Determining Rice Grain Orientation with Computer Vision

David M. Vermillion

Colorado State University Global

CSC 510-1

Dr. Bingdong Li

July 7, 2024

Determining Rice Grain Orientation with Computer Vision

The rice grain image dataset was obtained from Kaggle for this project (Koklu, 2022). The dataset contains 15,000 clean images of 5 varieties of rice for 75,000 total images. The initial approach is to build a search algorithm for rice varieties. Within each variety, the images of rice grains are classified by orientation category as either vertical or not vertical. This is defined on a First Oder Logic or symbolic level as *orientation_vertical(rice_grain, ±30°) else orientation_not_vertical*.

Experimental Setup

The images were saved in their default file structure at the same directory level as the script. Portions of required packages were set for import at the beginning of the script which included *Sci-kit Image, math, os,* and *matplotlib*. The directory was set on line 15 to ensure the script executed with the correct relative working directory for file searches. This was unnecessary while prototyping in a Jupyter Notebook, but proved necessary once executing as a python script. Four additional functions were defined in the script, one for finding the correct file using a directory tree walk, one for classifying orientation, and two for receiving user input. These are then stored as intermediate values to pull up the file. The directory tree walk was useful to ensure the filepath was correct to the desired image by automatically generating the relevant directory based on user-chosen parameters.

Upon selecting the image, line 65 converts the RGB image to grayscale for processing. Line 71 creates an outline of the rice grain. Because the outline is an intermediate step, it is intentionally not displayed in the final output. From here, line 75 creates an elliptical approximation of the rice grain, allowing an orientation calculation in radians based on the major and minor axes. Line 75 takes the longest out of each of these processes because multiple

attempts are required before a best-fit ellipse can be found. This process typically takes 3-5 seconds. The resulting ellipse is drawn in line 85 and placed on the original image on line 86. Lines 89 and 90 create a vertical reference line for easier visualization of orientation. Finally, lines 93-100 create the finished output graph. These are all shown on GitHub (Vermillion, 2024).

Experimental Findings

Using a Hough Elliptical transform worked, but presented challenges. For example, an accuracy of 20 and threshold of 200 worked for *Arborio (6).jpg* and *Basmati (12).jpg*, which were each chosen because of their highly elliptical formation. However, *Arborio (1).jpg* created a null set under those parameters, necessitating a change to a threshold of 150. After testing another 5 images chosen for non-uniformity, that threshold appears to hold well. Some of the ellipses are off by up to 20 degrees from what a human would likely draw defining its axis. However, that is still within the 30 degree threshold, which means that it should provide valid results. Additionally, some edge cases, such as *Jasmine (24).jpg* count orientation as outside the ±30° threshold, but graphically suggest they should be at or below 30°. The cause of this unexpected behavior was not identified during testing, but was left to a future iteration for improvement. This result serves as a basis for a future assembly-line product.

Figure 3: Arborio (1)

Figure 2: Arborio (6)

Figure 1: Basmati (12)

Figure 4: Jasmine (24)

Experimental Results

Most runs executed in 6 to 10 seconds after user input. Some edge-case orientations created issues. However, obvious orientations within ±30° or >60° created clear results, such as those shown in *Figures 5 and 6*. *Figure 7* shows a correct edge-case classification at 31° from vertical, although it does show how arbitrary that choice was. *Figure 8* shows the terminal output for a run of Jasmine (24). All five varietals (Arborio, Basmati, Ipsala, Jasmine, and Karacadg) from 1 to 15,000 are valid numbers to execute. However, as currently written, executing every single image would take 125 hours and is therefore not recommended.

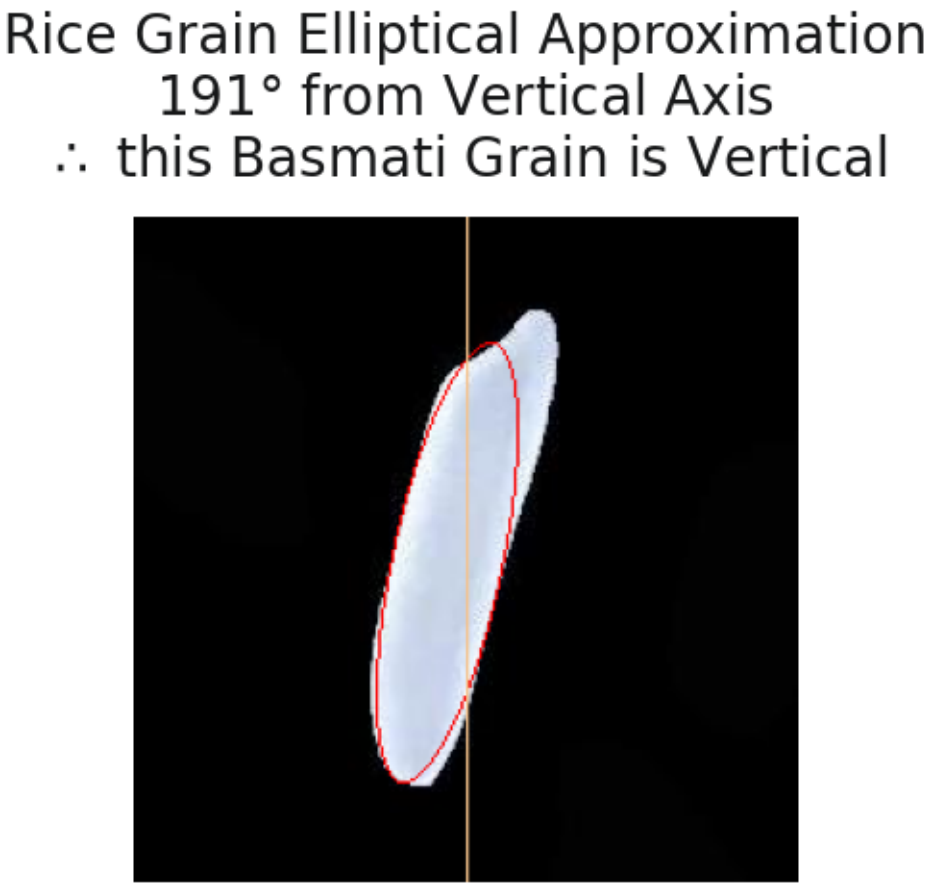

Figure 5: Basmati (1) Classification Result

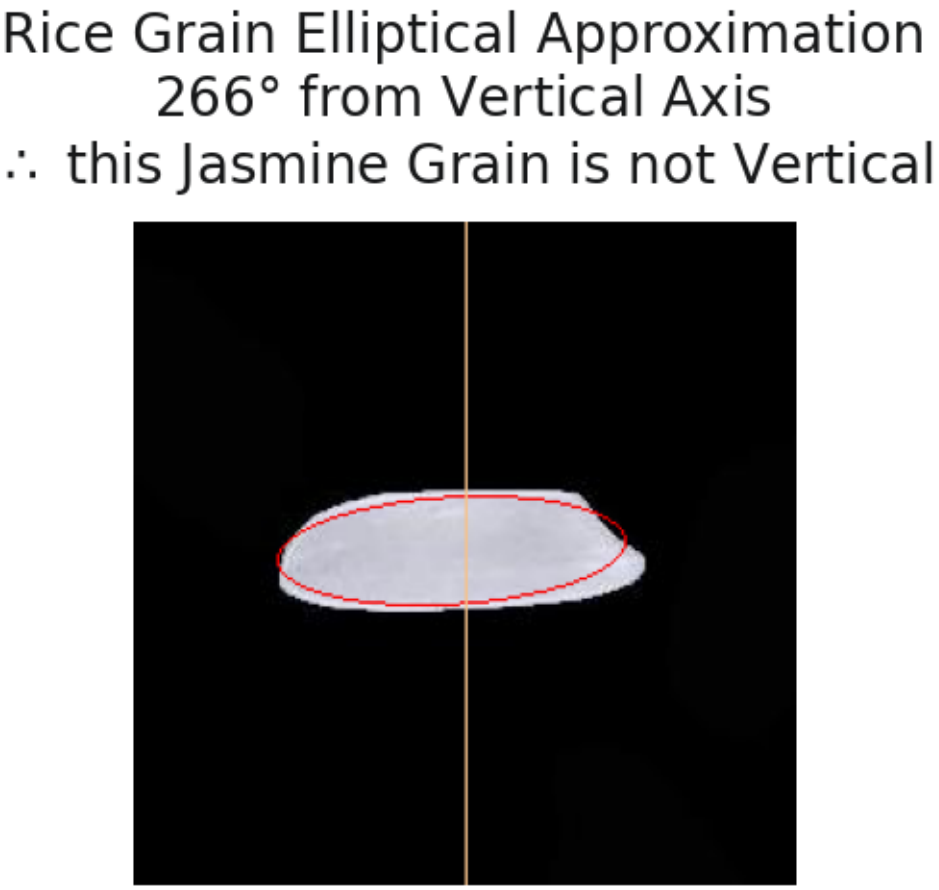

Figure 6: Jasmine (1) Classification Result

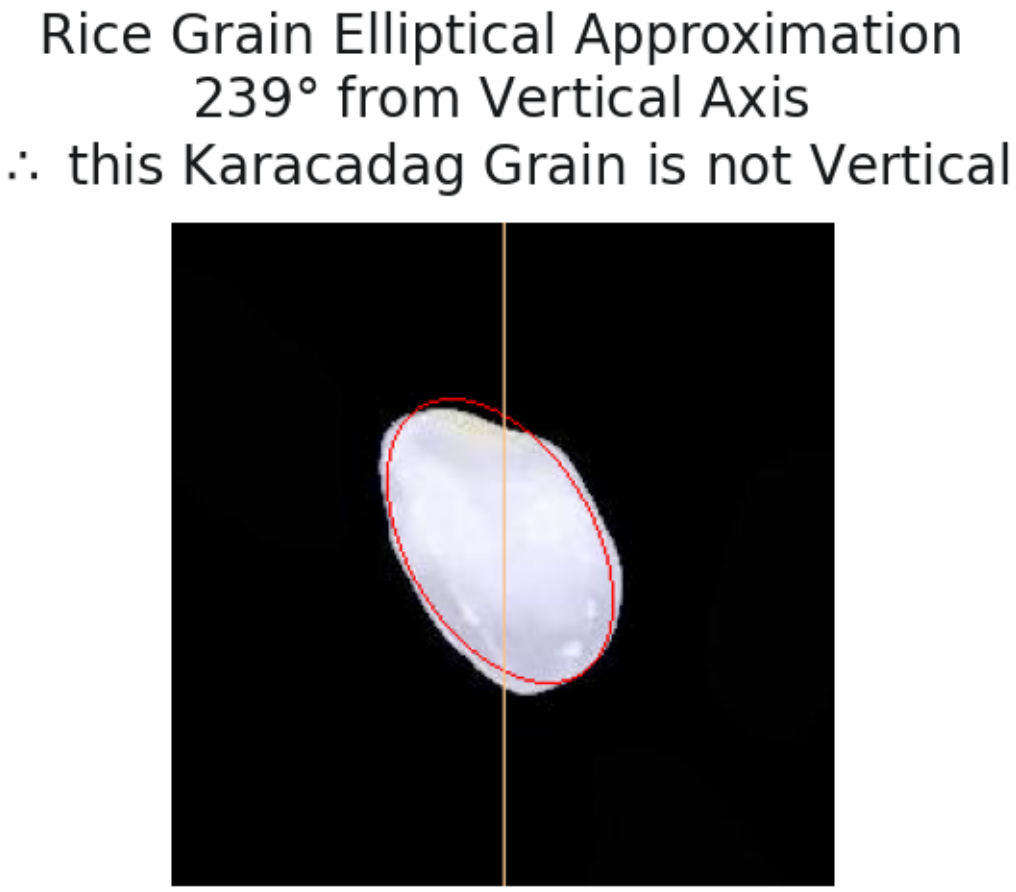

Figure 7: Karacadag (10) Edge Classification

```
(csc510redo) davidmvermillion@Davids-MBP CSUGHomework % /Users/davidmvermillion/opt/anaconda3/envs/csc510redo/bin/python /Users/davidmvermillion/Documents/G
itHub/CSUGHomework/CSC510/PortfolioProject/rice_orientation_classifier.py

Choose a whole number between 1 and 15,000

24

Choose one of the following rice varietals:
Arborio
Basmati
Ipsala
Jasmine
Karacadag

Jasmine

This should take between 1 and 10 seconds per query.
Please close any chart programs from this query to reset the script before running your next query.

(csc510redo) davidmvermillion@Davids-MBP CSUGHomework %
```

Figure 8: Terminal Output of Running Jasmine (24) through the Script

Code

```
# Setup
import matplotlib.pyplot as plt
from skimage import color
from skimage.feature import canny
from skimage.transform import hough_ellipse
from skimage.draw import ellipse_perimeter, line
from skimage.io import imread
from math import degrees
from math import pi
from os import walk, path, chdir
from os.path import abspath, dirname

# Set directory to current script location
# https://stackoverflow.com/a/69556612/13801562
chdir(dirname(abspath(__file__)))

# Functions
# https://www.tutorialspoint.com/file-searching-using-python
def find_files(filename, search_path):
   result = []

   # Walking top-down from the root
   for root, dir, files in walk(search_path):
      if filename in files:
         result.append(path.join(root, filename))
         result = str(result)
         result = result.replace("'", "")
         result = result.replace("[", "")
         result = result.replace("]", "")
   return str(result)

def grain_orientation(radians):
    if radians >= pi/3 and radians <= 2*pi/3 or radians >=
4/3*pi and radians <= 5/3*pi:
        return 'Vertical'
    else:
        return 'not Vertical'

# https://stackoverflow.com/a/23326219/13801562
def number():
    while True:
        try:
```

```
                chosen_number = int(input('\nChoose a whole
number between 1 and 15,000\n\n'))
                break
            except:
                print('Your answer must be an integer between 1
and 15,000. Try again!\n')
        return chosen_number

    def rice_options():
        while True:
            try:
                rice = input('\nChoose one of the following rice
varietals:\nArborio\nBasmati\nIpsala\nJasmine\nKaracadag\n\n')
                break
            except:
                print('\nYour answer must be one of the
following rice varietals:\nArborio\nBasmati\nIpsala\nJasmine\
nKaracadag\n\n')
        return rice

    # User Input Intermediate values
    image_number = number()
    rice_choice = rice_options()
    file = find_files('{} ({}).jpg'.format(rice_choice,
image_number), 'Rice_Image_Dataset')
    print('\nThis should take between 1 and 10 seconds per
query.\nPlease close any chart programs from this query to reset
the script before running your next query.\n')

    # Data Prep
    image_rgb = imread(str(file))
    image_gray = color.rgb2gray(image_rgb)

    # Orientation Algorithm
    #
https://scikit-image.org/docs/stable/auto_examples/segmentation/p
lot_regionprops.html
    #
https://scikit-image.org/docs/stable/auto_examples/edges/plot_cir
cular_elliptical_hough_transform.html
    # Find the edges of the rice grain
    edges = canny(image_gray, sigma = 2.0, low_threshold = 0.55,
high_threshold = 0.8)

    # Perform a Hough Ellipse Transform to find a valid
elliptical approximation of the grain
```

```
    # Threshold of 150 holds for a small sample of non-
symmetrical rice grains
    result = hough_ellipse(edges, accuracy = 20, threshold =
150)
    result.sort(order = 'accumulator')

    # Estimated parameters for the ellipse
    best = list(result[-1])
    yc, xc, a, b = (int(round(x)) for x in best[1:5])
    orientation = best[5] # This is what really matters for the
program

    # Result
    # Draw the ellipse on the original image
    cy, cx = ellipse_perimeter(yc, xc, a, b, orientation)
    image_rgb[cy, cx] = (250, 0, 0)

    # Draw vertical reference line
    vertical_line = line(0, 125, 249, 125)
    image_rgb[vertical_line] = (246, 194, 139)

    # Plot the rice grain and ellipse
    fig, ax = plt.subplots()
    ax.set_title('Rice Grain Elliptical Approximation\n${:.0f}\\
degree$ from Vertical Axis\n$\\therefore$ this {} Grain is
{}'.format((degrees(orientation) + 90), rice_choice,
grain_orientation(orientation)),
                 fontsize = 20, pad = 15).set_color('#171819')
    ax.imshow(image_rgb)
    # https://stackoverflow.com/a/25864515/13801562
    ax.axis('off')
    plt.tight_layout()
    plt.show()
```

Discussion

This approach allows for a branching file walk to find the desired file, which counts as an informed search. Once the image is found, it is iteratively processed for elliptical approximation using a simple machine-learning optimization approach. While this works on clean images of unbroken single rice grains as a proof of concept, additional work would be required to provide useful information for flow-rates on a conveyer belt in a processing plant. The amount of rice flowing at any given time exceeds the amount of reasonable processing possible from this

approach (Quora Team, 2021). With a conservative estimate of 30,000 grains per kilogram and no need for visualizing the resulting orientation, each kilogram can be classified in 50 sequential hours. Assuming the conveyer belt is moving 1 kilogram 1 meter per second, this would require 50,000 independent processes to be close to real-time predictable. Even so, it would likely still be processing far slower than the rice is moving. Therefore, while this is an interesting thought-experiment, a significantly different approach would be required for practicality on a real-world processing line. Examples of other approaches include double-stream convolutional neural networks and vector similarity searches (Dolata et al., 2017; Özal, 2023).

Conclusion

The rice grain image dataset was obtained from Kaggle and contains 75,000 individual images, 15,000 images of five rice varietals. These images were processed to determine verticality based on a threshold of ±30°. The script was set up with 100 lines of code. Verticality was based on the orientation of a machine learning elliptical approximation of the target rice grain. This was then displayed in an image with the result clearly shown in the image title. Most runs took less than 10 seconds. However, while this was not fast enough to be useful for a real-world prediction of realtime grain processing, it does show a proof of concept for different approaches to the same problem set.

References

Dolata, P., Mrzygłód, M., & Reiner, J. (2017). Double-stream Convolutional Neural Networks for Machine Vision Inspection of Natural Products. *Applied Artificial Intelligence*, *31*(7/8), 643–659. https://doi.org/10.1080/08839514.2018.1428491

Koklu, M. (2022). *Rice Image Dataset*. https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset

Özal, S. (2023, October 16). Vector Search For AI — Part 1—Vector Similarity Search Algorithms. *Medium*. https://medium.com/@serkan_ozal/vector-similarity-search-53ed42b951d9

Quora Team. (2021). *How many grains of rice are in a 1kg bag of rice?* Quora. https://www.quora.com/How-many-grains-of-rice-are-in-a-1kg-bag-of-rice

Vermillion, D. (2024, July 3). *Rice_orientation_classifier.py*. https://github.com/davidmvermillion/CSUGHomework/blob/main/CSC510/PortfolioProject/rice_orientation_classifier.py#L19