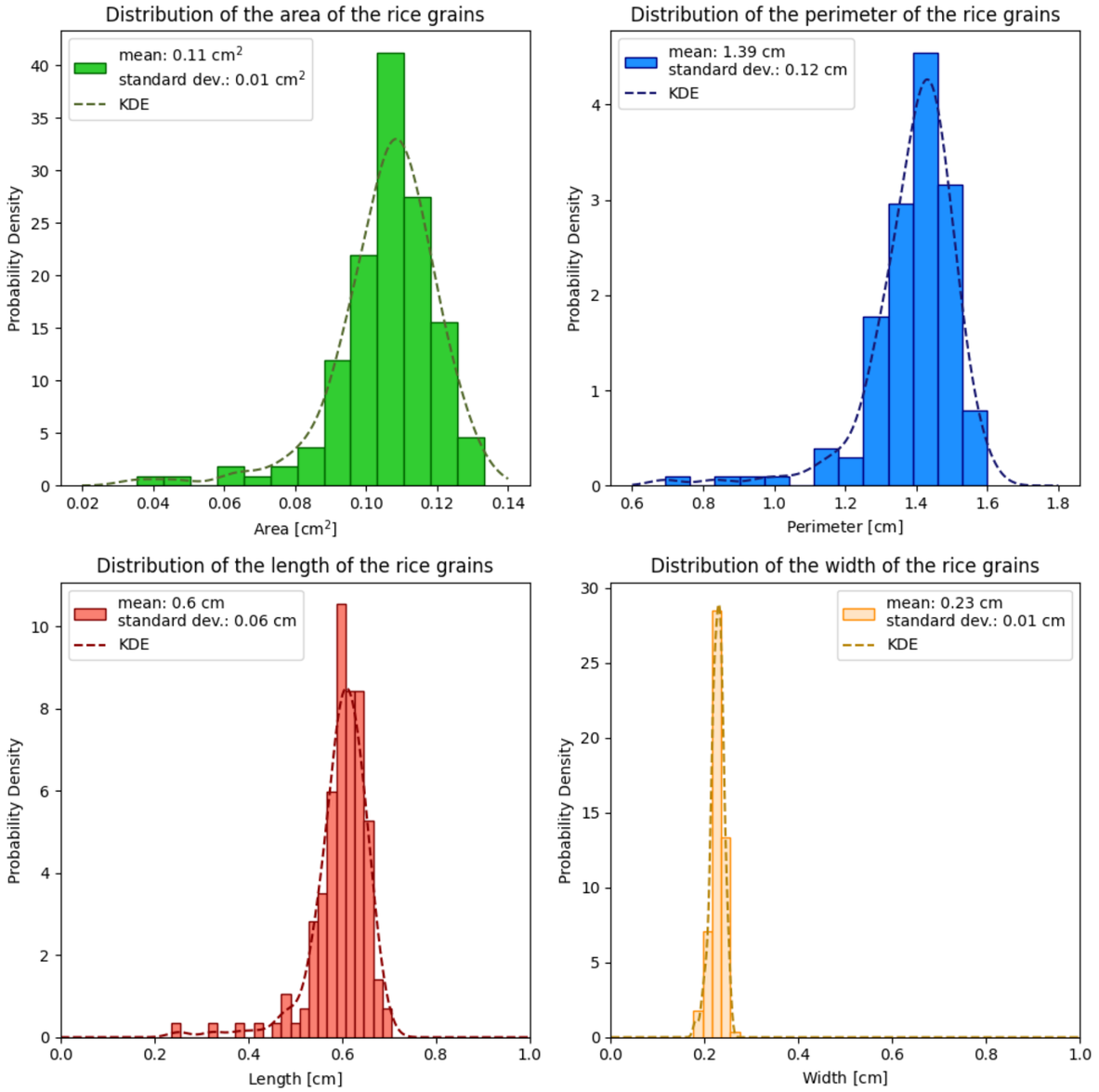
To the right are the distributions of the area (top left), perimeter (top right), length (bottom left) and width (bottom right) of the rice grains. The graphs show the histogram and the Kernel Density Estimates (KDE) of the rice grains. The KDEs were made by using scipy's *gaussian_kde* function and the bandwidth estimator that I used was Scott's method and it was assumed that the data points had equal weight. I used these parameters because these were the function's default parameters.

First looking at the distribution of the area of the rice grains, it has a mean of 0.11 cm^2 and a standard devation of 0.01 cm^2. From this information, we can see that the rice grains has a uniform area because its standard deviation is one order of magnitude smaller than the mean,

For the perimeter of the rice grains, it has a mean of 1.39 cm and a standard deviation of 0.12 cm. Similar to the rice grains' area, the rice grains have a uniform perimeter because its standard deviation is one order of magnitude smaller than the mean.

Finally, looking at the length and width of the rice grains, it can be seen that the rice grains have a mean of 0.6 cm and a mean of 0.23 cm for length and width respectively. The standard deviation of the lengths and widths are 0.06 cm and 0.01 cm respectively. This means that the lengths and widths of the rice grains are somewhat uniform because the standard deviation is one order of magnitude smaller than the mean.

These results show that my code has successfully obtained the mean area, perimeter, length, and width of the rice grains. Although the sources of error that I previousy mentioned in slide 5 are still present in the measurements of the rice grains, my measurements still fall within an acceptable percent error. And looking at the histograms, there are only a few outliers in the data, which represent the small crushed grains which can be seen in the previous slide.



# Dice with different shapes

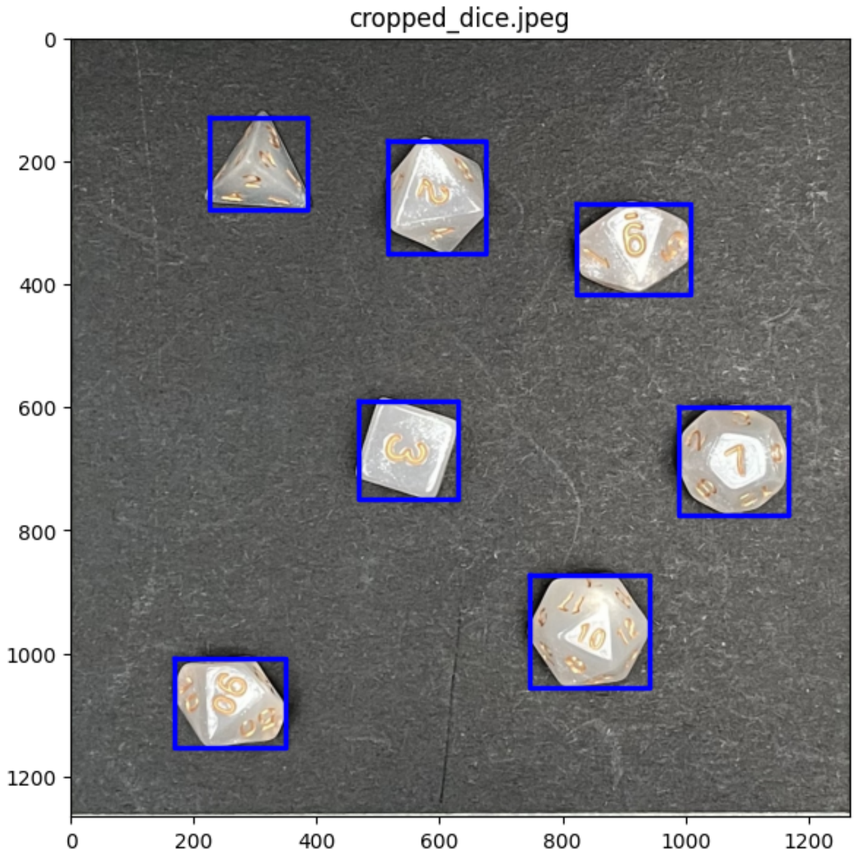
Playing table top games is one of my favourite hobbies. And one of the games that I play is Dungeons and Dragons and we use specific types of dice to simulate probabilities such as succesfully persuading someone, the amount of damage taken, blocking an attack, and sneaking through someone. So with this, I have a set of *n* sided dice and I want to extract their features. My goal is to determine which feature of an *n* sided dice spearates it from other the other types of dice.

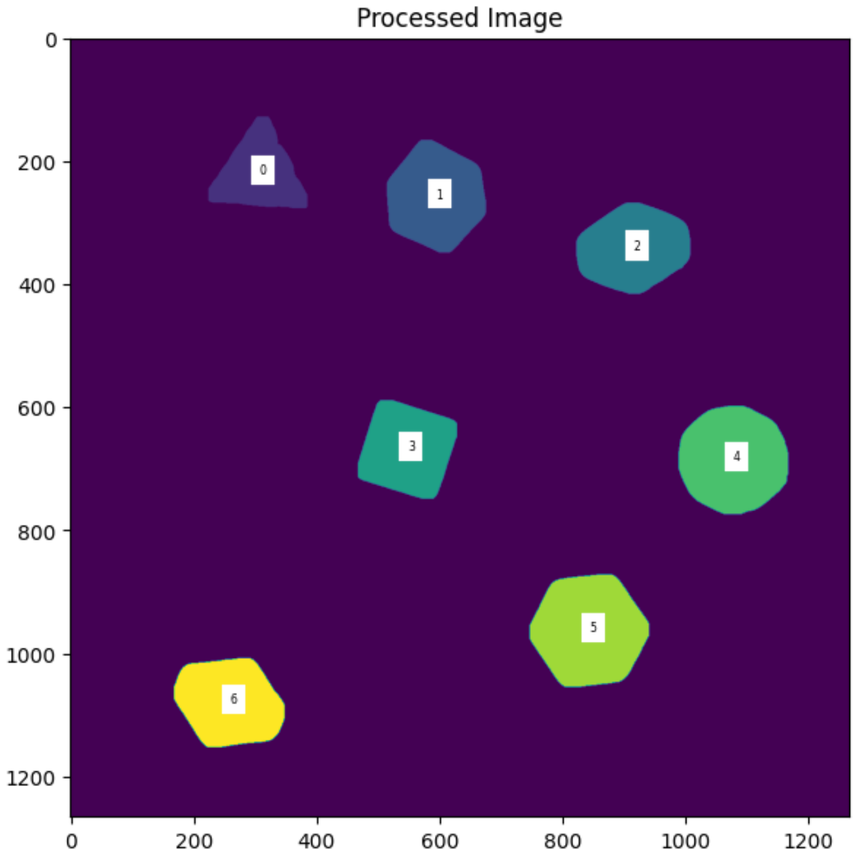
The processes that I did to detect the dice are:

- Converted the image to grayscale
- Median blur with a 15 x 15 kernel. I used median blur because it preserves the edges of the objects [1].
- Thresholded the image by setting all gray values less than the 89.8 th percentile to be 0 and the rest are set to be 1.
- Opening with a 18 by 20 rectangular strucuring element.

I then used scikit-image's *regionprops* and *regionprops_table* to extract the following features: area, centroid, bounding box, perimeter, eccentricity, minor axis, and major axis.

With these measurements, I converted the units of the area, perimeter, major axis (length), and minor axis (width) of the dice from pixels to centimeters. I then got the standard deviation of the area, perimeter, eccentricity, major axis, and minor axis to determine which feature sets apart the dice the most.



The raw data collected from the image (the area, perimeter, and axes are already converted to cm)

| | Unnamed: 0 | area | centroid-0 | centroid-1 | bbox-0 | bbox-1 | bbox-2 | bbox-3 | perimeter | eccentricity | orientation | axis_major_length | axis_minor_length |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2.028677 | 219.777247 | 307.158536 | 129 | 225 | 278 | 386 | 6.246482 | 0.154859 | 0.687910 | 1.746257 | 1.725191 |
| 1 | 1 | 3.050948 | 259.257074 | 593.960475 | 167 | 515 | 350 | 676 | 6.646740 | 0.369301 | 0.042321 | 2.053060 | 1.907929 |
| 2 | 2 | 2.858483 | 342.585944 | 915.109125 | 269 | 823 | 417 | 1008 | 6.504067 | 0.674653 | -1.365827 | 2.225544 | 1.642752 |
| 3 | 3 | 2.651137 | 669.483400 | 547.582896 | 590 | 468 | 750 | 629 | 6.545914 | 0.139936 | 1.553196 | 1.877042 | 1.858573 |
| 4 | 4 | 3.490066 | 686.081413 | 1077.558385 | 599 | 989 | 775 | 1167 | 6.953113 | 0.240544 | 1.304343 | 2.140160 | 2.077321 |
| 5 | 5 | 3.796102 | 963.944492 | 842.908176 | 873 | 747 | 1056 | 941 | 7.394027 | 0.168985 | 0.854485 | 2.219970 | 2.188044 |
| 6 | 6 | 2.842900 | 1080.694583 | 258.399486 | 1009 | 169 | 1154 | 349 | 6.494949 | 0.627484 | 1.204205 | 2.169770 | 1.689446 |

The standard deviation of the measured area, perimeter, eccentricity and axes.

| | Number of dice | Area stdev [cm^2] | Perimeter stdev [cm] | Eccentricity stdev | minor axis stdev [cm] | major axis stdev [cm] |
|---|---|---|---|---|---|---|
| 0 | 7 | 0.531183 | 0.349807 | 0.21008 | 0.189327 | 0.170464 |

The table above shows that standard deviation of the area, perimeter, eccentricity, minor axis, and major axis. These standard deviations indicate how much of the dice defer in terms of the these parameters. From the table, it can be seen that the area has the highest standard deviation. This means that the area of the dice is a good parameter in determining how many sides it has.

One application for this is for online gaming platforms. There is a website called Spell Table where people can play Magic the Gathering (another table top game and my personal favorite card game). The goal of the website is to stream the cards of the players (sort of like a zoom call, but instead of the players' faces, it's the cards that are being shown on video). The website can also identify the card from the video. So with this, being able to identify the type of dice is a step closer to making a platform where players can play table top games with actual dice. This can be further improved with machine learning to detect the value of the dice when it is rolled.

# Reflection

Overall, I believe that the results I got are correct. The measurements of the coins fall within an acceptable percent error when compared to their theoretical values. For the rice grains, a low standard deviation in the measurements of the area, perimenter, length, and width shows that there are only a few outliers. Visually inspecting the rice grains also show that most of them have a uniform area, perimeter, length, and width. For the dice, it has similar accuracy as my coin measurements. But looking at the pyramidal dice (the four sided dice), the edges are a bit jagged. But the main purpose of the measurement was to determine the differences of the features of the dice. So the jagged edges of the four sided dice will have little effect in my analysis.

The most tedious part for me is collecting the pictures. We had to find the perfect setup to minimize the glare from the coins and the dice. We took the picture in the middle of the research wing and it was really hot there so we had to endure it. For the coding part, I reused my code from the last activity and just added a few lines for the *regionprops* function.

I'd like to thank my instructors, Sir Rene Principe Jr. and Sir Kenneth Leo, for guiding me throughout the activity. I would also like to thank my professor, Ma'am Jing, for guiding me in my coding while my classmates and I worked in R202. I would also like to acknowledge my classmates: Abdel, Johnenn, Jonabel, Richmond, Lovely, Hans, Genesis, Jeruine, Rusher, and Ron for helping me complete this activity.

# Self Grade

| | | |
|---|---|---|
| **Technical Correctness** | I understood the lesson and met all the objectives. My results are complete and I got the expected results. | 35 |
| **Quality of Presentation** | The images I added to this report are of good quality and all the graphs are properly labelled. My code is also properly organized and labelled. | 35 |
| **Self Reflection** | I got the expected results, and acknowledged the contributions of my peers while doing this activity. I also properly cited online references. | 30 |
| **Initiative** | Apart from doing the required tasks, I also helped my classmates with their code and helped them by cross-referencing my results with theirs. | 10 |
| | **Total** | 110 |

# References

[1] Sekhon, M. (2019, August 11). Image filters in Python. Medium. https://towardsdatascience.com/image-filters-in-python-26ee938e57d2

[2] Bangko Sentral ng Pilipinas. (2022). The Philippine New Generation Currency Coin Series. Retrieved April 28, 2023, from https://www.bsp.gov.ph/Coins%20and%20Notes/Coins/NGCCS/NGCCoins.pdf

[3] Rafanan, J. (n.d.). Philippine Coin Sizes. Philippine Numismatics. Retrieved May 26, 2023, from https://www.numismatics.ph/resources/philippine-coin-sizes.html

[4] Mondal, R., Dey, M. S., & Chanda, B. (2020). Image restoration by learning morphological opening-closing network. Mathematical Morphology - Theory and Applications, 4(1), 87–107. https://doi.org/10.1515/mathm-2020-0103

[5] ISO/TC 42, “ISO 17850:2015 Photography – Digital cameras – Geometric distortion (GD) measurements,” First edition, 2015, https://www.iso.org/standard/60819.html