Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 22(3): 249-255, Sep. 2024

Regular paper

Simulator-Driven Sieving Data Generation for Aggregate Image Analysis

DaeHan Ahn[1]*, ***Member, KIICE***

[1]Department of Electrical, Electronic, and Computer Engineering, University of Ulsan 44610, Republic of Korea

Abstract

Advancements in deep learning have enhanced vision-based aggregate analysis. However, further development and studies have encountered challenges, particularly in acquiring large-scale datasets. Data collection is costly and time-consuming, posing a significant challenge in acquiring large datasets required for training neural networks. To address this issue, this study introduces a simulation that efficiently generates the necessary data and labels for training neural networks. We utilized a genetic algorithm (GA) to create optimized lists of aggregates based on the specified values of weight and particle size distribution for the aggregate sample. This enabled sample data collection without conducting sieving tests. Our evaluation of the proposed simulation and GA methodology revealed errors of 1.3% and 2.7 g for aggregate size distribution and weight, respectively. Furthermore, we assessed a segmentation model trained with data from the simulation, achieving a promising preliminary F1 score of 78.18 on the actual aggregate image.

Index Terms: Aggregate Analysis, Data Generation, Aggregate Detection, Aggregate Segmentation, Genetic Algorithm

I. INTRODUCTION

The emergence of deep learning has precipitated rapid advancements in the various industries over a short period. This evolution extended its benefits to architecture and led to the development of various innovative applications. In particular, the aggregate image analysis significantly enhanced the quality assessment of cement and concrete. This technology facilitates a direct approach to image analysis using a camera, thereby eliminating the need for manual measurements [1-4]. Despite these advancements, deep-learning-based applications in aggregate inspection face challenges in accelerating research and service development, primarily because of the necessity of compiling large-scale datasets for training.

The training data essential for conducting aggregate image analysis demands aggregate images and their specific attributes, such as particle size distribution (PSD) and weight, as both input and output for the learning models. Additionally, segmentation models that require meticulous labeling efforts to segregate each aggregate gravel individually can be employed for a detailed analysis of aggregates. To achieve this, the conventional data collection method is handled in the following sequence: first, sample preparation and raw image capture; second, aggregate analysis via an actual test; and finally, recording and storing the analysis results. Large-scale data collection faces time and cost challenges, as the entire process—from preparing samples to recording results—is labor-intensive, time-consuming, and costly at every stage. Additionally, for segmentation, manual labor is required to categorize images captured by humans using a direct masking technique, which also significantly increases both time and cost. The second issue is the diversity of the aggregate labels or attributes. Conventional methods do not support the pre-assignment of labels to samples and only permit

Received 8 May 2024, **Revised** 31 May 2024, **Accepted** 31 May 2024
***Corresponding Author** DaeHan Ahn (E-mail: daehani4@gmail.com)
Department of Electrical, Electronic, and Computer Engineering, University of Ulsan, 44610, Republic of Korea

Open Access **https://doi.org/10.56977/jicce.2024.22.3.249** print ISSN: 2234-8255 online ISSN: 2234-8883

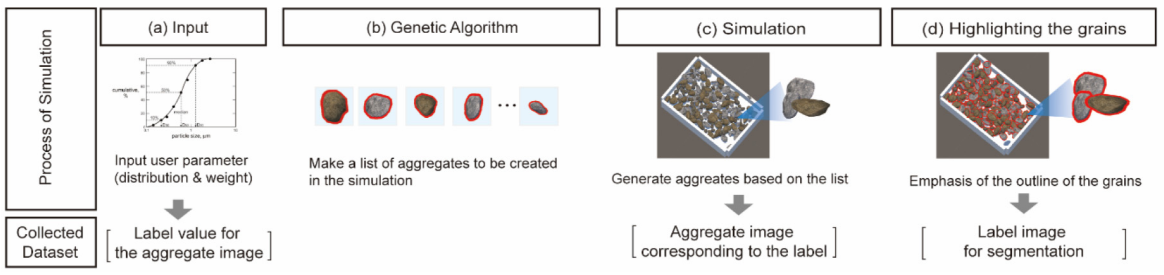

Process of Simulation
(a) Input
Input user parameter (distribution & weight)
(b) Genetic Algorithm
Make a list of aggregates to be created in the simulation
(c) Simulation
Generate aggregates based on the list
(d) Highlighting the grains
Emphasis of the outline of the grains
Collected Dataset
Label value for the aggregate image
Aggregate image corresponding to the label
Label image for segmentation

Fig. 1. The processes of simulation-driven aggregate data generation and the collected datasets.

random sample collection, making it difficult to gather data with a diverse range of labels and produce the desired labels. In particular, aggregate samples collected from the same location tend to have similar labels, making imbalanced labels unavoidable.

To address these challenges, we introduce a simulation-driven approach for collecting a large-scale dataset, as illustrated in Fig. 1, which is designed to emulate aggregate images with characteristics that correspond to user-defined labels in the simulation, collecting three key elements: label, aggregate images corresponding to the label, and aggregate images for the segmentation task. Consequently, this method enables the automated generation of significant volumes of aggregate data and does not require manual intervention.

Moreover, to capture an aggregate image corresponding to the desired label, we propose an aggregate generation model using a genetic algorithm (GA) [5]. For instance, when the model receives user-defined labels regarding the PSD and total weight of the aggregate, the GA model generates a list of aggregates that align with the aggregate's specific labels to be produced. Subsequently, the simulation uses the list to emulate the aggregate, as guided by the list. This approach facilitates the generation of a diverse aggregate dataset featuring the desired characteristics. Additionally, to acquire a ground-truth image for segmentation, we designed an outline shader to emphasize the edge of the aggregate grain. This process enables the extraction of edge information for segmentation, thereby eliminating the need for human effort.

This study proposes a simulation-based approach for collecting large-scale datasets. The approach was designed to emulate the characteristics of a set of images according to custom labels, collecting labels, corresponding set images, and setting images for segmentation tasks. This method allows for the automatic generation of large-scale datasets without manual intervention.

In summary, this study makes three primary contributions: 1) we propose a simulation-based large-scale dataset collection method for vision-based aggregate analysis; 2) we introduce an aggregate-set generation model using a genetic algorithm; and 3) we introduce a ground-truth image generation method for segmentation tasks.

II. BACKGROUND

A. Sieve Analysis of Aggregate

The PSD is crucial in determining the physical properties of concrete mixtures, such as their strength, durability, and stability, which are crucial for the construction industry. Achieving the appropriate PSD is essential for ensuring the desired strength and workability of concrete and asphalt compositions [6,7]. Sieving is one of the most common methods used to assess PSD [8,9,10]. This process involves passing the aggregate through a series of sieves that have progressively smaller mesh sizes, as illustrated in Fig. 2 The amount of aggregate that each sieve retains is weighed, and from these measurements, a PSD curve is generated. This curve provides a detailed analysis of the aggregate granularity, allowing engineers to accurately derive the composition of aggregates. Based on this distribution, predictions can be made regarding the strength and durability of the aggregates. Understanding these properties is fundamental to optimizing the quality and enhancing the performance and longevity of construction materials.

B. Vision-based Particle Size Distribution Analysis

Conducting a traditional sieve analysis, a method that necessitates drying, washing, and measuring as per standards [9,10], is a resource-intensive task that is costly and time-consuming. The advent of modern computer vision tech-

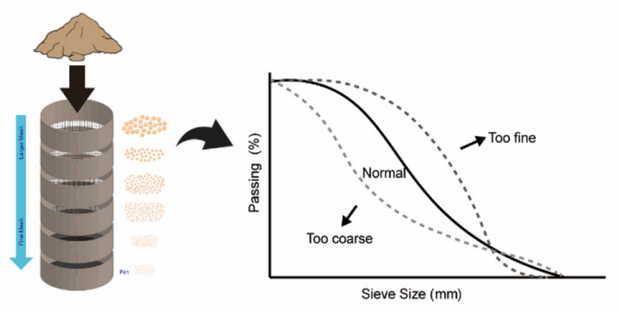


Fig. 2. Particle size distribution analysis by sieving.

niques has transformed aggregate analysis by providing real-time testing capabilities and significantly reducing manual effort, duration, and cost, thereby offering a more efficient and precise method for evaluating PSD.

An application based on computer vision for aggregate analysis was developed by analyzing 2D images captured by cameras. In previous studies [1-4], models utilizing convolutional layers to extract various features from aggregate images were proposed. This model presents an approach for inferring the characteristics of aggregates by mapping the overall characteristics of aggregate images to the actual distribution values of the aggregates. To enhance the performance of aggregate analysis, models incorporating segmentation techniques such as the Canny and Laplacian filters [11,12] have been proposed, which separate individual aggregate gravels in the image and analyze them based on a combination of their characteristics. Recently, there has been a trend toward improved performance using deep-learning-based segmentation models [13,14].

Despite these advancements, computer vision-based solutions for aggregate analysis remain limited, primarily because of the challenge of the large-scale dataset collection required for training deep neural networks. The conventional data collection process requires manual effort for edge delineation, culminating in considerable time and cost expenditure. Furthermore, the inability to predefine data labels renders the collection of diverse datasets more challenging. To address these obstacles, our primary objective is to establish a simulation environment capable of generating large-scale datasets with diverse labels. This facilitates the training of deep-learning-based segmentation models, overcoming the limitations of current data collection and preparation methods.

III. AGGEGATE DATA GENERATION

A. System Overview for Aggregate Data Generator

For a vision-based aggregate analysis, the dataset must contain both an image of the aggregate sample and its corresponding test result, which is referred to as a label. Additionally, for segmentation, an image that delineates the highlighted outlines is essential. Considering these prerequisites, we propose a simulation-based approach to generate an aggregate dataset, as illustrated in Fig. 1. By leveraging a virtual engine to mirror the real world, this approach facilitates the rapid and accurate acquisition of diverse data, surpassing the efficacy of manual testing conducted by humans.

Following prior studies [1,2,3], we designed the simulation environment to include a plate measuring 35 cm × 30 cm × 10 cm, positioned under a camera at a height of 50 cm, specifically designed to capture aggregate images. The design of the aggregates generated in the simulation environment is referenced in [15]. Once the desired particle size distribution and total weight of the aggregate are input into the proposed simulation, as shown in Fig. 1(a), the GA model generates a list of aggregates to be created within the simulation, as shown in Fig. 1(b). The simulation then sequentially generates aggregates in the virtual environment according to this list. After the aggregate generation is complete, the pair of generated aggregate images captured by the virtual camera and the input characteristics for the aggregate are collected. These processes are illustrated in Fig. 1(c). Additionally, an outline shader was applied to the generated aggregate images to produce images with emphasized outlines, which were stored as labels for segmentation tasks, as shown in Fig. 1(d).

B. Data Generation Using Genetic Algorithm

The simulation aims to produce aggregates to obtain data that adhere to the desired criteria, such as the distribution and weight value. Hence, merely generating aggregates at random is insufficient, and an optimization process is essential. To accomplish this, we implemented a genetic algorithm as illustrated in Fig. 3. The overall processes reflect the basic GA, which consists of initialization, evaluation, selection, crossover, and mutation. We customized the algorithm to address our specific problem and organized it as follows. Initially, we introduced each individual, also known as a chromosome, to represent a possible solution, with further details discussed in Section III.B.1. Second, we developed a fitness function to evaluate and select a well-defined chromosome for the next generation, as detailed in Section III.B.2. Finally, we implemented a crossover strategy that addressed the overweight and underweight issues encountered during the crossover. for which solutions are presented in Section III.B.3.

1) Chromosome Generation for Genetic Algorithm

A chromosome, which was the target to be optimized by the GA model, was designed to represent a predefined list of aggregates created in the simulation. Consequently, a gene in the chromosome represents gravel and contains characteristics, such as shape, size, volume, and intended placement, to be created in the simulation.

Fig. 4 depicts the chromosome generation process. Initially, a gene that symbolizes aggregate gravel in the simulation is

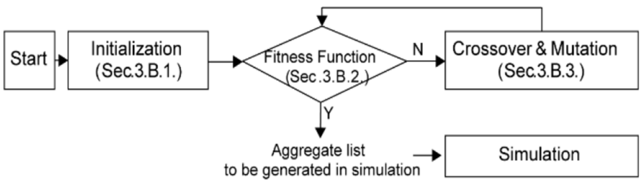

Start
Initialization (Sec.3.B.1.)
Fitness Function (Sec.3.B.2.)
N
Crossover & Mutation (Sec.3.B.3.)
Y
Aggregate list to be generated in simulation
Simulation

Fig. 3. Overall process of genetic algorithm for simulation driven aggregate data generation.

randomly structured as a vector $\mathbb{R}^{1\times d}$, encompassing the essential attributes of the aggregate, such as grain type, size, weight, and location, to be emulated in the simulation. The newly formed genes were appended to the last chromosome index. This cycle is repeated until the cumulative weight of the chromosome aligns with the desired weight criterion predetermined by the user. Consequently, all chromosomes were optimized to the desired weight values during the initialization process. The comprehensive steps involved in chromosome generation are described in Algorithm 1.

2) Fitness Function

The fitness function is designed to produce aggregate lists that satisfy both the desired weight and particle size distribution curve criteria. To achieve this goal, we introduced Eq. (1) to serve as the fitness function.

$$\text{Fitness} = (1 - \boldsymbol{W})^2 + 1/\text{n}\Sigma_{i=1}^{n}(1 - \boldsymbol{D}_n)^2 \tag{1}$$

The equation comprises two components: an evaluation of the weight condition and an assessment of the particle size distribution curve. For weight condition evaluation, the fitness function calculates the ratio between the total weight W_T of the generated aggregates and the desired weight W_D. The desired weight represents a predetermined parameter, whereas the total weight is determined by aggregating the weight of each gene within a chromosome. The formula is as follows:

$$\boldsymbol{W} = W_T / W_D \tag{2}$$

Thus, as the total weight of the aggregates to be generated in the simulation approximated the desired weight, the resulting value closed 1.0. Based on this fact, the fitness for the total weight can be evaluated by the difference between 1.0 and the ratio W. Conversely, the second component of the proposed Eq. (1) assesses the extent to which the particle size distribution within a chromosome aligns with the specified desired distribution, as calculated using Eq. (3).

$$\boldsymbol{D}_n = d_n(\ddot{d}_n \pm \alpha\gamma) \tag{3}$$

$\boldsymbol{D}_n$ represents the fitness associated with the distribution of aggregate gravel passing through an n-sized sieve, and d_n signifies the percentile of particles passing through the *n*-sized sieve, calculated by aggregating particles within a chromosome. $\ddot{d}_n$ represents the desired distribution value for each sieve. The constant *a* serves as a regularization factor to adjust the strictness of the fitness function, while γ (≤1.0) is another constant that modulates *a* value. During the GA iteration, the value of *a* is updated via $a \leftarrow a\gamma$, indicating a gradual decrease towards zero with each GA cycle. The fitness function for the distribution was formulated by averaging the values for each sieve.

Finally, the outcome of the fitness function will approach

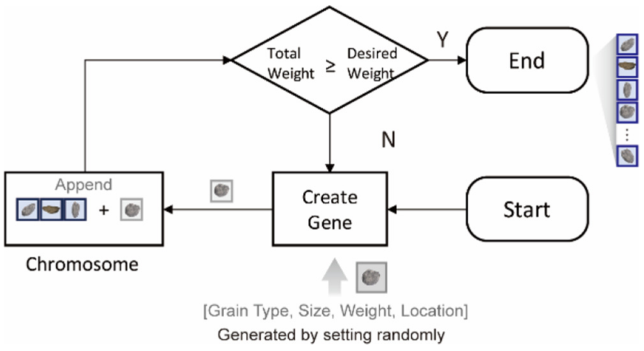

Total Weight ≥ Desired Weight
Y
End
N
Append
+
Create Gene
Start
Chromosome
[Grain Type, Size, Weight, Location]
Generated by setting randomly

Fig. 4. Chromosome generation process.

Algorithm 1. A chromosome generation

```
Input: Desired weight $W_D$
Output: A chromosome for genatic algorithm

Initialize: Total weight $W_T = 0$
Loop $W_T \le W_D$ do
        $Gene_n$ ← Generate a gene ranomly
            [Particle Type, Size(x,y,z), Weight of particle $W_n$, Location(x,y,z)]
        Chromosome ← Append($Gene_n$)
        Total weight of chromosome $W_T$ ← Add($W_n$)
end
return Chromosome
```

0.0 when a chromosome precisely aligns with the desired weight and distribution. Subsequently, using this fitness function, the GA model selects chromosomes in a rank-based manner.

3) Crossover Process

The crossover stage involves creating a new chromosome by combining two or more existing chromosomes. In our GA procedure, this process can introduce variations in weight as the crossover operation is conducted without considering the weight factor and is performed randomly. To address the total weight discrepancy post-crossover, we introduce the post-processing method depicted in Fig. 5, which is detailed in Algorithm 2. This method accounts for two scenarios, in which the resulting chromosome is either over or under the desired weight. In cases of overweight, a reduction was applied to align with the target weight. Because the weight needs to be reduced, genes are deleted individually starting from the last index of the chromosome to decrease the weight. If the weight value reaches the set value, the process is halted, and the result is output as a new chromosome. Conversely, for chromosomes falling under weight, additional genes were incorporated using the chromosome generator described in the Algorithm. 1, until the weight requirement was satisfied. Consequently, this approach guarantees that all chromosomes consistently match the desired weight value throughout the GA process.

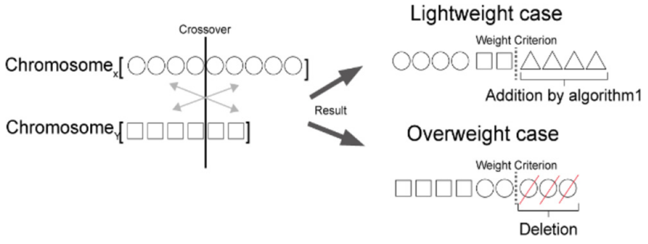


Fig. 5. Crossover strategy for maintaining the weight value.

Algorithm 2. Crossover process

```
Input: Randomly selected two chromosomes ($C_X$, $C_Y$)
Output: New chromosome for next generation ($C_N$)

$\alpha$ is a constant for margin of error
$W$ is a constant for target weight

P ← set crossover point randomly
$C_N$ ← Crossover by concatenate[$I_X$[0:P], $I_Y$[P:end]]
Total weight of new chromosome $W_T$
if $W_T < (W+\alpha)$ then
        While $W_T < (W+\alpha)$ do
                Initialize new gene $G_n$ using Algorithm 1.
                Concatenate [$C_N$; $G_n$]
                Update total weight of chromosome $W_T$
        end
elseif $W_T > (W-\alpha)$ then
        while $W_T < (W-\alpha)$ do
                Delete the last gene in chromosome $C_N$
                Update total weight of chromosome $W_T$
        end
end
return $C_N$
```

C. Aggregate Image for Segmentation

For effective image segmentation, a sufficient number of target images, specifying how the input images should be segmented, must be provided to train the model. However, generating a segmented target image is a laborious, time-, and cost-intensive task. This challenge can be addressed in a simulation by emphasizing the edges of objects, using a technique known as the outline shader. The outline shader is a shading technique used to highlight the outlines of 2D or 3D objects. This method achieves an edge-highlighting effect by rendering a slightly expanded version of the original object's mesh and then rendering the original object on top. The process of creating expanded object $\boldsymbol{O}_e$ can be expressed mathematically as follows:

$$\boldsymbol{O}_e += \text{vertex of } \boldsymbol{O}_e + \alpha \tag{5}$$

By duplicating the original object to create a new object, and obtaining the values of the vertices of the object mesh, an expanded mesh can be obtained by slightly moving each vertex in the direction of the normal vector. Applying a single color to the expanded object and rendering it before the original object causes the two objects to overlap with the expanded object appearing behind the original object. We adopted this method and designed a simulation to easily obtain aggregate images for segmentation labels from the original aggregate images by applying this technique.

IV. EXPERIMENTS

This study aims to generate aggregates in a simulation that meets the user's desired specifications, such as total weight and particle size distribution. To achieve this goal, we present a GA model that creates a list of aggregates to be generated in a simulation while satisfying the user specifications. Therefore, the evaluation of this model involved a comparison between the user requirements and the characteristics of the aggregates produced by the model. Specifically, the model was evaluated based on the accuracy of how the characteristics of aggregates obtained through the model matched the user's requirements. Fig. 6 shows the accuracy obtained while optimizing the list of aggregates to be generated when the requirements are given. Our model consistently meets the required weight value during the generation and crossover processes. Performance exceeds 98.2% from the initial iterations and remains stable with further iterations. The simulation showed a difference of 2.7 g in weight. It means an error of less than 1.0% (≈2.7 g/5 Kg). For particle size distribution, it initially showed considerably low performance because it was randomly set, but as iterations proceeded, it approached the requirement conditions. Figs. 6 and 7 present the comparison results between the user requirements for the ground truth (PSD) and the distribution generated by the model. As shown in the results, there is a high degree of alignment between the two outcomes, with an evaluation showing a performance score of up to 98.7%. These results demonstrate that we can obtain an aggregate image in the simulation that matches the user requirements.

Fig. 8 demonstrates the proposed simulation and application of outline shading for segmentation. Because all the aggregates can be identified and controlled in the simulation, it is possible to accurately emphasize the outline for all the aggregates. Based on this, it is clear that the proposed model can contribute to minimizing human effort and time costs.

The datasets generated by the simulation can help train deep neural networks and are particularly effective for segmentation models. To demonstrate the feasibility of the dataset, we preliminarily evaluate the segmentation model [14], as shown in Fig. 9. The model used in this experiment was trained solely on the generated datasets and not on actual aggregate datasets and was applied to real aggregate data for evaluation. The evaluation metric used was the F1 score, which assessed how well the segmentation lines (i.e., the

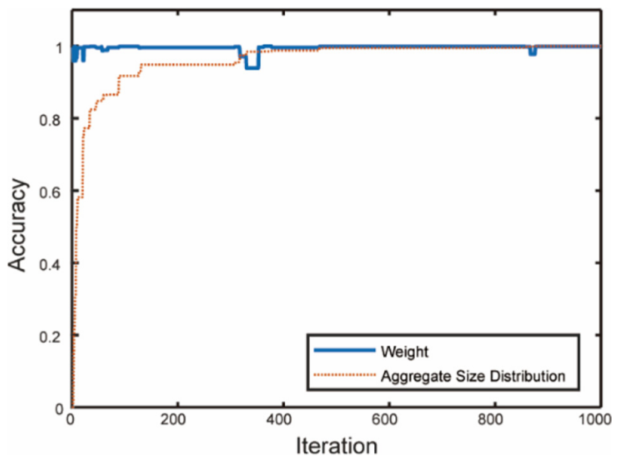


Fig. 6. Evaluation result of aggregate generation of the proposed GA model.

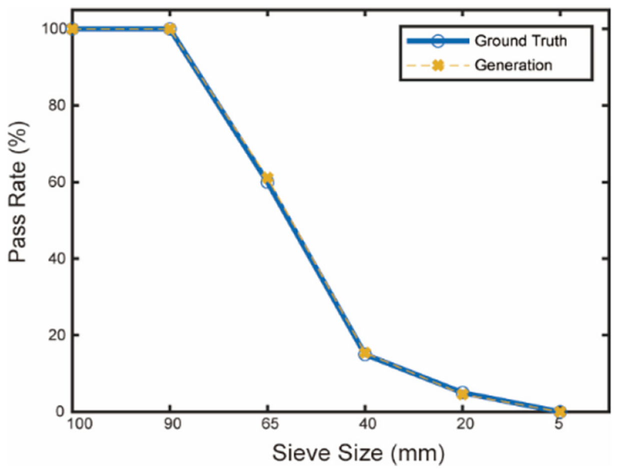


Fig. 7. Comparison between the ground truth and the particle size distribution of a sample generated by the proposed model.

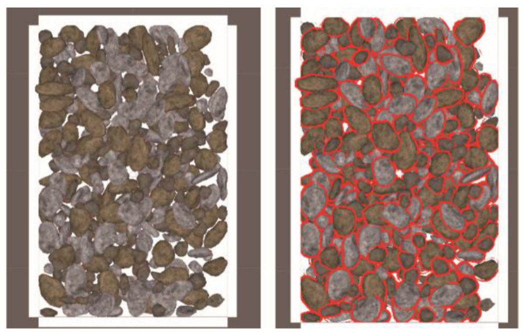
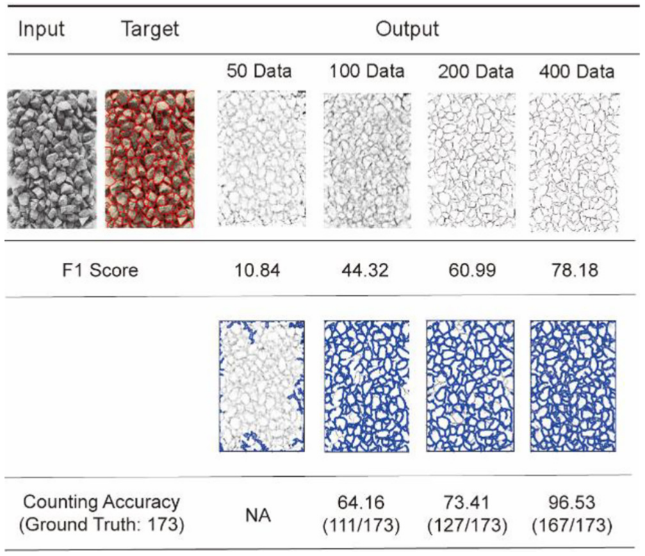


Fig. 8. Demonstration of outline shader for segmentation.



Input	Target	Output			
		50 Data	100 Data	200 Data	400 Data
F1 Score		10.84	44.32	60.99	78.18
Counting Accuracy (Ground Truth: 173)		NA	64.16 (111/173)	73.41 (127/173)	96.53 (167/173)

Fig. 9. Evaluation of a segmentation model trained by the generated samples.

outlines of the objects) were created. The results showed that the performance improved as more virtually generated data were used, achieving a maximum F1 score of 78.18. Additionally, the accuracy of the aggregate recognition based on the segmentation results was evaluated to determine how well the aggregates could be recognized. This evaluation measures the number of aggregates that can be recognized using an object-recognition algorithm [16,17]. The evaluation results proved that, at an F1 score of 78.18, the object recognition accuracy was 96.53%, demonstrating that nearly all objects could be recognized. This evaluation sufficiently showed the possibility of analyzing real aggregates, even when training was conducted using simulated aggregates.

In this discussion, we provide a simulation for collecting aggregate analysis data based on image processing. However, additional tasks are required to improve the performance. For example, incorporating diverse stone modeling techniques and high-resolution images for meshes can enhance the quality of the dataset. Furthermore, optimizing the neural network performance with the generated dataset requires consideration of variables, such as the location and intensity of the light source, camera noise, and other environmental factors, all of which can be readily simulated. Future evaluations will focus on the impact of texture resolution, lighting conditions, plate size, and other pertinent external factors on the segmentation performance. Advancing the current gravel-generation algorithm to ensure data diversity can significantly boost the effectiveness of the simulations.

VI. CONCLUSIONS

This study introduced an innovative strategy for gathering large-scale datasets for vision-based deep learning models for aggregate analysis. The simulator designed for this purpose efficiently creates aggregates within a virtual setting and instantly provides reference data for segmentation, drastically diminishing the resources and time required for data acquisition. A GA was employed to enhance the precision of the simulated data and ensure that they met the specific weight and particle size distribution criteria for the aggregates. The method has a remarkably low error rate, achieving 2.7 g and 98.7% for the weight value and particle size distribution, respectively. Additionally, the segmentation model was trained and evaluated using data generated by the simulation, resulting in an F1 score of 78.18 and an aggregate detection accuracy of 96.53%, proving that the developed model can adequately analyze actual data.

The results obtained from this investigation offer substantial support for the development and assessment of segmentation models tailored for deep-learning-driven aggregate analyses. Notably, the simulator delineated in this study is invaluable for verifying the segmentation model performance and identifying errors in 2D images, thereby enabling further advancements in segmentation research.

ACKNOWLEDGMENTS

This research was supported by Training of experts in the smart manufacture in 2022 funded by the Korea Industrial Complex Corporation.

REFERENCES

[1] M. Coenen, D. Beyer, C. Heipke, and M. Haist, “Learning to sieve: prediction of grading curves from images of concrete aggregate,” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* Nice, France, pp. 6-11, 2022. DOI: 10.48550/arXiv.2204.03333.

[2] S. K. Kim, W. B. Choi, J. S. Lee, W. G. Lee, G. O. Choi, and Y. S. Bae, “Examination of Aggregate quality using image processing based on deep-learning,” *The Transactions of the Korea Information Processing Society (KTSDE),* vol. 11, no. 6, pp. 255-266, 2022. DOI: 10.3745/KTSDE.2022.11.6.255.

[3] H. W. Park, and T. J. Park, “Virtual coarse aggregate analysis simulation using image segmentation based on deep learning,” Hanyang University Thesis, 2021.

[4] Y. I. Lee, B. H. Kim, and S. J. Cho, “Image-based spalling detection of concrete structures using deep learning,” *Journal of the Korea Concrete Institute,* vol. 30, no. 1, pp. 91-99, 2018. DOI: 10.4334/JKCI.2018.30.1.091.

[5] M. Mitchell, “An introduction to genetic algorithms,” MIT Press, 1998.

[6] L. D. de Oliveira Haddad, R. R. Neves, P. V. de Oliveria, W. J. dos Santos, and A. N. de Carvalho Jr. “Influence of particle shape and size distribution on coating mortar properties,” *Journal of Materials Research and Technology,* vol. 9, No. 4, pp. 9299-9314, 2020. DOI: 10.1016/j.jmrt.2020.06.068.

[7] B. A. Tayeh, M. H. Akeed, S. Qaidi, and B. H. A. Baker, “Influence of sand grain size distribution and supplementary cementitious materials on the compressive strength of ultrahigh-performance concrete,” *Case Studies in Construction Materials*, vol. 17, pp. e01495, 2022. DOI: 10.1016/j.cscm.2022.e01495.

[8] G. W. Gee, and J. W. Bauder, “Particle-size analysis,” *Methods of soil analysis: Part 1 Physical and mineralogical methods,* vol. 5, pp. 383-411, 1986. DOI: 10.2136/sssabookser5.1.2ed.c15.

[9] Standard Test Method for Sieve Analysis of Fine and Coarse Aggregates, 2014, [Online], Available: https://www.kssn.net/search/stddetail.do?itemNo=K001010103324 (Accessed Jun. 06, 2024).

[10] Standard Test Method for Sieve Analysis of Fine and Coarse Aggregates, 2020, [Online], Available: https://www.astm.org/c0136_c0136m-19.html (Accessed Jun. 06, 2024).

[11] P. Dhankhar, and N. Sahu, “A review and research of edge detection techniques for image segmentation,” *International Journal of Computer Science and Mobile Computing,* vol. 2, no. 7, pp. 86-92, 2013. url: https://ijcsmc.com/docs/papers/July2013/V2I7201329.pdf.

[12] P. P. Achajya, R. Das, and D. Choshal, “Study and comparison of different edge detectors for image segmentation,” *Global Journal of Computer Science and Technology,* vol. 12, no. 13, pp. 29-32, 2012. url: https://api.semanticscholar.org/CorpusID:49221102.

[13] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence,* vol. 39, no. 12, pp. 2481-2495, 2017. DOI: 10.1109/TPAMI.2016.2644615.

[14] O. Ronneberger, P. Fisher, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *In Medical image computing and computer-assisted intervention-MICCAI 2015: 18th international conference,* Munich, Germany, Springer International Publishing, pp. 234-241, 2015. DOI: 10.1007/978-3-319-24574-4_28.

[15] Rock package, AssetStore by Unity [Online], Available: https://assetstore.unity.com/packages/3d/props/exterior/rock-package-118182. (Accessed Jun. 06, 2024).

[16] S. Rani, D. Ghai, and S. Kumar, “Object detection and recognition using contour based edge detection and fast R-CNN,” *Multimedia Tools and Applications,* vol. 81, pp. 42183-42207, 2022. DOI: 10.1007/s11042-021-11446-2.

[17] R. Deng, and S. Liu, “Deep structural contour detection,” in Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, pp. 304-312, 2020. DOI: 10.1145/3394171.3413750.

DaeHan Ahn

DaeHan Ahn is an assistant professor at University of Ulsan. He holds a Ph.D in Information and Communication Convergence Engineering from Daegu Gyeongbuk Institute of Science and Technology. His current research interests include industrial AI, Data Science, and Cyber Physical Systems.