Optimization of Center of Sphere Detection in a Granular Pile

Paul M. Hoffmann
[1]*University of California, Davis, Department of Physics*
1 Shields Ave, Davis, CA 95616
e-mail address: p.m.hoffmann@live.com

This paper presents analysis work on the stability of two-dimensional granular mixtures in a rotating drum setup within the UC Davis summer Research Experiences for Undergraduates (REU) program. A large dataset of avalanche images in a model system of steel bearings in dimer and hexagon shapes was available. The goal was to understand the exact geometric configuration in the mixture causing instabilities that lead to avalanches by using a neural network to identify specific bearing shapes. The work presented here describes a new algorithm to find the centers of the bearing spheres more accurately and to improve previous analysis results.

I. INTRODUCTION

Granular materials, as clusters of macroscopic solid objects, appear everywhere in our daily lives, from foods (like beans, nuts, grains, and their processed grounds), to nature (sand or snow), and even manufactured small parts like ball bearings. Granular materials are important to study because of their many applications. While consisting of single particles, their interactions are fluid-like as granular material can take the shape of its container. A particular area of interest in research is how different grain sizes segregate and how patterns in mixtures are formed. One famous example (termed the 'Brazil Nut Effect') looks at binary granular materials under stimulated vibrational movements. An explanation for the commonly observed vertical size segregation was found by assigning a certain temperature equivalent (of shaking amplitude and frequency) in molecular dynamic simulations by Hong et al. [1]. A second kind of study, looking at granular material under a rotational stimulus, is described in this paper. The experimental setup was described in previous reports [2].

The basic premise was that a large number of 1/8" diameter steel ball bearings were used as grains in a two-dimensional artificial pile. They were confined into a single plane by two sheets of Plexiglas. The drum setup was then rotated at a low frequency to generate avalanches which were recorded using a video camera. The frames immediately before and after each avalanche were extracted from the video and stored as image files. The angle of stability before an avalanche defines the stability of that particular arrangement of grains (note also that angles after the avalanche show a correlation). The figure on the right shows earlier data taken in the lab of a binary mixture of single spheres and hexagons (made by welding together single spheres) [3]. It shows that the addition of the larger hexagon structures increased the overall pile stability. One explanation is that the different sizes lead to a radial segregation, with smaller, single spheres moving to the center of the drum setup. Their larger concentration is responsible for the instability which causes the avalanches. However, in the following paper, the focus is on a slightly different configuration of hexagon (marked green to aid identification) and dimer (vs. single) spheres. The rotating drum of this setup is shown in Figure 2. The following paper describes updates and changes to the image recognition software detailing on detection of sphere centers. Results and improvements are shown and discussed.

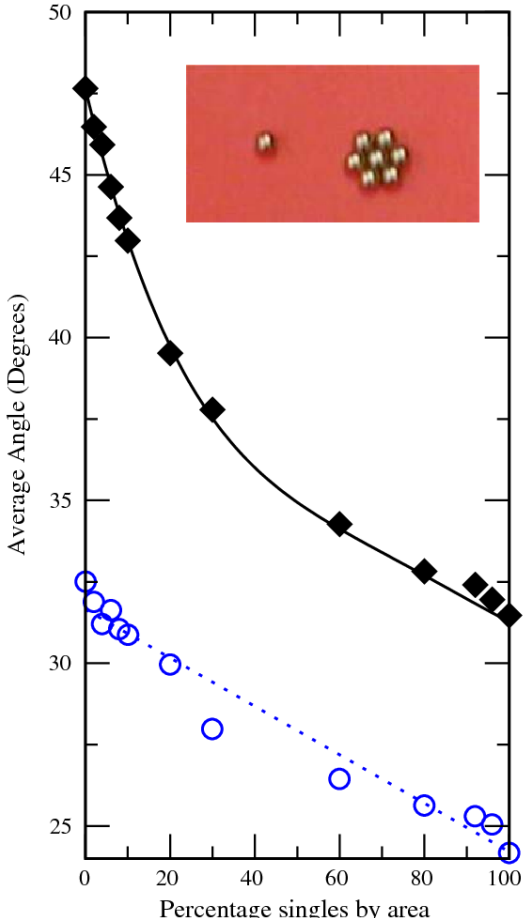

FIG. 1. Hexagons are significantly more stable than single ball bearings. Angles of stability before (black) and after (blue) avalanches are plotted over the percentage of single spheres used, adapted from reference [3].

II. IMAGING SOFTWARE

Image analysis was the central method of this experiment because it showed the detailed distribution of the different sphere shapes at times of the avalanche. The code was written in Python, for which many image processing libraries are available, including OpenCV [4]. OpenCV contains algorithms to perform functions such as tracking, editing, and analysis of images or videos. When applied with machine learning, such as via a neural network, it can cluster and classify data. In our case we trained a neural network [2] on labeled color sets so that it could classify on its own. The reason this approach was taken is shown in the figures below. Figure 2 shows the setup while Figure 3 is a magnified view of the upper spheres. The resolution of the recording camera (at 480x640px) was chosen to accomplish a large field of view with fast image transfer and processing speed. Hence we need to have software able to identify the shapes in a limited resolution image while running the experiment continuously over long periods in real time.

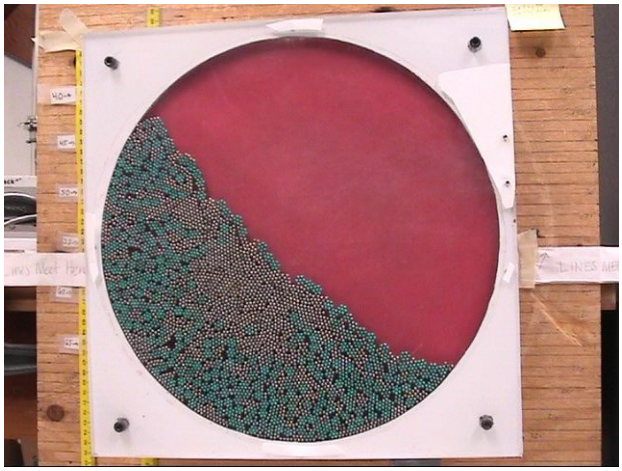

FIG. 2. Image of the experiment of the rotating drum set up used for the data with dimers (silver) and hexagons (green).

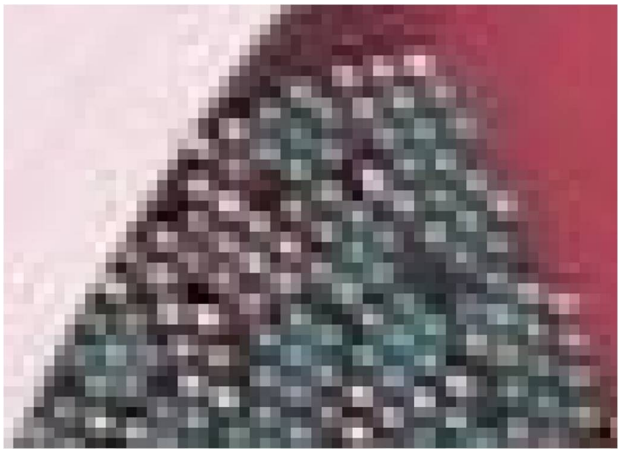

FIG. 3. Zoom-in image to show the resolution of the camera.

III. SOFTWARE SETUP

The program code (developed in previous years [2]) started by importing a single image (from before each avalanche) to be analyzed with OpenCV. It was converted to a RGB scale image, after which the 'Laplacian' [5] was taken on a single color (red) of the image to detect the edges of objects. In a high level summary [2.1], the program was able to detect the border of the drum and mark the angle of the granular pile, after which it segregated the pile from the rest of the drum and started getting a rough estimate of the centers of the spheres by doing two scans. First it looked for the brightest pixels and assumed that these corresponded to the center of a sphere. Those pixel locations were then stored in a text file. Then it looked for the second brightest pixels and appended their locations to the same list. Afterwards the program populated a 3x3 matrix around the bright pixel (note that previously a 5x5 matrix was described [2.1]). Then a center of mass calculation was done to get a more exact center of each sphere. The new centers of spheres were sent to the neural network to find the hexagons in the pile of spheres. It used Delaunay Triangulation to find the first and second nearest neighbors from each center of the spheres by Voronoi partition [6]. This is done by partitioning a Euclidean plane (our two-dimensional image) into convex polygons with the list of centers collected. It does this by using convex bisectors to draw a perpendicular line between two points (a_1, a_2) which are our centers. The circumcenters of Delaunay triangles are the vertices of the Voronoi diagram. Then the neural network labeled spheres ‘green’ and ‘silver’. If it could not decide it labeled it ‘fuzzy’. Lastly, the neural network classified hexagons with all the gathered data. This is pictured in Figure 4.

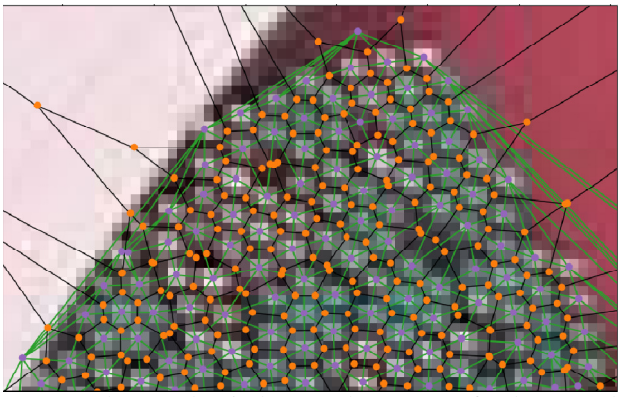

FIG. 4. The purple circles are the center of spheres and the orange are the vertices of the Voronoi diagram.

A significant part of this project involved solving problems that arose in attempting to translate existing functionality to new software and hardware. The program has a long history, having originally been written in IDL before being translated to Python [2.2]. A recent upgrade to a new workstation using an Intel i7 processor caused

compatibility problems. Also, the Python packages that had been used no longer existed because an upgrade of the package to version 3, so the program as a whole did not work. This was similar for the OpenCV package, which is now OpenCV2. For this the syntax needed to be changed. Many lines of code were no longer needed since OpenCV2 compacted calculations. Afterwards the program still did not run completely to expectation, it currently scans only one image correctly, but this was sufficient to optimize the center finding algorithm.

IV. OPTIMIZING THE ALGORITHM

In previous work [e.g. 2.1 and 2.2] some hexagons were skipped during the identification stage. This happened because some of the detected centers were not perfectly seated in the middle of each single sphere. The lengths of the lines drawn by the Delaunay Triangulation technique to connect the centers of the hexagons were too long to classify a hexagon correctly (Figure 5). In the step before the program uses Delaunay Triangulation, it gets a rough estimate of the center locations of each sphere and stores them in a list called 'fauxcenters'. The previous algorithm populated a 3x3 matrix around the fauxcenters and did a center of mass calculation. In our new algorithm, this approach is optimized to give a more refined center of each sphere. In section V the data from the previous and new algorithms are compared.

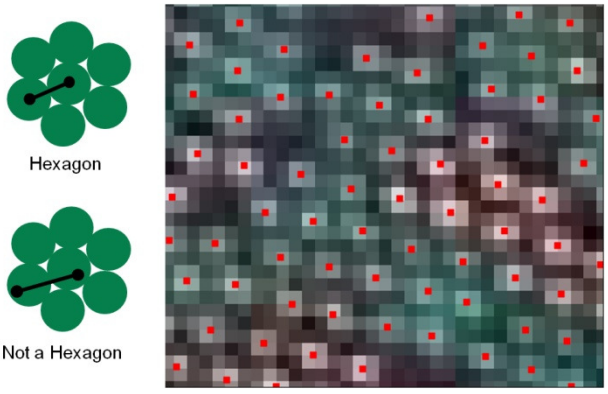

FIG. 5. Example of hexagon identification. The top diagram on the left shows a pattern that is correctly identified as a hexagon, while the bottom is not. The red dots in the image show the centers of each sphere (right). If not properly centered the neural network will have trouble identifying hexagons.

The algorithm is based on a center of mass ($\boldsymbol{R}$) calculation, where the weighted average of the positions r_n is weighted by their corresponding masses m_n.

$$\boldsymbol{R} = \frac{m_1 r_1 + m_2 r_2 + m_3 r_3}{m_1 + m_2 + m_3} \quad (1)$$

The algorithm looks at three pixels vertically and horizontally at a time. Then it looks at each pixel's brightness level by taking the average of the 'RGB' value. This gives our amplitude of brightness 'A'. Looking at equation 1, the m in our case is now treated as 'A'. The algorithm was set to give exact results in three cases for A from left to right (or up to down, respectively) positions indicated by indices from -1 to 1:

1) $A_{-1}=A_0$, 2) $A_{-1}=A_1$, 3) $A_0=A_1$,

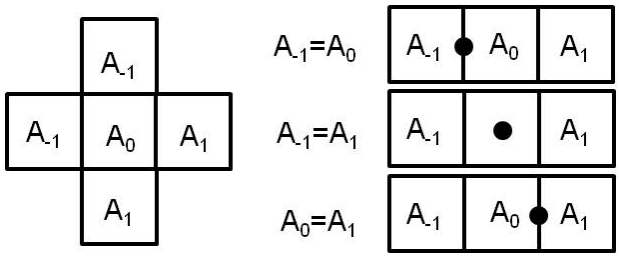

FIG. 6. Overview of the pixel representation, each box represents a pixel with A_0 in the center (left), and the three cases where the new centers equal shifted faux centers (right). Note that the horizontal representation is also true for a vertical arrangement.

In the first case the center is shifted to the left. In the second case, the center is moved in the middle of the middle pixel. Finally, the last case shifts the center to the right. How far the center gets shifted depends on the value of 'A'. The upcoming section shows the derivation of the algorithm for shifts in the horizontal x-direction, but the same will apply for y. The centers should be only shifted in the range between:

$$-\frac{1}{2} \leq \boldsymbol{x} \leq \frac{1}{2} \quad (2)$$

For example, Eq. 1 can be written in this form:

$$\boldsymbol{x} = \frac{-\boldsymbol{\alpha_{-1}}(x)\boldsymbol{A_{-1}} + \boldsymbol{\alpha_1}(x)\boldsymbol{A_1}}{\boldsymbol{\alpha_{-1}}(x)\boldsymbol{A_{-1}} + \boldsymbol{\alpha_0}\boldsymbol{A_0} + \boldsymbol{\alpha_1}(x)\boldsymbol{A_1}} \quad (3)$$

$$\boldsymbol{\alpha_0} = \mathbf{1} \quad (4)$$

$$\boldsymbol{\alpha_1}(\boldsymbol{x}) = \boldsymbol{x} + \frac{1}{2} \quad (5)$$

$$\boldsymbol{\alpha_{-1}}(\boldsymbol{x}) = \frac{1}{2} - \boldsymbol{x} \quad (6)$$

$$\boldsymbol{\alpha_{-1}}(\boldsymbol{x}) + \boldsymbol{\alpha_1}(\boldsymbol{x}) = \mathbf{1} \quad (7)$$

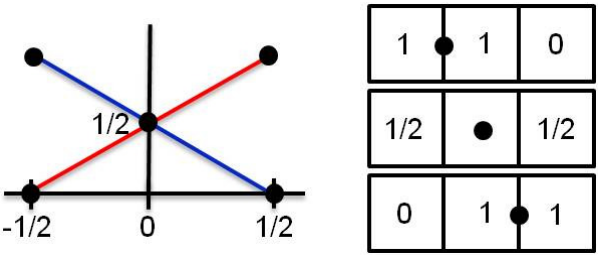

FIG. 7. Linear scale used (left), the red and blue lines represent the positive and negative slopes 'α', respectively. The weight for each box with 'α' values (right).

A parameter alpha (α) is introduced as a weight scale for this algorithm. Values α_1 and α_{-1} use the linear scale (Figure 7) and always need to sum up to one. The weight α_0 for A_0 stays constant and equal to 1 (omitted in the figure center). Thus we get the following form:

$$x = \frac{\left(x+\frac{1}{2}\right)(A_1+A_{-1})-A_{-1}}{\left(\frac{1}{2}-x\right)A_{-1}+A_0+\left(x+\frac{1}{2}\right)A_1} \quad (7)$$

$$(A_1 - A_{-1})x^2 - \left(\frac{A_{-1}}{2} - A_0 + \frac{A_1}{2}\right)x + \left(\frac{A_{-1}}{2} - \frac{A_1}{2}\right) \quad (8)$$

This is the general quadratic formula and can be solved to get the new coordinates. If the quadratic coefficient equals zero no shift will be applied, but if it is non-zero then the positive quadratic equation is applied.

V. DATA SUMMARY

To evaluate the new algorithm, data looking at the processed x and y coordinates (of one image, Figure 2) are plotted below. Shifts are indicated without the 'integer part', only keeping the float value or the numbers past the decimal point. For example, a number of 0.1 would indicate a small shift in one direction, while a 0.9 value would indicate a small shift in the opposite direction. Large shifts would show at numbers around 0.5. It should be noted that perfectly identified centers will appear in a uniform distribution, since the ball centers should have no correlation to pixel locations.

The goal was to obtain a 'flatter' distribution of coordinates than the previous algorithm did. Figures 9 and 10 show the previous and improved algorithms, respectively. The old algorithm (Figure 9) shows a non-even distribution of shifts, with too many small shifts. The new algorithm (Figure 10) improves this but shows now an inverting trend on the distribution compared to the previous one. The peak

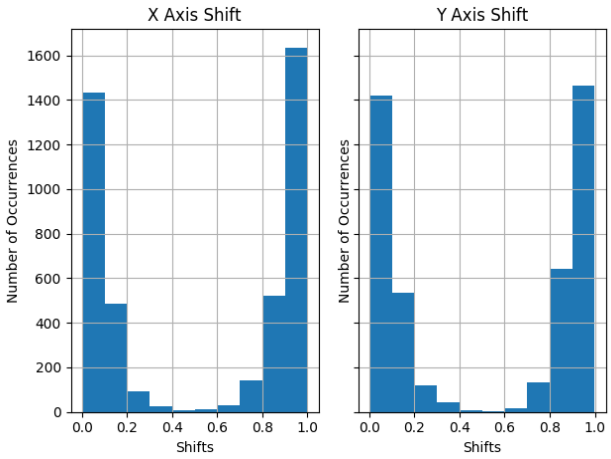

FIG. 9. Histogram of coordinate shifts obtained by the pre-existing algorithm with too many small shifts.

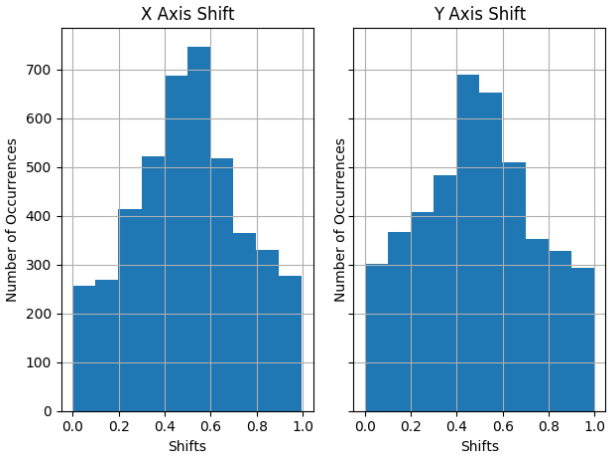

FIG. 10. Histogram of coordinate shifts obtained by the new algorithm. This improved the results but is still uneven because centers are over-represented in the middle.

in the middle means that the center now tends to be displaced too much. This suggests that the less bright pixels around the sides are being weighted too heavily. However, looking at squared values of amplitude 'A' yields the desired result of a flatter distribution (Figure 11). The reason is that 'square' values increase larger numbers more than smaller ones, hence it increases the relative weight of the center pixel and flattens the histogram. This approach also tells us that non-linear scales might work better and leaves the discussion open to try other scaling laws.

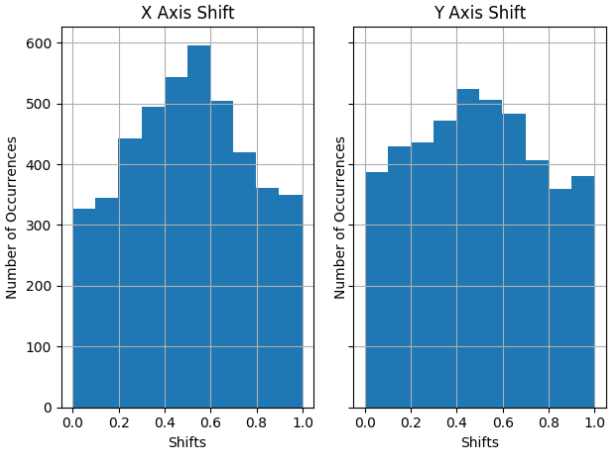

FIG. 11. Histogram of coordinate shifts obtained by the new algorithm and with squared values of the pixel amplitudes. These settings best satisfy the goal of a flatter distribution of shifts.

However, as mentioned above, the evaluation was only performed on a single image. A statistical summary to provide an overview on the percentage of correctly identified shapes (as performed in the previous work by Ortiz [2.1]) is still outstanding.

VI. CONCLUSIONS AND FUTURE WORK

The existing rotating drum setup utilized ball bearings in dimer and hexagon shapes to study their influence on stability in a granular pile. This work demonstrated an improvement in the algorithm to recognize and analyze avalanche images. Within this work the whole code was updated to Python3 and OpenCV2. It works now on a new workstation using an Intel i7 processor. The main focus of the work was on improving the center finding algorithm. The goal was to get a more accurate center of each sphere so that the neural network can find more hexagons during the scan. This was achieved by implementing a new way of shifting the fauxcenters and applying a square weight to the pixel amplitudes. As a result, it provides a flatter distribution of coordinates compared to the previously used algorithm. Following this work, we can plan toward the main goal to present the results of the current setup as shown in Figure 1. Next steps are in reapplying the program to the rest of the existing data and to provide numbers of correctly identified shapes compared to previous works. In the ongoing improvement effort, it would make sense to rewrite/restructure the whole program, utilizing the recent advancements in Python and its new libraries like SciPy, Scikit-learn, and the latest version of OpenCV.

ACKNOWLEDGEMENTS

I want to thank the NSF for providing the opportunity at the UC Davis REU program. I am especially thankful to Dr. Zieve and her lab for the support.

[1] D.C. Hong, P.V. Quinn, and S. Luding, Phys. Rev. Lett. **86**, 3423 (2001)
[2] UC Davis REU program, http://london.ucdavis.edu/~reu retrieved 8/20/2019
[2.1] S. Ortiz, ~/REU18/Papers/ortiz.pdf
[2.2] N. Flowers, ~/REU15/Papers/flowers.pdf
[3] A. G. Swartz, J. B. Kalmbach, J. Olson, and R. J. Zieve, Granul. Matter **11**, 185–191 (2009)
[4] OpenCV, https://opencv.org/
[5] Laplacian Operator, https://docs.opencv.org/2.4/doc/ tutorials/imgproc/imgtrans/laplace_operator/laplace_operator.html
[6] Delaunay Triangulation, http://mathworld.wolfram.com/ DelaunayTriangulation. html