Residual Quantity in Percentage of Factory Machines Using Computer Vision and Mathematical Methods

Seunghyeon Kim[1], Jihoon Ryoo[2], Dongyeob Lee[3], and Youngho Kim[4]

I. INTRODUCTION

Fig. 1. Image of a vibrating hopper [3]

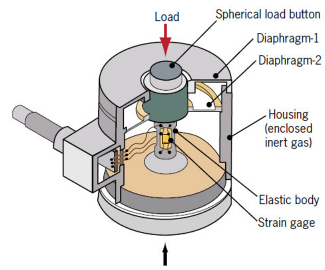

Fig. 2. Simple diagram of a loadcell [6]

Computer vision is an extremely advanced technique in computer sciences that arose a lot of new patents and APIs that could help the world change. Computer vision involves all the analysis that is being done by using the pixels of the images such as machine-learning, CNN, etc. Some manufacturing industries recently found out that they need human resources in order to keep track of whether the machine is working properly or not, and if there are enough materials that are going into the hopper. From an economic perspective, using human resources in order to keep track of things can be an inefficient allocation of resources which could relate to the welfare loss. Not only the welfare loss but using human resources could also hurt them because the hopper is 2m tall; in addition, requires people to go on a ladder to keep things on track. Had the person fallen from the ladder, it could lead to an industrial accident and a significant loss of profit for the company.

Using computer vision, however, could solve the problem. Some of you might think, "Why not use the loadcell specific to the hopper to keep track of the change in mass?" Such solutions could be very useful when the hopper does not vibrate. However, a lot of the manufacturing industries have a vibrating hopper which may not be very helpful to implement loadcells. Then how would computer vision solve the problem? Computer vision algorithm specific to this case uses one camera and a computer to analyze the data received from the camera. They can return how full the hopper is. It does not require any human resources which could minimize the welfare loss and the industrial accident. The precise algorithm that was used would be explained later.

II. Related Works

A. Loadcell

First, a load cell that measures the weight of the materials in a hopper could be used as a solution, yet it is not used because most of the manufacturing industries use vibrating hoppers which can equally distribute the materials inside the hopper and minimize jamming. The vibrating hopper does not only wear out the load cell, but it could also exhibit oscillating motion that does not return a precise weight. Therefore, it is not a very profiting solution to the problem.

B. Machine Learning Techniques

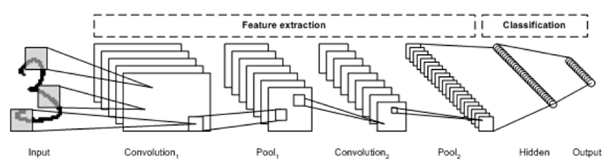

Fig. 3. Simple diagram of how deep learning technique works [2]

Deep learning techniques are very common ways to approach this problem. Deep learning techniques offer very precise analyses of the images, and they are very good-looking with computer vision. Nevertheless, they are not recommended since deep learning techniques have time lags. They do not return the results as quickly would a human eye would do, and it requires more than 100 sample data in order to make deep learning quasi-excellent. They also require a very good Graphics Processing Unit(GPU) in order to offer excellent analyses of data which is not a very profitable way for the owners of the industries.

III. Challenge and Target Scenario

The challenge was to create an algorithm that can efficiently analyze how much the hopper is full in terms of 10%, 25%, 50%, 75%, and 100%. The first way I thought of was to use K means clustering algorithm to simplify the image into 2 colors and analyze it. However, there was a huge outlier that was found when I used the sample data. As a result, I had to come up with a whole new algorithm that could do the work. The new algorithm uses OpenCV, standard deviation, and variance.

A. Theory

The theory of the algorithm is based on the fact that more materials in the hopper affect the standard deviation of the target line from the image(target lines will be explained in the upcoming subsections).

Fig. 4. Sample Data of 10% Full Hopper

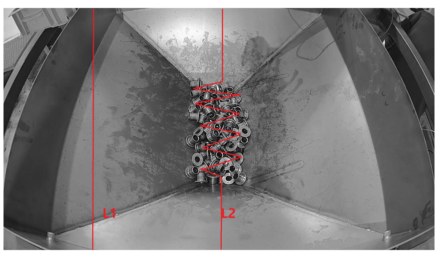

Fig. 5. Modified Sample Data of 10% Full Hopper

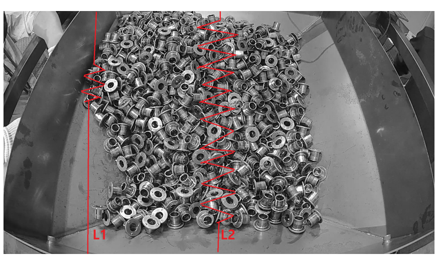

Fig. 6. Modified Sample Data of 50% Full Hopper

$$\sigma = \sqrt{\frac{\sum(r_i - r_{avg})^2}{N}} \tag{1}$$

Where σ = standard deviation, r_i = value of each pixel, r_{avg} = average pixel value, and N = size of the pixel array.

$$\sigma^2 = \frac{\sum(r_i - r_{avg})^2}{N} \tag{2}$$

Where σ^2 = variance and the rest of the variable are the same as formula number 1.

$$A_1 = \frac{\sigma_1 + \sigma_2}{2} \tag{3}$$

Where σ_1 = standard deviation of pixels in L1, σ_2 = standard deviation of pixels in L2, and A_1 = arithmetic mean of the standard deviation of L1 and L2.

$$A_2 = \frac{\sigma_1^2 + \sigma_2^2}{2} \tag{4}$$

Where σ_1^2 = variance of pixels in L1, σ_2^2 = variance of pixels in L2, and A_2 = arithmetic mean of the variance of L1 and L2.

Figure 4 is the sample data of a 10% filled hopper, and figure 5 depicts the two abstract targeted lines that will be used for this algorithm. According to figure 5, the image is converted into a greyscale image from an RGB image, and there are 2 lines: "L1" and "L2." L2 is vertically located at the center of the image so that L2 could determine the standard deviation at the center, and determine how full the hopper is. L1 is another line to determine whether the hopper is 75% full or 100% full(the materials will be reached to L1 if it overflows 75%).

The algorithm retrieves the values of the pixels that are located in the lines. As the values are retrieved, the code calculates standard deviation through formula number 1. According to figure 5, the standard deviation of L2 seems more affected in terms of standard deviation by the texture of the materials inside the hopper than that of L2 in figure 6. The length of the zigzagged lines inside the figures depicts the chunk of variation created by the materials inside the hopper to the standard deviation. Were the hopper filled, even more, the chunk of zigzag would have been larger, making the value of standard deviation greater. However, the problem arose when contrasting the standard deviation values. The formula that involves the standard deviation of L1 and L2 should be determined.

Using the arithmetic mean of among variance, standard deviation, or A_1^2(A_1 in formula number 3) was proven effective for contrasting with accuracy over 85%(will be explained at the choice of algorithm section). They each had their pros and cons. For example, the mean of variance showed 4-digit magnitudes that can be used to compare the values between each type of hoppers. The squared arithmetic mean of standard deviation also exhibits 4-digit magnitudes that share the same pros as the mean of variance does but with different values. For the arithmetic mean, however, the magnitudes are not large enough(2-digit) to determine the precise threshold between each percentage of hoppers which can be problematic, and it is extremely similar to the squared arithmetic mean. In fact, the larger the magnitude is, the more it can allow a range of errors. Therefore, the choice of the algorithms was narrowed down to the mean of variance and the squared value of the arithmetic mean of standard deviation.

B. Experimental Results of Algorithms

According to figure 7, the accuracy of the algorithms tended to be more than 85%, and the average time taken for the code to analyze the image was less than a second which is an overwhelming result compared to 1.8s of run

```
PS C:\Users\[redacted]
0.04648048981376316
0.9130434782608695
PS C:\Users\[redacted]
0.0438040132107942
0.8695652173913043
```

Fig. 7. The Accuracy of the Algorithm

time and barely over 80% accuracy for existing deep learning techniques in the common smart factory industry. The problem, however, is that whether 0.003s of run time is worth 5% of accuracy compared to each algorithm mentioned. This may not be a debatable question when only seeing the 5% value, but the fact is that only 23 data were given. 1/23 is approximately 4.3% which is related to the difference in accuracy that the accuracy of the algorithm differs only by 1 test case. As I mentioned about the vibrating hopper in the first section, the given test case images were taken at extreme environments(such as figure 6 where the materials are skewed to one side). Therefore, the mentioned algorithms may exhibit high accuracy in the real data set. The specific explanations will be included in the evaluation section.

IV. Evaluation

The overall algorithm seems successful until now. Nevertheless, the process of which I took to prove the effectiveness of the algorithm must be explained.

A. Run Time Measurement and Accuracy Measurement

```
init = time.time()
LP.append(PKMF.PhotoAnalysis2(FoldL+each))
lin = time.time() - init
avgT += lin
```

Fig. 8. Sample code to determine the run-time

From figure 7, the run-time and the accuracy of the algorithm could be seen. The approach to measure the run time was using the "time" library in python. It is a general approach to determine the run-time of the code, but I needed to determine the run-time for 23 data at once. Thus, I used the "time" library in order to determine the run-time of each and every 23 test data and averaged them which is exhibited in figure 7. The accuracy also considers all the outliers(it could be considered as an inaccurate output had the algorithm return 50% instead of 75% when the actual image is 75%) as wrong outputs, so it only returns

$$R/T = accuracy$$

where

$$R = \text{Correct number of outputs}$$

and

$$T = \text{Total number of outputs.}$$

Figure 8 shows the way how it was coded in order to determine the run-time for the code. As an example, a company called FronTech Inc. located in the Republic of Korea runs 50 hoppers of the machines in total. Had the algorithm been based on deep learning techniques, a server computer would take a total of 60 seconds to process the images from 50 cameras(1.2s for an image to be processed * 50 = 60s). In real industry, 1 minute will let a worker check 9 10 hoppers in the case of FronTech Inc. However, the suggested algorithm takes only 2.3 seconds to process 50 data from the cameras(0.046 for an image to be processed * 50 = 2.3s). Indeed, the company would require at least 5 GPU-installed computers to process in each camera with deep learning techniques, but the suggested algorithm requires only 1 for 50 which is economically and technically efficient.

B. Test Data Sets and Definitions

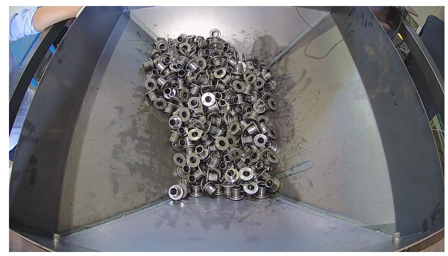

Fig. 9. Test Data of 25% Full Hopper

Fig. 10. Test Data of 25% Full Hopper

The test data cases provided to verify the algorithm were only 23 in total. In addition, most of the data sets were recorded in extreme cases. As an example, figure 9 shows skewed material on the left top corner which is biased data and highly unlikely for a vibrating hopper. Figure 10 shows biased data where the materials are skewed towards the left bottom. Had the hopper been vibrating, the materials are supposed to be closely equally distributed due to the gravitational law.

Fig. 11. Test Data of 10% Full Hopper

Fig. 12. Test Data of 100% Full Hopper

Some data, on the other hand, were not proper data. Figure 11 shows a 10% filled hopper, but the worker's arm is shown which would not return a proper output. Figure 12 also shows data where the picture was taken while the objects were falling. As such, 2 data out of 25 data were improper data that had to be removed.

Fig. 13. Test Data of 75% Full Hopper

Were the data to be correct and there were at least 50 of them, it would be enough to validate the algorithm. In the current situation, however, there is no alternative way to validate the data, so, though the data is biased, using the given data would be the only thing I could do to validate my algorithm. Figure 13 was one of the data that the algorithm could not correctly identify. FronTech Inc. stated that the data exhibits 75% filled hopper, but the algorithm identifies 50%. Even with human eyes, people would identify the data as a 50% filled hopper. They are stacked on top of each other; as the hopper vibrates, however, they should be distributed equally.

Not only the data set, but the way how FronTech Inc. defined 10%, 25%, 50%, 75%, and 100% was abstract. Surely they have a general tendency to have patterns, but not all the data could be said in exact percentages. In fact, the data such as figure 13 may be problematic because when I decide the range of standard deviation for each percentage, it may force me to make the range more incorrect. Or else, it will be an outlier.

V. Future Work & Discussion

In proceeding, the techniques mentioned can be used in hoppers in the various smart factory industries. Surely there are a few things that have to be manipulated in order to keep track of the materials correctly such as the coordinates of the L1 and L2 line axes. Had I been given more time, I would like to upgrade the algorithm to identify the coordinates of L1 and L2 automatically from the image of hoppers, increase the number of lines, and make an algorithm that can count the number of specific materials in the hopper.

You might be wondering "what a simple and useless algorithm?" The suggested algorithm is as simple as '1+1.' Nonetheless, such algorithms are not being researched upon because of the fact that the domestic firms in most of the countries demand deep learning techniques and luxury algorithms that could propaganda their companies. In my opinion, it is a waste of welfare and an inefficient solution for a problem like this. In fact, sometimes, people must think simpler to solve the problem.

VI. Conclusion

Overall, it was a cumbersome process in creating the algorithm and correcting the data. Though there were limitations while validating the algorithm, the algorithm itself still exhibits high accuracy and seems ready to be deployed in the real situation. In creating the algorithm, there were a lot of trials and errors that I faced. It was a tough decision to choose which method to use between the arithmetic mean of variance and square of average standard deviation between L1 and L2. In deciding which algorithm to sacrifice, I decided to include both in the GitHub link[4]. The first time I came up with this idea, I thought "Could the solution be that simple?" Despite the challenges I faced, I felt that this is my valuable idea, and it should never be underestimated. I feel like my algorithm may have more potential in the computer vision field, and it could become an algorithm that stands as an alternative for deep learning techniques. It surely needs more work, and there are some outliers due to the fact that it relies on the texture of the materials. On the other hand, it could be applied to solutions to many other problems.

References

[1] Amilkanthawar, Vivek. "Choosing a Deep Learning Framework." Medium, The HumAIn Blog, 28 Feb. 2019, https://medium.com/in-pursuit-of-artificial-intelligence/choosing-a-deep-learning-framework-5669a85ebc3f.

[2] "CNN, Convolutional Neural Network abstract." Blog main page, 4 Jan. 2018, http://taewan.kim/post/cnn/.

[3] "Home." Magnetic Food Feeding Vibratory Hopper - Buy Magnetic Vibratory Feeder,Food Feeding Conveyor,Food Industry Vibrating Feeder Product on Alibaba.com, https://www.alibaba.com/product-detail/Magnetic-food-feeding-vibratory-hopper_60488056903.html.

[4] Kim, Seunghyeon. "Residual Quantity in Percentage of Factory Machines Using Computer Vision and Mathematical Methods." GitHub, 1 Nov. 2021, https://github.com/seunghk1206/Residual-Quantity-in-Percentage-of-Factory-Machines-Using-Computer-Vision-and-Mathematical-Methods/settings.

[5] "Standard Deviation." Corporate Finance Institute, 31 Jan. 2020, https://corporatefinanceinstitute.com/resources/knowledge/standard-deviation/.

[6] WhitePaper. "Load Cell vs. Force Sensor." Tekscan, 5 June 2020, https://www.tekscan.com/resources/whitepaper/load-cell-vs-force-sensor.