Core Segmentation and Fracture Path Detection Using Shadows

Hasan Ozturk and I. Turgut Saricam
Mining Engineering Dept., Middle East Technical University, Ankara, Turkey
Email: ozhasan@metu.edu.tr, turgutsaricam@gmail.com

***Abstract*—Drilling is one of the routine operations carried out in geotechnical projects in order to retrieve samples from the ground. The retrieved samples, i.e. cores, are stored in boxes and analyzed by the geologists and mining engineers to determine several parameters required for rock mass classification systems, such as RMR (Rock Mass Rating), GSI (Geological Strength Index), and Q. For this routine task to be automated, cores should be segmented properly. In this paper, a method is introduced for the segmentation of cores and detection of their fracture paths by using shadows. First of all, three digital true color images of a core box, with the same camera position but different light source positions, are taken using a high resolution camera. After the detection of the core box with color thresholding, the sections of the box are detected by using Hough transform and boundary tracing algorithms. Then, after extracting cores from each row of the box using color thresholding, touching cores are separated from each other with the help of shadows, concave points, and edges. Finally, fracture paths of the cores are detected by taking positions of the light sources into account and tracing the boundaries of the detected shadows. All coding routines are developed in MATLAB 2017a. Two different core boxes with 4 and 5 rows storing HQ and NQ diameter cores having various joint/bedding plane angles are photographed to conduct the study.**

***Index Terms*—core segmentation, fracture path detection, shadows, image processing**

I. Introduction

Taking samples from the ground is one of the tasks carried out before starting a geotechnical project on or in the ground. These samples are generally stored in boxes so that they can be analyzed by geologists and mining engineers later on to determine a number of parameters required for rock mass classification systems, such as RMR (Rock Mass Rating) [1], GSI (Geological Strength Index) [2], and Q [3]. One of the first steps for automating this process is detecting the cores from a digital image properly [4], [5].

In literature, two approaches were used for core segmentation using photography and scanning techniques. Lemy et al. [6] used an edge-detection-based approach. In their study, after detecting the edges with Steger algorithm [7], they followed an edge reconstruction process which calculates a cost for pairs of line segments and connects the line segments presenting the lowest cost to find the breaks. Their algorithm considers the core box full of cores, i.e. no core-free regions (lost or washed-out cores), which might cause including core-free regions as cores. Since the algorithm relies on edges only, it might consider thick fillings as breaks as well. Olson et al. [8] used a 3D non-contact laser digitizer to generate 3D point cloud data of the entire core box. They detected the breaks and location of each core by analyzing the 3D point cloud. They also created a fracture characterization algorithm which finds out whether a fracture is formed due to a mechanical break or not. While using a 3D laser digitizer provides an excellent accuracy in core segmentation, it is an expensive device and the process is slow, tedious, and hard to apply in the field. In this paper, we propose a shadow-based method for segmentation of cores and detection of the fracture paths from digital images of core boxes. The method is cheap and relatively easily applicable in field conditions.

II. Equipment and Setup

The equipment used in this study are a digital camera (Canon EOS 7D Mark II with Canon EF-S 18-135mm IS STM), a wireless remote controller (Canon RC-6), a flash (Canon Speedlite 600EX II-RT), a flash diffuser, a tripod with horizontal arm, a tripod head, a 5-cm-long pink marker, and a few black plastic trash bags. The study is carried out on two core boxes with 4 and 5 rows whose lengths are 100 cm.

III. Methodology

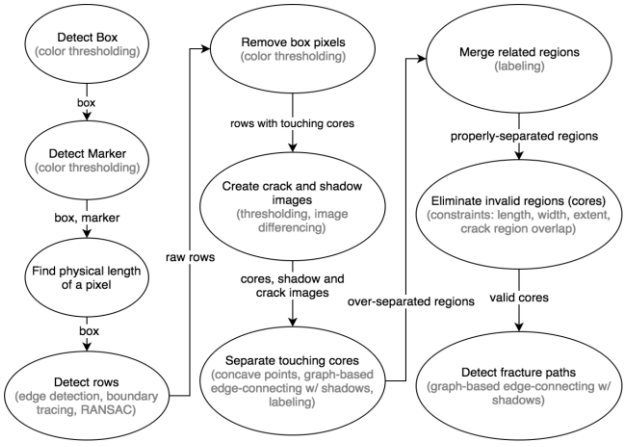

Figure 1. Data flow diagram of the algorithm

Manuscript received February 16, 2018; revised June 20, 2018.

doi: 10.18178/joig.6.1.69-73

The data flow diagram showing the processes of the algorithm can be observed in Fig. 1. These processes are explained in the following parts of this section.

A. Detecting the Rows

To detect the rows, first, strong edges in the box image are found by Canny edge detector [9]. Then, after detecting vertical lines using Hough transform [4], the edges belonging to these lines are traced to find vertical edges. Next, the vertical edges that are horizontally close to each other are connected to create longer vertical edges. Then, these edges are extended using RANSAC [10] so that they divide the image into vertical regions. Finally, the pixels belonging to the core box are removed from each row region by using color thresholding.

B. Creating Crack and Shadow Images

Cracks are actually the regions having too low intensity. The pixels whose intensity is lower than some threshold value are found in all three images of the core box. Then, the found low-intensity regions are combined to create a single mask.

Shadows are found by comparing the shadow images to the shadow-free image. Since the differences between these images are the shadows, a pixel-based change detection approach is adopted. For this purpose, image differencing method [11] is used.

The crack and shadow masks can be observed in Fig. 2.

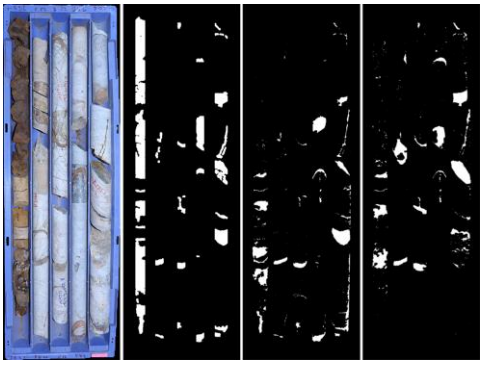

Figure 2. Crack and shadow masks. From left to right: 1) RGB image of the core box, 2) crack mask, 3) left shadow mask, 4) right shadow mask

C. Separating Touching Cores

First, strong edges in gray core box image are found by Canny edge detector [9]. The edges that come from the box itself are removed by applying the core mask to the edge image. Next, a contour connection graph is created.

A *contour connection graph* is a graph which is formed by considering contours (detected edges) as graph nodes and path of the smallest distance between the contours as graph edges.

To create the graph, first, contours in the close vicinity of each contour, then, the smallest distance between the contours that are close to each other are found. Next, a connection is created between the two contours using the two points that create the smallest distance between them. By following this procedure, each contour is virtually connected to the contours in their close vicinity.

As the next step in detecting cores, the boundary of the regions that store the cores is separated into two parts as left and right boundaries. The left boundary of a core region is the part of its boundary that is closer to the row separator on its left. Likewise, the right boundary is the part of its boundary closer to the row separator on its right. The theory is that if the left boundary can be connected to the right boundary by following a short path passing through the detected contours, that path is the path that separates two touching cores. To implement this logic, the left boundary and the right boundary are assigned as sink and source nodes, respectively, in the contour connection graph, and every contour in the graph is virtually connected to both sink and source nodes.

After contour connection graph creation, all shortest paths that connect the left and the right boundaries and that pass through one of the contours overlapping a crack region are found by using Dijkstra's algorithm [12]. The contours forming the shortest paths are connected to each other with a straight line to make sure the paths separate the core region into at least two regions. Note that the final path is not cleaned from possible branches to allow natural separations to happen. Hence, most of the time, the crack region is separated into smaller regions by the branches of the shortest paths as well. These small regions will be merged together in the following parts of this section.

The core regions are further divided into regions by using concave points as well. The theory behind this idea is that when there is a concave point in a core region, it is probably created due to two adjacent cores. This is because a typical core is cylindrical and it does not have concave points on it. The concave points are found by calculating Euclidean distances from the boundary points of the core region to the boundary points of the region's convex hull. Convex hull of a region is the smallest convex region that contains the region.

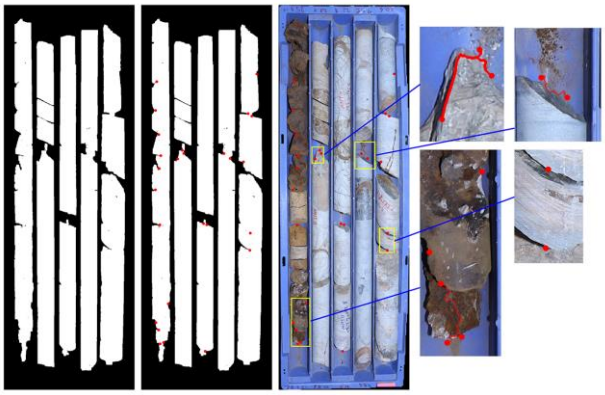

Figure 3. Connected concave points

After the distances from each point of convex hull's boundary to the points of the region's boundary are calculated, the points whose distance to the hull's boundary is greater than a threshold are selected as concave points. Then, the concave points that are in the close vicinity of each other are connected by following low-intensity pixels, if the distance to be travelled is lower than a threshold value. The path created by connecting the two points should divide the region. Hence, two close

points that are on the same side of the core are not connected. Found concave points and connections between them can be observed in Fig. 3. In this figure, from left to right: 1) core mask, 2) core mask with concave points represented by red dots, 3) RGB of the box with concave points. Yellow frames show the locations of the zoomed-in regions. Red lines in the zoomed-in regions are the valid connections between concave points.

The paths created between concave points are used to divide the core mask into smaller regions. Separation results can be seen in Fig. 4. In this figure, the left-most image is the label image showing the regions created after separating touching cores. Each color represents a different region. Regions that have the same color but are not adjacent to each other are different regions. Red dots are the concave points.

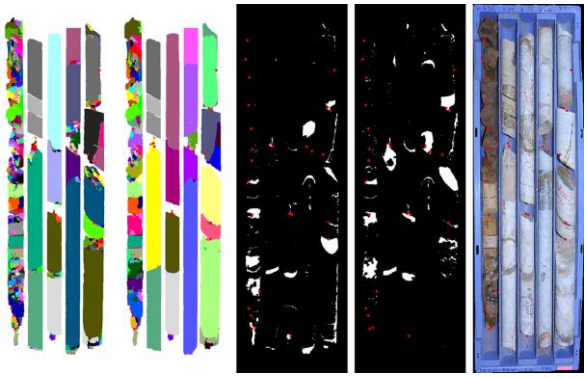

Figure 4. Merged labels. From left to right: 1) labels after separating touching cores, 2) labels after merging related regions, 3) left shadow image, 4) right shadow image, and 5) RGB of the core box.

D. Merging Related Regions

Because of the procedure followed to separate touching cores, there are several small and big regions created after the touching cores are separated. For a more precise segmentation, the regions that are found to be related to each other are merged with each other. Whether two regions are related or not is decided by using concave points and shadows.

First, before doing any merging, the regions that might represent a core are found. To decide this, there are several thresholds defined, which are *minimum length*, *minimum width*, *minimum extent*, and *maximum crack region overlap*. Minimum length threshold is a value that defines the minimum vertical length a region must have so that it can be considered as a core region. Likewise, minimum width threshold defines the smallest width of a core region. Minimum extent constraint looks at the ratio of the number of pixels in the region to the number of pixels in its bounding box. Maximum crack region overlap is the threshold that sets a limit to how much of a core region overlaps a crack region at most.

According to the assumption made about the concave points, a concave point only exists around where two cores touch each other. Therefore, the regions staying between a core region and a concave point are considered as they are related to that core region.

After the regions are merged by using the concave points, another merge operation is performed on the created label image. This time, shadows are used to find related regions. Regions that overlap the same shadow region are considered as related. First, the shadow regions are found in the left and the right shadow masks. Next, areas that the two shadow masks overlap are found. Among these areas, smaller ones are discarded. Then, the remaining shadow areas are used to group the labels of non-core regions. After merging related non-core regions with each other, which core region they belong to is decided by looking at the shadow regions and concave points. When a non-core region is adjacent to a core region or they overlap the same shadow area, and there are no concave points between them, they are merged with each other.

E. Eliminating Invalid Regions

After separation of touching cores and merging of related regions, the regions are subjected to an elimination process. Core regions are found by applying the thresholds described previously. Regions that are not classified as cores are eliminated. Fig. 5 and Fig. 6 show the detected cores. In these figures, from left to right, the first image is RGB of the core box. The second image is the labels decided as core regions. Each color represents a different region. Regions that have the same color but are not adjacent to each other are different regions. The third one shows the cores. Red frames are bounding boxes of each core. An ID is given to each core. The IDs are shown in green.

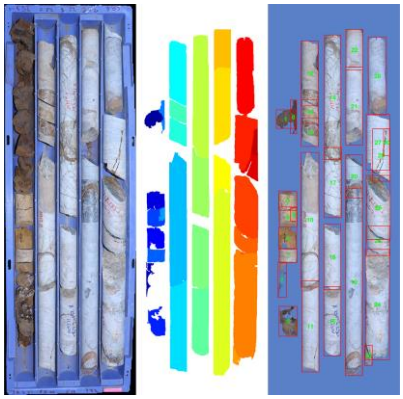

Figure 5. Detected cores of core box 1

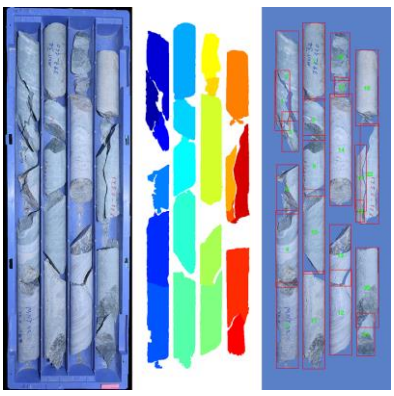

Figure 6. Detected cores of core box 2

F. Detecting Fracture Paths

Fracture paths are the paths that show where the cylindrical part of the core ends and its fractured region starts. The paths are detected using the shadow masks. For each detected core, related shadows are retrieved from the left and the right shadow masks. When a core piece is illuminated from its left side, shadows are created on its right side. The opposite of this statement is also valid. This can be seen in Fig. 7.

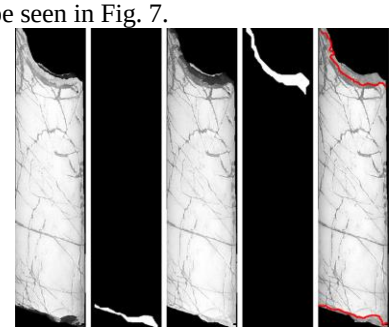

Figure 7. Fracture path detection. From left to right: 1) core illuminated from its left (top), 2) shadows detected in 1, 3) core illuminated from its right (bottom), 4) shadows detected in 3, 5) shadow-free core image. Detected fracture paths are shown in red.

First, edges are detected for the two versions of the core images using Canny edge detector [9]. Then, a contour connection graph is created for each edge mask. Next, for each edge that overlaps the found shadow boundaries, the shortest path that connects the left-most end and the right-most end of the image is found using Dijkstra's algorithm [12]. Finally, the best path is found by looking at how many edges passing through the shadow boundary a path contains. The paths containing the most overlapping edges are selected.

IV. Conclusions

Main conclusions from this study are as follows:

i. The algorithm takes 60 to 90 seconds (with 2.5 GHz Intel Core i7 processor, 16 GB 1600 MHz DDR3 RAM) to complete depending on the number of cores, shadow areas and edges in the image, and available processing power.
ii. Since shadows are utilized, edges created by the patterns on the cores do not cause a misbehavior in separation of touching cores.
iii. Because creation of the shadows requires only a light source, the automation can be achieved at a low cost.

Although the method works well for most of the time, some variables might affect the segmentation negatively and some improvements can be made.

i. The method cannot properly differentiate two cores whose ends fit each other perfectly such that no shadow is created.
ii. Rows with dirt causes a change in core box color. Since the method separates the box from the cores by using color-thresholding, the dirt is considered as core as well. Although the dirty areas will be eliminated because of their size and shape, they might be considered as a part of a core, which causes inaccuracy in the final results.
iii. The light sources causing the creation of shadows should be placed such that the illumination creates enough shadows to properly segment the touching cores as well as to decide where the non-cylindrical parts of the cores are. In addition, it is important that the light is diffused so that it does not create a light burst. Hence, lighting has a vital role for the method to succeed, which requires a controlled lighting environment that might be hard to create.
iv. Cores having deep wears on their cylindrical parts, which cause shadow creation, may mislead the segmentation, if these shadows are not cleared properly.
v. Too dark cores might be considered as cracks because the cracks are detected by thresholding the gray versions of the photographs.

For the algorithm to work properly, the following conditions are recommended to be provided:

i. The core box should be blue. If blue is not an option, then a color which is distinct from any color that can be seen on a core should be selected, and the algorithm should be modified so that it will consider the selected color as the box color.
ii. The color of the core box should be different from the colors of the cores in the core box.
iii. The core box should be free of dirt and dust. It should not contain anything other than the cores themselves.
iv. Before taking the photographs, the cores that are too close to each other such that no shadow can be created between them should be separated from each other to allow shadow creation.
v. The cores should not be marked.
vi. The cores should be cleaned such that no unwanted shadows are created on them.
vii. The cores should not be sprayed with water because it causes unwanted glitters to occur.

For the future work, more core boxes containing cores of various rock types are suggested to be evaluated.

Acknowledgment

The authors would like to thank the Scientific and Technological Research Council of Turkey, TUBITAK, Grant No: 116M692, for financial support.

References

[1] Z. T. Bieniawski, *Engineering Rock Mass Classifications: A Complete Manual for Engineers and Geologists in Mining, Civil, and Petroleum Engineering*, John Wiley & Sons, 1989.
[2] P. Marinos and E. Hoek, "GSI: A geologically friendly tool for rock mass strength estimation," in *Proc. ISRM International Symposium*, International Society for Rock Mechanics, 2000.
[3] N. Barton, R. Lien, and J. Lunde, "Engineering classification of rock masses for the design of tunnel support," *Rock Mechanics*, vol. 6, no. 4, pp. 189-236, 1974.
[4] P. V. C. Hough, "Method and means for recognizing complex patterns," No. US 3069654, 1962.

[5] The MathWorks, "MATLAB and statistics and image processing toolbox release 2017a," Natick, Massachusetts, United States, 2017.
[6] F. Lemy, J. Hadjigeorgiou, P. Cȏté and X. Maldague, "Image analysis of drill core," *Mining Technology*, vol. 110, no. 3, pp. 172-177, 2001.
[7] C. Steger, "An unbiased detector of curvilinear structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 2, pp. 113-125, 1998.
[8] L. Olson, C. Samson, and S. D. McKinnon, "3-D laser imaging of drill core for fracture detection and rock quality designation," *International Journal of Rock Mechanics and Mining Sciences*, vol. 73, pp. 156-164, 2015.
[9] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679-698, 1986.
[10] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.
[11] M. İlsever and C. Unsalan, *Two-Dimensional Change Detection Methods: Remote Sensing Applications*, Springer Science & Business Media, 2012.
[12] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269-271, 1959.

Hasan Ozturk is Assoc. Prof in the department of mining engineering at the Middle East Technical University with an expertise in rock mechanics and geotechnical engineering.

İ. Turgut Saricam is PhD Candidate in the department of mining engineering at the Middle East Technical University with an expertise in rock mechanics and digital image processing.