Machine Vision for Road Pavement Applications Bitumen Coverage and Grain Size Estimation

Källén, Hanna

2013

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Machine Vision for Road Pavement Applications

Bitumen Coverage and Grain Size Estimation

Hanna Källén

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Licentiate Theses in Mathematical Sciences 2013:2
ISSN 1404-028X

ISBN 978-91-7473-742-4 (printed)
ISBN 978-91-7473-743-1 (pdf)
LUTFMA-2036-2013

Printed in Sweden by MediaTryck, Lund 2013

Abstract

In this thesis some research questions regarding durability and quality of roads has been investigated. The questions are analyzed from an image analysis point of view and aims to be a complement to existing methods for analyzing asphalt.

One important factor for the durability of the asphalt layer on roads is the affinity between the stones in the asphalt and the binder that holds the stones together, called bitumen. One step in testing the affinity is to manually estimate the degree of bitumen coverage after the stones covered in bitumen has been washed for a while. The goal with the first two papers is to replace this manual estimation by image analysis methods. The first paper deals with the easier problem where there is a clear color difference between the stones and the bitumen. By using reference images to get information of the typical stone and bitumen color and a graph-cut algorithm we get result that seems to be close to the real degree of bitumen coverage. In the second paper we instead look at the problem with darker stones. In this case we cannot see a clear color difference between the stones and the bitumen. Instead we notice that bitumen and stones reflect light in different ways and take multiple images with lighting from different directions. The degree of bitumen coverage is then estimated by detecting specular reflections in the images.

Another quality control of asphalt is to estimate the size distribution in an asphalt sample and see if it corresponds to the recipe for the asphalt. This is investigated in the third paper, where slices of the asphalt are analyzed. The analysis consists of segmenting the stones individually so that the size of all grains can later be estimated.

Preface

This thesis is based on the following papers:

- Hanna Källén, Anders Heyden, Kalle Åström and Per Lindh, "Measurement of Bitumen Coverage of Stones for Road Building, Based on Digital Image Analysis", *IEEE Workshop on Applications of Computer Vision*, Breckenridge, 2012.
- Hanna Källén, Anders Heyden and Per Lindh, "Measuring Bitumen Coverage of Stones using a Turntable and Specular Reflections", *International Conference on Computer Vision Theory and Applications*, Barcelona, 2013.
- Hanna Källén, Anders Heyden and Per Lindh, "Estimation of Grain Size in Asphalt Samples using Digital Image Analysis", *to be submitted to IEEE Workshop on Applications of Computer Vision*, Steamboat Springs, 2014.

Contents

Introduction

Segmentation of images occurs as subproblems in all the research problems in this thesis. Segmenting an image means to separate the image in different regions where pixels in the same region belong to the same object or are similar in some other way. The most common segmentation is foreground and background segmentation. In this case we only have two regions where the first represent the interesting part of the image, the foreground, and the rest the not so interesting background. This introduction aims to present some of the segmentation techniques available that will be used through this thesis. Also some morphological operations to manipulate the segments will be presented.

1 Morphology

Morphological operations [3] are used to shrink or expand shapes or to smoothen the boundaries of an object. Some of the most common operations are erosion, dilation, opening and closing.

Morphological operations are often used on binary images but could also be extended to gray scale images. Here we will only focus on binary images. All the morphological operations are performed with two sets, one of them is the image and the other one is a small set called structuring element. The structuring element could for example be a square, a cross or a disc.

1.1 Erosion and Dilation

Erosion is used to shrink objects by removing elements at or close to the boundary. It is defined by

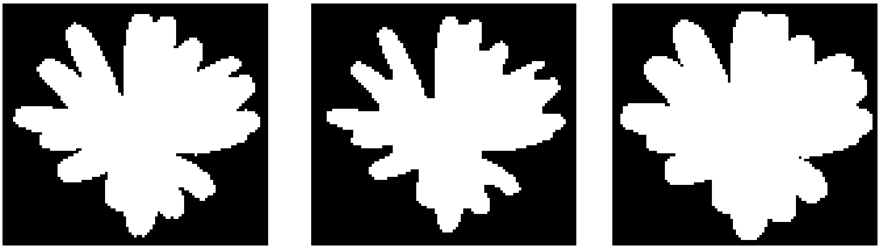
$$A \ominus B = \{x \in X | x + b \in A \text{ for every } b \in B\}, \tag{1}$$

where A is the object and B the structuring element. X is the whole image.

Dilation is used to expand the object by adding elements close to the boundary. Dilation is defined as

$$A \oplus B = \{x \in X | x = a + b \text{ for some } a \in A \text{ and } b \in B\}. \tag{2}$$

Figure 1 shows an example of erosion and dilation on the object seen in Figure 1a. The structuring element used in both the erosion and dilation is in this case is a 3×3 square. The image after erosion can be seen in Figure 1b and Figure 1c shows the image after dilation.



(a) Original image.

(b) After erosion.

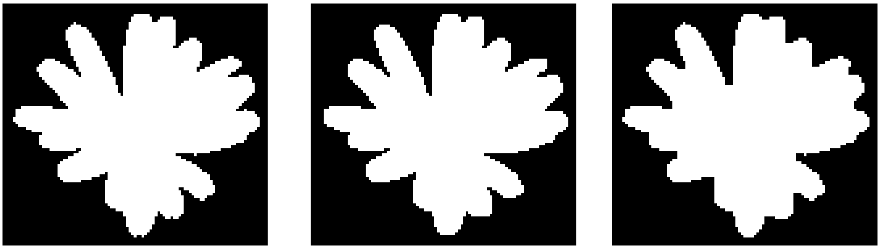
(c) After dilation.

Figure 1: Erosion and dilation performed on the same object.

1.2 Opening and Closing

The opening of a set is defined as erosion with a structuring element followed by dilation with the same structuring element on the resulting object. The closing is the opposite, dilation followed by erosion. Both opening and closing will cause some smoothing of the contour of the object. The opening will remove small out-sticking elements but will not affect deep valleys, which the closing will.

Figure 2 shows opening and closing with the same square structuring element on the image shown in Figure 2a. The opening can be seen in Figure 2b and the closing in Figure 2c.



(a) Original image. (b) After opening. (c) After closing.

Figure 2: Opening and Closing performed on the same object.

2 Thresholding

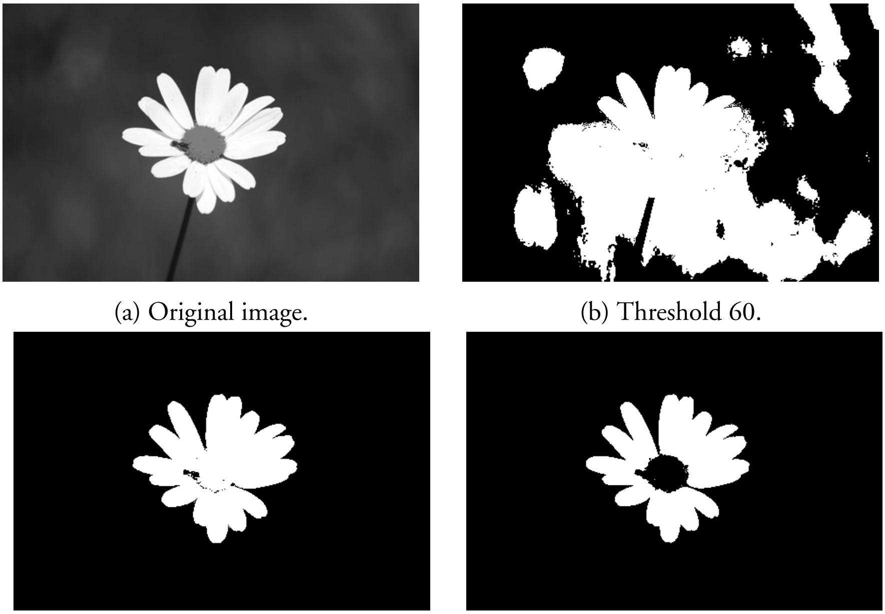
A simple way to segment an image into foreground and background is by thresholding the intensities. Pixels with intensities below the threshold are labeled as background and pixels above the threshold are labeled as foreground or vice versa. Thresholding works well in those cases where there is a clear intensity difference between the foreground and background, but it is very sensitive for which threshold that is chosen.

Figure 3 shows segmentation of an image with three different thresholds. The original image is seen in Figure 3a, the threshold in Figure 3b is too low and does not give any satisfying segmentation of the image. The threshold in Figure 3d is chosen too high while the threshold in Figure 3c is a good choice of threshold.

2.1 Otsu Method

There are some methods that try to find a good threshold automatically. One of the most famous method is called the Otsu method and was described in [5]. The idea behind this method is to minimize the within-class variation where the threshold is separating the pixels in two different classes. The best threshold should then be the one that separates the pixels in a way that the variations within the classes are as small as possible.

First a histogram for the gray levels is computed, this histogram is then normalized and regarded as a probability distribution so that $p_i = n_i/N$, where n_i is the number of pixels with intensity i and N is the total number of pixels. The threshold k separates the pixels into two classes, background pixels, C_0, with intensities lower or equal to k and foreground pixels, C_1, with intensities higher



(a) Original image.

(b) Threshold 60.

(c) Threshold 90.

(d) Threshold 140.

Figure 3: Segmentation of an image with three different thresholds.

than k. The probability that a pixel belongs to the background is computed as

$$\omega_0 = \sum_{i=0}^{k} p_i. \tag{3}$$

In the same way we compute the probability that a pixel belongs to the foreground class

$$\omega_1 = \sum_{i=k+1}^{K} p_i, \tag{4}$$

where K is the maximum intensity.

The within-class variation can then be defined by

$$\sigma_w^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_0^2, \tag{5}$$

where σ_0^2 is the variation of the background pixels and σ_1^2 is the variation of the foreground pixels.

Otsu shows that minimizing this within-class variation is the same as maximizing the between-class variation defined by

$$\sigma_b^2 = \omega_0 \omega_1 \left(\mu_1 - \mu_0\right)^2, \tag{6}$$

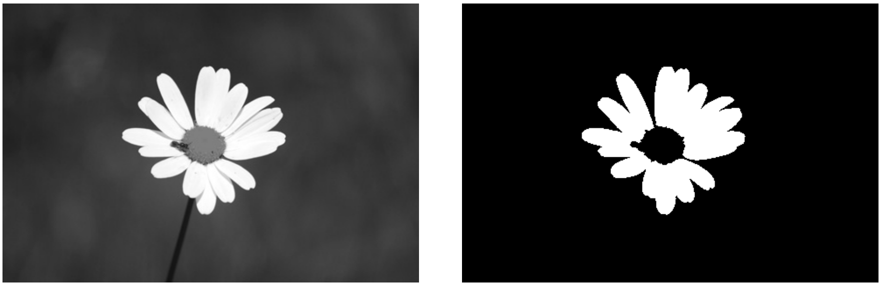
where μ_1 and μ_0 are the mean intensity of the two classes given by

$$\mu_0 = \sum_{i=0}^{k} i p_i / \omega_0 \tag{7}$$

$$\mu_1 = \sum_{i=k+1}^{K} i p_i / \omega_1. \tag{8}$$

We find the optimal threshold by computing the between-class variance for all thresholds and then choosing the one that gives the highest value of the between-class variation.

Figure 4 shows the segmentation of an image by thresholding with a threshold chosen by the Otsu method.



(a) Original image.

(b) Threshold by Otsu.

Figure 4: Segmentation of an image with threshold chosen by the Otsu method.

3 Graph-Cuts

Graph-Cuts [1, 2, 4] is often used for segmenting images. The idea is to express the problem in terms of minimizing an energy function. This function is represented by a graph, where nodes are the variables with edges between them. Minimizing the function will be the same as calculating the maximum flow in the graph if all weights on the edges are non-negative, which there are many algorithms for.

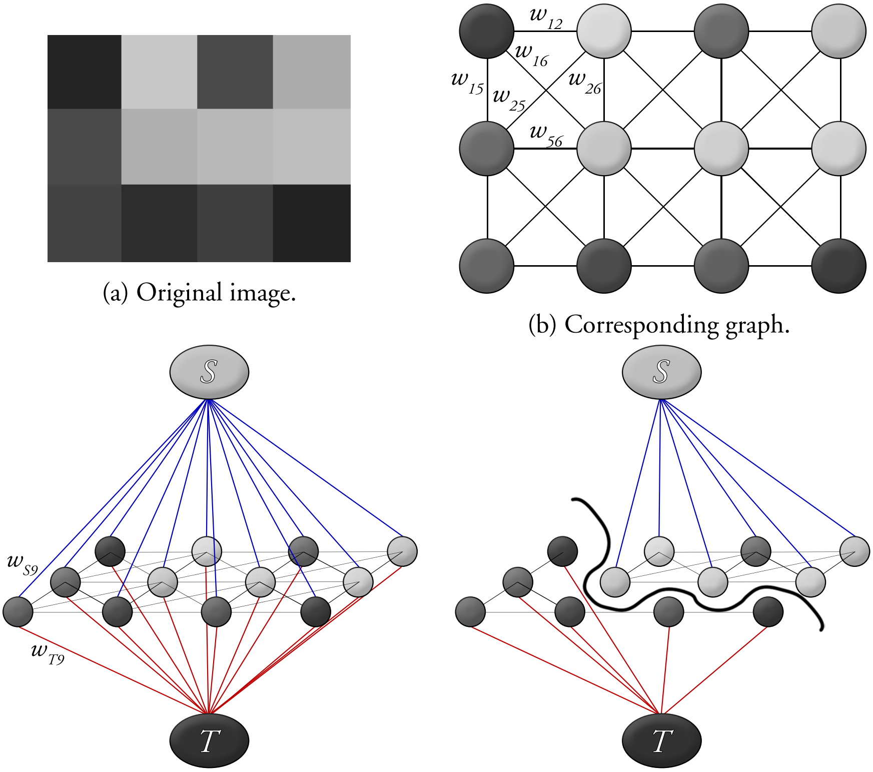
We want to find labels, f_p, for all pixels, p, in the best possible way. These labels are in this case foreground or background. We want to do this in a way that pixels close to each other are more likely to be assigned to the same label. The optimal solution to this problem will be to minimize an energy function consisting of a data part and a regularization part, on the form

$$E(f) = \underbrace{\sum_{i \in \mathcal{P}} D_i(f_i)}_{\text{data part}} + \underbrace{\sum_{i,j \in \mathcal{N}} w_{ij}(f_i, f_j)}_{\text{regularization part}}, \tag{9}$$

where D_i is a term that typically measures how well label f_i fit the data and $\mathcal{P}$ is the set of all pixels. The term w_{ij} describes how hard we should punish if two neighboring pixels have different labels and the set $\mathcal{N}$ is the set of all interacting pixels, neighbors. The terms w_{ij} can either be set individually for each pair of pixels or to a constant, same for all pairs.

An illustration of the graph-cut method is shown in Figure 5. Figure 5a shows a small image of 3×4 pixels. The corresponding graph can be seen in Figure 5b, the pixels are connected in a 8-neighborhood, meaning that pixels not at the border has 8 neighbors. The nodes in the graph correspond to the pixels in the image and they are connected to their neighbors by the edges shown in the illustration. Every edge has a weight associated, denoted by w_{ij}, which is the weight between pixel i and pixel j.

Then all nodes are connected to a foreground and background node called source, S, and sink, T. This can be seen in Figure 5c. The weight of the edge from the foreground node, S, to pixel i is denoted by w_{Si} and usually depends on the intensity difference between the foreground node and the pixel. In the same way the weight for the edge between the background node and a pixel is denoted w_{Ti}. Those will correspond to the term D_i in Equation 9. Then an optimization



(a) Original image.

(b) Corresponding graph.

(c) Foreground and background node.

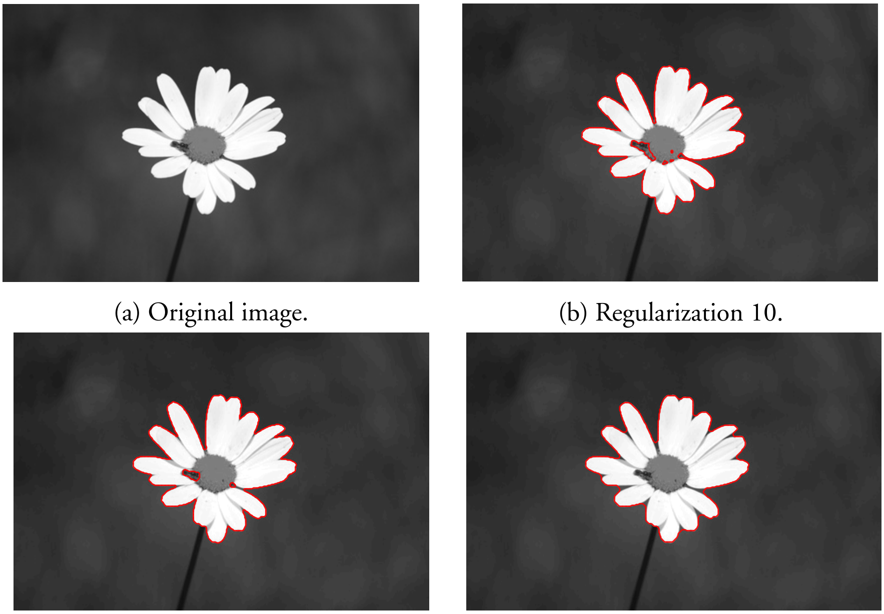
(d) Separated graph.

Figure 5: Illustration of a graph.

algorithm is performed to find the best possible cut, meaning the cut where the total cost is as low as possible. The cut must be done in such a way that all pixels in the end have an edge to either the foreground or the background node, not both. There cannot exist any edges from one side of the graph to the other. The cost for cutting one edge depends on the weight, high weight means high cost, and the total cost is the sum of the weight for all edges that must be cut. Figure 5d shows the separated graph with the cut marked with a black line. The pixels that afterwards are connected to the foreground node are classified as foreground and the others to background.

3.1 Segmentation of an Image using Graph-Cut

Figure 6 shows an example of a segmentation done by graph-cuts for three different values of the regularization term. The weights between the pixels are set to a constant, different in the different images, and the weights to the foreground and background nodes are set to be the intensity difference between the pixel and the foreground and background node respectively. The original image can bee seen in Figure 6a and the result of the segmentation can be seen in Figures 6b, 6c and 6d.



(a) Original image.

(b) Regularization 10.

(c) Regularization 25.

(d) Regularization 100.

Figure 6: Segmentation of an image by the graph-cut method with three different regularization terms.

4 Fast Marching

The fast marching algorithm was presented by J. A. Sethian in [7] and [8]. It is a numerical technique that follows the evolution of an interface. Figure 7 shows such an interface expanding in the directions of the arrows.

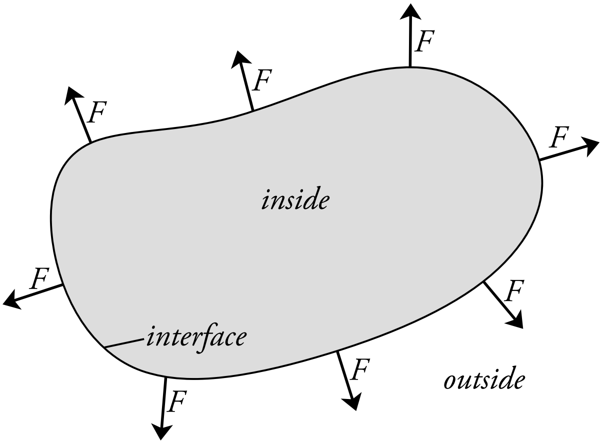


Figure 7: An interface expanding in the direction of the arrows.

In the fast marching method we start with some initial boundary and let this curve expand according to some speed function, F, that depends on the application. In this thesis we will use the fast marching method for segmenting images and therefore we will start the curve somewhere where we are certain it is background and let the curve propagate from this.

The equation of the arrival function in one dimension can be derived by using the equation $x = F \cdot T(x)$, where x is the distance, $T(x)$ the arrival function and F the speed of the curve. Taking the derivative we get

$$1 = F\frac{dT}{dx}.$$

In multiple dimensions this becomes the Eikonal equation

$$|\nabla T|F = 1, \tag{10}$$

where T is the arrival time for the curve and F is the speed function for the curve. The arrival time at a point tells at which time the curve is expected to reach that point. The speed function at a point decides how fast the curve is propagating at a certain point.

After a while we will stop the propagation and let the pixels inside the interface be set to background and the pixels that have not been reached by the curve to foreground.

4.1 Algorithm

We discretize the space by a finite grid, in the case of images it is already discretized and the grid points are the pixels. The grid points are then put in one of the following classes: *Known*, *Trial* or *Far*. *Known* consist of the grid points on the boundary of the interface or points already passed by the curve. These points have already been assigned with an arrival time. *Trial* consist of the points that are neighbors to the boundary and not in *Known*. For these points a temporary arrival time can be computed, and the points are also put in a min-heap to easily and efficiently find the element with the smallest arrival time. *Far* consist of all other points.

The min-heap is a data structure, a tree, organized in a way that the top element is always the smallest one, in this case the smallest means that it has the smallest arrival time. This makes it very efficient to find the smallest arrival time without having to search through all elements.

The fast marching algorithm goes as follows:

1. Initialize, add the points at the initial boundary to *Known*, calculate the temporary arrival time for the neighbors, not in *Known*, according to Section 4.2, add them to *Trial* and to the heap.

2. Take out the first element from the heap which is the point with the smallest arrival time, add it to *Known* and remove it from *Trial*.

3. For all neighbors not in *Known*: update arrival time according to Section 4.2, add the ones not already in *Trial* to *Trial* and to the heap.
   While updating the arrival times, also update the heap.

4. Repeat 2-3 until the heap is empty or until the smallest of the arrival times is larger than some threshold.

4.2 Updating the Arrival Times

To find the arrival function we want to find a solution for the Eikonal equation derived earlier

$$|\nabla T|F = 1.$$

We solve this differential equation numerically by using the following updating scheme

$$\begin{pmatrix} \max(D_{ij}^{-x}T, 0)^2 + \min(D_{ij}^{+x}T, 0)^2 \\ + \max(D_{ij}^{-y}T, 0)^2 + \min(-D_{ij}^{+y}T, 0)^2 \end{pmatrix}^{1/2} = \frac{1}{F_{ij}}, \tag{11}$$

where F_{ij} is the speed at the point (i, j), D_{ij}^{-x} is the one sided derivative in the negative x-direction defined by $D_{ij}^{-x} = \frac{T(x)-T(x-h)}{h}$. In the same way D_{ij}^{+x} is the one sided derivative in the positive x-direction defined by $D_{ij}^{-x} = \frac{T(x+h)-T(x)}{h}$. D_{ij}^{-y} and D_{ij}^{+y} are defined in the same way.

If we use a slightly different approximation of the gradient, introduced in [6], we get the more convenient upwind scheme

$$\begin{pmatrix} \max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 \\ + \max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2 \end{pmatrix}^{1/2} = \frac{1}{F_{ij}}. \tag{12}$$

In this thesis we will use the fast marching algorithm for segmenting images. In this case the grid points are the pixels in the images and the derivatives can be computed more easily. If we call the arrival time we want to calculate T and the arrival times for the neighbors as shown in Figure 8, we can calculate the derivatives as

$$\begin{aligned} D_{ij}^{-x}T &= T - a \\ D_{ij}^{+x}T &= b - T \\ D_{ij}^{-y}T &= T - c \\ D_{ij}^{+y}T &= d - T, \end{aligned} \tag{13}$$

where a, b, c and d are the arrival times for the neighbors of the interesting pixel.

If we assume that the arrival times a and b are both known we can have one of the following cases:

- Both a and b are smaller than T, $T > a$, $T > b$

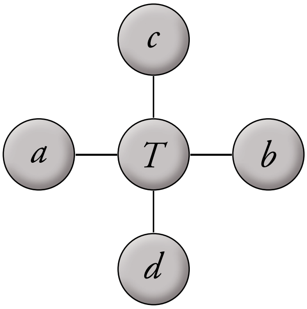


Figure 8: The arrival times for the current grid point and its neighbors. The arrival time we want to update is denoted by T and the arrival times for the neighbors are denoted a, b, c and d.

- a is smaller than T but b is larger, $T > a, T \leq b$
- b is smaller than T but a is larger, $T \leq a, T > b$
- Both a and b are larger than T, $T \leq a, T \leq b$.

For the first case we get $\max(T - a, -(b - T), 0) = \max(T - a, T - b, 0) = T - \min(a, b)$. For the second case we get $\max(T - a, T - b, 0) = T - a$, but since we also know that $b > a$ this will be the same thing as writing $\max(T - a, T - b, 0) = T - \min(a, b)$. The same thing applies for the third case. For the last case we get $\max(T - a, T - b, 0) = 0$.

Then we can reduce the four cases above to just two. In similar way we get two cases when c and d are known.

Equation 12 will look different depending on the size of T. Therefore we have to solve it for the three following cases:

- $T > \min(a, b)$ and $T > \min(c, d)$
- $T > \min(a, b)$ and $T \leq \min(c, d)$
- $T \leq \min(a, b)$ and $T > \min(c, d)$.

The fourth case $T \leq \min(a, b)$ and $T \leq \min(c, d)$ can never occur since T has to always increase and therefore has to be larger than the arrival time for at least one of its neighbors.

In the first case when $T > \min(a, b)$ and $T > \min(c, d)$ we get

$$\begin{aligned}
&\max(T-a, T-b, 0)^2 + \max(T-c, T-d, 0)^2 \\
&= (T - \min(a, b))^2 + (T - \min(c, d))^2 \\
&= 2T^2 - 2(\min(a, b) + \min(c, d))T + \min(a, b)^2 + \min(c, d)^2.
\end{aligned}$$

And we can rewrite Equation 12 as

$$T^2 - (\min(a, b) + \min(c, d))T + \frac{1}{2}\left(\min(a, b)^2 + \min(c, d)^2 + \frac{1}{F^2}\right). \tag{14}$$

This is a quadratic equation with two solutions and we choose the larger of the two. The solution will be real as long as $|\min(a, b) - \min(c, d)| \geq \frac{\sqrt{2}}{F}$. If this does not hold we ignore this case.

For the second case when $T > \min(a, b)$ and $T \leq \min(c, d)$ we instead get

$$\max(T-a, T-b, 0)^2 + \max(T-c, T-d, 0)^2 = (T - \min(a, b))^2 + 0,$$

and we can rewrite Equation 12 to

$$T = \frac{1}{F} + \min(a, b). \tag{15}$$

In the third case we get in the same way

$$T = \frac{1}{F} + \min(c, d). \tag{16}$$

These three cases will yield three different arrival times for the pixel. We are interested in the largest possible solution. For the first case the arrival time must satisfy $T > \min(a, b)$ and $T > \min(c, d)$. If it does not we cannot use this equation.

Since F is always greater than zero it is enough to just look at one of the cases two and three. If we choose $T = \frac{1}{F} + \max(\min(a, b), \min(c, d))$ the arrival time will automatically be greater than both $\min(a, b)$ and $\min(c, d)$, violating the constraints.

By this we can reduce the two latter cases to one, namely

$$T = \frac{1}{F} + \min(a, b, c, d). \tag{17}$$

We can only have two possible outcomes for the solution of this equation, either T is between $\min(a, b)$ and $\min(c, d)$ or greater than both, in the latter case we cannot use this equation.

In the end we just have to calculate the arrival times according to Equation 14. If the solution gets real and it does not violate the constraints this will be the largest of the solutions. Otherwise we calculate the arrival times according to Equation 17.

Usually the arrival times for all the neighbors are not known, but this will not make any substantial difference in the calculations. If we know one of a and b we just set $\min(a, b)$ to the one we know. In the same way we set $\min(c, d)$ to the one we know of c and d. Setting this we can calculate the arrival times exactly as before. If we do not know any of a and b we have to know at least one of c and d, in this case we calculate the arrival time according to Equation 17 setting $\min(a, b, c, d) = \min(c, d)$.

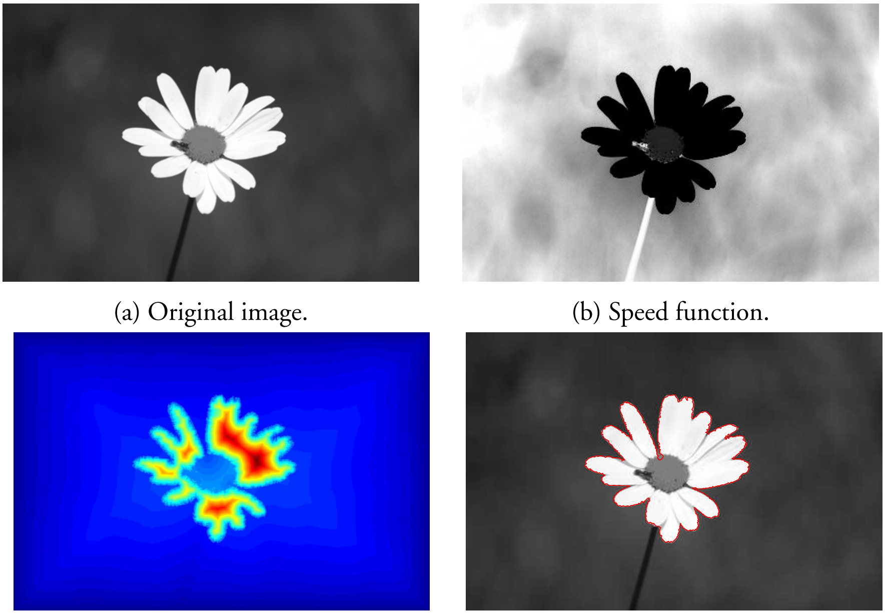
4.3 Segmentation of an Image using Fast Marching

Figure 9 shows an example when fast marching was used to segment an image into foreground and background. The original image can be seen in Figure 9a. This image is then transformed to a speed image shown in Figure 9b where white means high value and black low. The speed function has to be chosen depending on the application. In this case we have a dark background and a bright foreground so we choose a speed function that will give high values for dark pixels and low values for brighter pixels. The speed function is given by

$$F(x, y) = \frac{1}{1 + e^{I(x,y)/v}}, \tag{18}$$

where $I(x, y)$ is the intensity of the given point and v determine the steepness of the function.

Then we initialize the fast marching to start at the edges of the image expanding inwards to the center. The arrival times can be seen in Figure 9c, blue means low arrival times and red and yellow color means higher arrival times. We get the final segmentation shown in Figure 9d by thresholding on the arrival times with



(a) Original image.

(b) Speed function.

(c) Arrival times.

(d) Segmentation.

Figure 9: Segmentation of an image using the fast marching method.

some suitable threshold. The red line shows the border between foreground and background.

References

[1] Y. Boykov and V. Kolmogorov. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124 –1137, sept. 2004.

[3] Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (4):532–550, 1987.

[4] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[5] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[6] E Rouy and A Tourin. A viscosity solutions approach to shape-from-shading. *SIAM journal on numerical analysis*, 29(3):867–884, 1992.

[7] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):pp. 1591–1595, 1996.

[8] J.A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1999.

Paper A

Measurement of Bitumen Coverage of Stones for Road Building, Based on Digital Image Analysis

HANNA KÄLLÉN[†], ANDERS HEYDEN[†], KALLE ÅSTRÖM[†]
AND PER LINDH[‡]

[†] *Centre for Mathematical Sciences, Lund University*
[‡] *Peab, Peab Sverige, Helsingborg, Sweden*

Abstract: The top layer of a road is made up of a mixture of stones and bitumen and the durability is dependent on how well the bitumen adheres to the stones. The standard way of determining the bitumen coverage in the industry is the so called rolling bottle method, where a number of stones covered with bitumen are put in a rolling bottle and the bitumen coverage is estimated after different times. This paper describes a novel method for measuring the bitumen coverage of the stones by using advanced segmentation methods instead of manual inspection. The stones are put on a table and a number of images with different exposure times are taken. The images are normalized and the stones are segmented from the background based on a threshold obtained from an optimality criterion. Then the bitumen covered parts of the stones are segmented based on a graph-cut method. The results are compared to manual inspection and are well in agreement with these.

1 Introduction

When building roads one wants them to be as lasting as possible to avoid expensive repairs. Usually the surface of the road consists of a mixture of stones of different sizes and a petroleum-based material called bitumen. To avoid that stones get loose from the pavement the affinity between the stones and bitumen has to be as good as possible. The affinity is measured by the rolling bottle method. The goal with this paper is to improve the manual analysis in this method using digital image analysis techniques.

1.1 Rolling Bottle Method

The rolling bottle method is a method to investigate the affinity between stones and bitumen. The stones are first mixed with bitumen so that they are completely covered in bitumen. After they been stored for a few days the stones covered in bitumen are put in a glass bottle filled with distilled water.

The glass bottles are then put on a bottle rolling machine, see Figure A.1. On this machine the bottles are rolling for a couple of hours so that some of the bitumen gets teared off from the stones. In this study we first let the bottle roll for four hours. After rolling a few hours the bottle is removed from the machine to estimate the degree of bitumen coverage. The stones are put on a piece of silicon coated paper and two experienced laboratory assistants are visually observing the stones in order to estimate the degree of bitumen coverage.

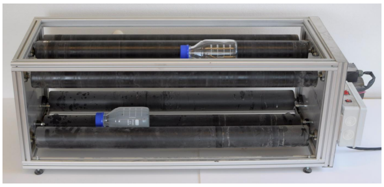


Figure A.1: A bottle rolling machine, the bottles filled with distilled water and stones are rolling on the machine so that bitumen get teared off from the stones.

Afterwards, the stones are put back into the bottle and the bottle continues rolling. After rolling in total 24, 48 and 72 hours, the degree of bitumen coverage is again estimated by the same laboratory assistants.

A problem with current state of the art is that it is not objective, two different labs can get different result since the degree of bitumen coverage is estimated by different laboratory assistants in different labs. It is also very hard to make a correct estimation and the accuracy of the estimations are not sufficient. The purpose of this project is to improve the estimation by taking photographs of the rolled stones and then use digital image analysis techniques to analyze the stones. This would make the method more objective since the same computer program can be used in different labs.

1.2 Previous Work

There is not a lot written about trying to estimate the degree of bitumen coverage by using image analysis. In [12], an algorithm for doing this has been developed. In the proposed method, a cyan-colored background for easy segmentation of the background has been used. To avoid sparkles and reflections in the image a cyan-colored truncated cone, with the camera in one of the bases, is used. To classify pixels either as stones or bitumen, a principal component analysis was implemented. Using the first component the images were thresholded and pixels below the threshold were classified as bitumen. However, this is a completely different approach to the one proposed here.

Concerning segmentation there is a vast literature describing several different segmentation methods. The first methods were based on thresholding and region growing techniques. Also methods from mathematical morphology were frequently used (opening, closing, etc.) in order to smoothen out the contours. The starting point of modern segmentation methods, based on variational formulations, was the introduction of active contours, so called snakes, see [9], later developed as geodesic active contours [5, 6], active region [15].

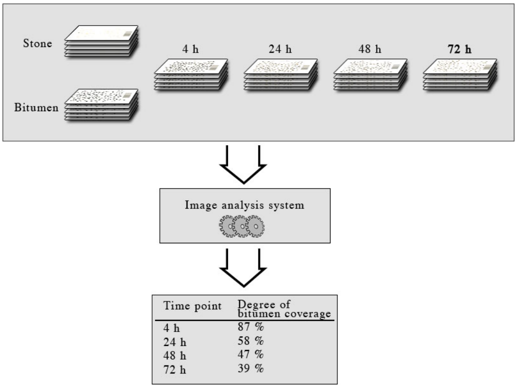
A development of active contours to more general level-sets was done by Osher and Sethian in [14] and [13]. The main advantage of the level-set representation is the flexibility to change topology and improved numerical methods. A faster version of level-sets, so called fast marching, was presented in [16].

Another approach to segmentation based on variational methods is the so called area based methods. The pioneering work, the Chan-Vese method, is based on the Mumford-Shah functional, see [7]. Yet, the main drawback of those methods is the existence of local minima due to non-convexity of the energy functionals. Minimizing those functionals by gradient descent methods makes the initialization critical. A number of methods have been proposed to find global minima such as [1, 8].

A new development into discrete methods, based on graph-theory, is the so called graph-cut methods, introduced by Boykov, Kolmogorov and others, [3, 4, 10]. The main advantage of these methods is that they can guarantee that the solution reaches the global minimum and they are usually very fast.

1.3 Overview of the System

Figure A.2 shows an illustration of the analysis system. Input to the image analysis system are two series of images of stones; one before covered in bitumen and one of stones completely covered in bitumen. These two image series are used as reference images. There is also a series of images from each time point we want to calculate the degree of bitumen coverage. The image analysis system for calculating the degree of bitumen coverage for one time point works in several steps. First adjust the intensity of the image is adjusted so that images with the same exposure time get approximately the same intensity on the background. By thresholding the images we segment the stones from the background, and for each stone we create a mask. For all stones the pixels inside the mask are classified either as bitumen or stone using a graph cut method. When all pixels in the image series are classified, we calculate the degree of bitumen coverage by counting the number of bitumen pixels and stone pixels.



Time point	Degree of bitumen coverage
4 h	87 %
24 h	58 %
48 h	47 %
72 h	39 %

Figure A.2: Illustration of the image analysis system. As input to the system we have a number of series of images, these images are analyzed and we get the degree of bitumen coverage for the requested times.

2 Segmenting Foreground from Background

We first want to segment the foreground from the background, for each stone we want to find a cut-out from the original image and a mask. The segmentation is made by thresholding with a suitable threshold. Since we have a series of images we match the stones in different images to each other and the mask from each images are merged to create a final mask for the stone.

2.1 Intensity Adjustment

In order to be able to compare different images to each other they need to have the same color intensity at the background. In the images there is a calibration stick with three fields; a white, a light gray and a dark gray. The calibration stick is used as a reference and the images are adjusted so that images with the same exposure time all have the same intensity in the three calibration fields. These fields are quite large homogeneous squares. In order to locate them we first use an edge detector to locate edges. Inside the squares there are no edges since the fields are homogeneous, therefore we search for large squares without edges to find the three fields. For each field we then compute a median value for the intensity in the three color channels; red, green, and blue. The images of the stones before they are covered in bitumen are used as reference images and the other images are adjusted according to them. Figure A.3 shows a plot for the intensity for the three fields for one single channel. On the x-axis we have the intensity for the images that we want to adjust and on the y-axis we have the intensity for the reference image. A linear function with slope k and constant term m has been fitted to the points in the plot using the least square method.

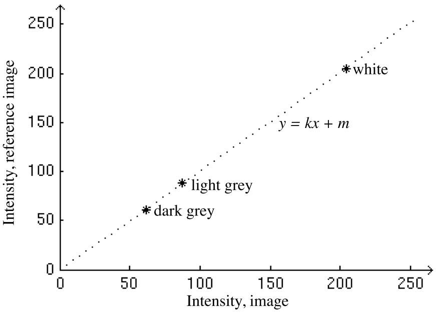


Figure A.3: The intensity for the three fields for one channel, the straight line has been fitted to the points and the equation of this line is used to adjust the image on the x-axis to the same intensity as the reference image on the y-axis.

To adjust one channel of the image to the intensity of the reference image the image is multiplied with the slope of the linear function and then the constant term is added. The new intensity is calculated by

$$I_{new} = kI + m \quad , \tag{A.1}$$

where I_{new} is the adjusted intensity, k the slope of the function, I the original intensity and m the constant term. This adjusts the intensity of one channel in the image, the other channels are adjusted in the same way.

2.2 Segmentation of the Stones

To segment out the stones from the background, we threshold the image. The thresholding is done considering the color difference between the image pixels and a completely white pixel. We calculate the difference by

$$d_i = \sqrt{(I_{vr} - I_{ir})^2 + (I_{vg} - I_{ig})^2 + (I_{vb} - I_{ib})^2} \;, \qquad \text{(A.2)}$$

where d_i is the color difference between the image pixel number i and a white pixel, I_{vj} the intensity for the white pixel in channel j and I_{ij} the intensity for pixel i in channel j.

Since the background is white, background pixels will get small values on d_i and foreground pixels will get larger values. To segment out the foreground from the background, a suitable threshold is chosen and pixels with larger values on d_i than the chosen threshold is considered to be foreground.

A detail from an image is shown in Figure A.4 a) and the color difference between the image pixels and a white pixel is shown in Figure A.4 b). Black means small difference and white large difference. The result of the thresholding is shown in Figure A.4 c), where the white areas are foreground.

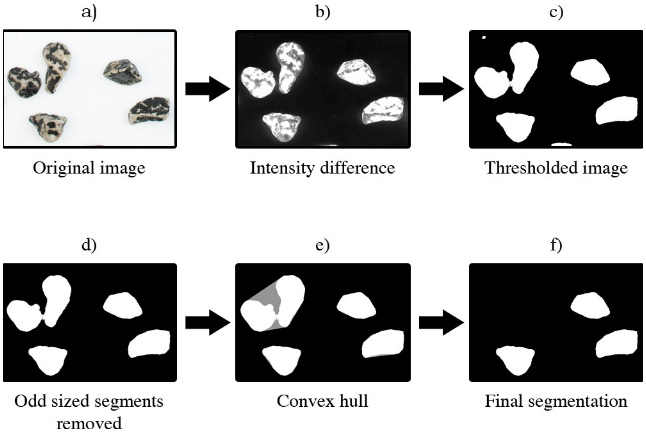


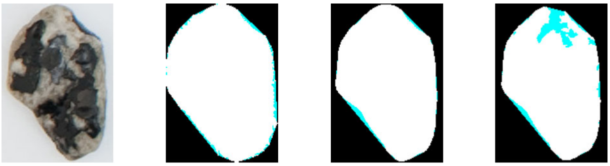
Figure A.4: The segmentation process, first the color difference between the image and a white pixel is calculated for the image, then the image is thresholded. Small segments and segments of strange shape are removed to get the final segmentation.

To remove some foreground that are not stones, we remove segments that are either too small or too large, the remaining segments of appropriate size is likely to be stones. The result can be seen in Figure A.4 d). If two stones are lying too close to each other, the segmentation often fails and classifies some of the background as foreground. This results in that two stones close to each other will be connected. We remove these connected segments by comparing the area of the convex hull to the stones with the area of the stones itself. The convex hull to a stone is the smallest possible convex area which contains the stone. Figure A.4 e) shows the convex hull to the stones, the stones are white and the gray areas are the part of the convex hull that lies outside the stones. If the area of a stone and the area of the convex hull differ too much, the segment is removed. The holes in the segments that are left are filled and the final segmentation can be seen in Figure A.4 f).

Choosing threshold The segmentation algorithm needs a good threshold to work properly. However, the threshold varies between images. To automatically find the most suitable threshold, the following procedure is used. First we threshold the image with a few different thresholds and counts the number of stones, segments of appropriate size. The thresholds are chosen so that the steps between them are quite large. We choose the first threshold that gives us sufficient many stones as a starting threshold. The image is then thresholded with the chosen threshold, segments that are at appropriate size and with not too big difference between the area and the area of the convex hull are classified as stones. The individual stones are then thresholded with a number of higher thresholds and for all thresholds and stones, the area of the stone and the difference between the area of the stone and the area of the convex hull are stored.

Figure A.5 shows the result for one stone after thresholding with three different thresholds. The white areas represent the foreground and the cyan-colored areas are the part of the convex hull to the foreground that lies outside of the foreground. If we set a too low threshold, second image to the left, we will get a non-smooth border between to foreground and background which result in a bigger difference between the area of the convex hull and the area of the foreground. With a too high threshold, the image to the right, we get bays instead which also results in big difference between the area of the convex hull and the area of the foreground.

We define a discrete function, $f(k)$, that depends on which threshold we use. For each threshold we calculate the difference of the areas of the stones and the areas of the convex hull and summarize over all stones. For normalization we divide all area differences by the area difference for the first threshold. The



Original image Threshold = 66 Threshold = 106 Threshold = 154

Figure A.5: Result after thresholding one stone with three different thresholds. The white area is the foreground and the cyan-colored areas are the parts of the convex hull that lies outside of the foreground.

function is defined by

$$f(k) = \sum_{j} \frac{\delta A_{jk}}{\delta A_{j1}} \ , \tag{A.3}$$

where k is the threshold, δA_{jk} is the area difference for stone j and threshold k which is calculated by $A_{convex\ hull} - A_{stone}$.

Figure A.6 shows the graph to the function for a number of thresholds, k. We want the difference between the areas and the areas of the convex hull to be small, and a small value on f means that the sum of all differences are small. Therefore to find a suitable threshold we find the minimum value of f and select the threshold that gives that.

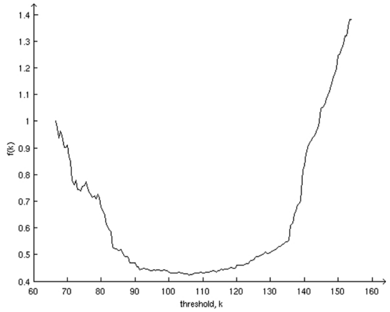


Figure A.6: The sum of area differences as a function of the thresholds. We choose the threshold where the function has its minimum.

2.3 Matching the Stones between Different Images

Sometimes the images in the image series are displaced relative each other. To use all images in the series we have to register all images so that they are on top of each other. The displacements is estimated by matching the stones from different images to each other and compare the coordinates of the center of mass of the stones. The stones in the first image in the series are compared with the stones in the rest of the images. For all stones in the first image, the distance, with respect to the difference in x- and y-coordinates and difference in the area, to all stones in the rest of the images is calculated. The distance is given by

$$d_{ijk} = \|\boldsymbol{s}_i - \boldsymbol{s}_{kj}\|_2 \ , \tag{A.4}$$

where d_{ijk} is the distance between stone number i in the first image, $\boldsymbol{s}_i$, and stone number j in the current image, $\boldsymbol{s}_{kj}$. The vector $\boldsymbol{s}$ contains the area and the x- and y-coordinates of the center of mass of the stone, $\boldsymbol{s} = (A, m_x, m_y)$.

For all stones, $\boldsymbol{s}_i$, and image, k, we select the stone, $\boldsymbol{s}_{jk}$, that matches the stone $\boldsymbol{s}_i$ best, i.e. the stone that gives the smallest distance, d_{ijk}. To avoid mismatches which occurs when the segmentation algorithm fails to segment out stones in some of the images, we set a limit for how much the x- and y-coordinates are allowed to deviate from the x- and y-coordinates for the stone in the first image.

2.4 Merging of the Masks and Images

When the stones are segmented from the background a mask for all stones are stored. The mask is a binary image that is white where there is stone and black otherwise. After the matching process we get a series of masks representing the same stone in the different images. The mask differs somewhat from each other and we want to put these masks on top of each other to create a resulting mask. Figure A.7 a) shows a number of masks representing the same stone.

The masks are then aligned so that the centers of mass coincide. After this, the masks are added together which can be seen in Figure A.7 b). The white pixels correspond to pixels that were inside all masks and the black pixels pixels that were outside all masks. The gray pixels at various shades are pixels that sometimes were inside, sometimes outside the masks.

Pixels that were inside sufficiently many masks, typical half of them or more, is set to white and the rest of the pixels are set to black. Since the pixels at the

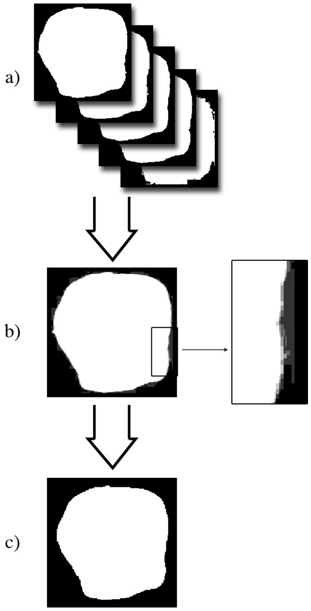


Figure A.7: Merging of the masks, first the masks are aligned and added together. Pixels that were outside of more than half of the masks are removed from the mask, finally the mask is reduced in size with a few pixels.

edge of the masks are uncertain we reduce the size of the mask using morphological erosion and set pixels close to the edge to black. The result can be seen in Figure A.7 c).

To each stone there is also a series of images of the stone, which are cutouts from the original images. These images are also aligned according to the displacements of the masks into a multidimensional image.

3 Segmenting Bitumen from Stone

To segment bitumen pixels from stone pixels we use a graph-cut algorithm, instead of making one big graph of the entire image we set up a small graph for each stone. The weights between pixels in the image depend of the color difference between the pixels. The weights between a pixel to the foreground and background node depends of the similarity between the pixel and a typical bitumen or stone pixel respectively.

3.1 Graph Segmentation

To split the image in foreground and background we use a graph-cut algorithm, the image is represented as a graph which is divided in two parts into foreground and background. A graph consists of nodes and edges, in the graph representation of the image a node represent a pixel in the image, and between adjacent pixels there are edges connecting the pixels. Figure A.8 a) shows a small image with 2×4 pixels and the corresponding graph can be seen in Figure A.8 b). The example shows a 4-connected neighborhood, the pixels are only connected with an edge if they lie next to each other. The weights w_{ij} denotes the dependency between pixel i and pixel j, higher values of w_{ij} makes the pixels more likely to belong to the same class, i.e. foreground or background.

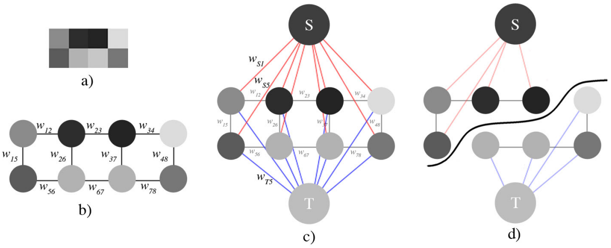


Figure A.8: Illustration of a graph. An image is represented as a graph with edges of different weight between the pixels, the pixels are also connected to a foreground and background node. The graph is then split so that the sum of the removed weights is as small as possible.

All nodes are then connected to a foreground and background node, source and sink, which can be seen in Figure A.8 c). The foreground node is denoted with an S, the background node with a T and the weight between the foreground node and pixel i is denoted with w_{Si}. To segment the image we want to cut edges in the graph in a way so that all nodes in the end have an edge to either the foreground or the background node. We also want the total cost for doing that to be as small as possible, were the cost to remove an edge depends on the weight of the edge. The total cost for splitting the graph is the sum of the weight of the edges we have to remove. Figure A.8 d) shows the separated graph with the cut marked with a black line. Pixels connected to the S node are classified as foreground and pixels connected to the T node are classified as background.

3.2 Clustering

To determine which pixels that is stone and bitumen respectively, we want to find information about how a typical stone and bitumen pixel looks like. To find this information we use the two reference series to cluster the pixels in different cluster where pixels that are similar to each other belong to the same cluster. When we do the clustering, we use all images in the series and the pixels will therefore be multidimensional arrays with three channels from each image. The clustering process for clustering pixels in one stone is shown in Figure A.9. The pixel vector to the right in Figure A.9 a) corresponds to the pixel marked with a square in the original images to the right. All pixel vectors are then clustered together to some different clusters, which is illustrated in Figure A.9 b). The clustering is done using a k-means algorithm, described in [11]. For each cluster a cluster center is calculated, which is the mean value for all vectors belonging to that cluster. Figure A.9 c) shows the result of the clustering, where the different colors in the figure corresponds to different clusters.

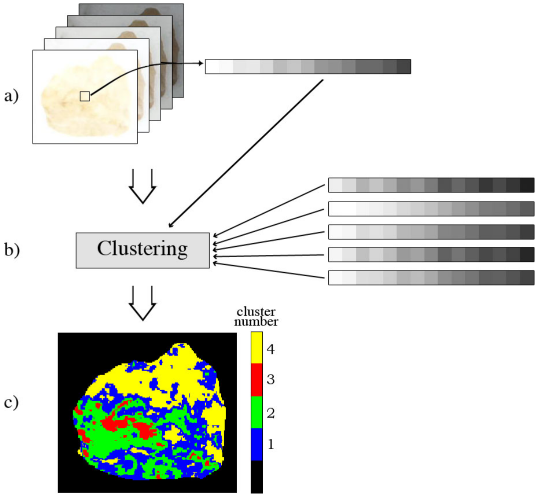


Figure A.9: Illustration of clustering. All pixels from the image is extracted and clustered together to a few cluster. The last image shows which pixels that belong to which cluster.

In the same way, we cluster the pixels in the images of the stones completely covered in bitumen, in this way we get some pixel vectors, the cluster centers, representing a typical bitumen pixel.

3.3 Segmenting the Stone Pixels from the Bitumen Pixels

For segmenting the images into stone and bitumen we use the graph-cut method described in Section 3.1. Each stone is represented with a small graph where the weight between the pixel nodes and the stone, background, and bitumen, foreground, node depends on how similar the pixels are to the typical stone and bitumen pixel respectively. For each pixel, the distance to all cluster centers are calculated, both the cluster centers representing the stones and the centers representing bitumen. The distance between a pixel and a cluster center is calculated by

$$d_{ik} = \|\boldsymbol{p}_i - \boldsymbol{p}_k\|_2 \ , \tag{A.5}$$

where d_{ik} is the distance between pixel number i and cluster center number k and $\boldsymbol{p}_i$ and $\boldsymbol{p}_k$ are the pixels vectors containing the intensity for all images and channels for pixel i and cluster center k.

Then the weights to the S-node, bitumen, are chosen, inspired by [2], as

$$w_{Si} = e^{-x_i^2/\left(2\sigma^2\right)} \ , \tag{A.6}$$

where x_i is a normalized distance from the pixel to the closest bitumen cluster and σ a constant that controls how fast the exponential function decreases. The distance x_i is calculated by

$$x_i = \frac{d_{ib}}{d_{ib} + d_{is}} \ , \tag{A.7}$$

where d_{ib} is the distance from pixel i and the closest bitumen cluster and d_{is} the distance to the closest stone cluster. The weights to the background node, stone, w_T are then set to $w_T = 1 - w_S$.

We choose the weight between pixels in the image to an exponential function that depends on the color difference between the pixels. Then we calculates the weight by

$$w_{ij} = 10e^{d_{ij}^2/\left(2\sigma^2\right)} \ , \tag{A.8}$$

where d_{ij} is the color difference between pixel i and pixel j and σ a constant. The distance d_{ij} is normalized, so that it is independent of how many images it is in the image series and calculated by

$$d_{ij} = \frac{\|\boldsymbol{p}_i - \boldsymbol{p}_k\|_2}{255\sqrt{n}} \quad , \tag{A.9}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_k$ are the pixel vectors for pixel number i and j respectively and n is the length of these vectors.

When we know all weights we can split the graph to background and foreground according to Section 3.1. This will tell us which pixels that are bitumen and which pixel that are stone.

3.4 Calculation of the Degree of Bitumen Coverage

To calculate the degree of bitumen coverage we simply count the number of pixels that were classified as bitumen and the number of pixels that were classified as stone. The degree of bitumen coverage is given by

$$db = \frac{n_b}{n_s + n_b} \quad , \tag{A.10}$$

where db is the degree of bitumen coverage, n_b is the number of bitumen pixels and n_s the number of stone pixels.

4 Experiments

To evaluate the accuracy of the image analysis system, different stone materials with different color and darkness were tested. Each material was analyzed both by the image analysis system and by experienced laboratory assistants. As the standard prescribes, the visual estimation of the degree of bitumen coverage by the laboratory assistants were done by investigating a few number of stones and compare to prescribed guidelines. Since the exact degree of bitumen coverage is hard to estimate it is only done by 5 % accuracy.

Figure A.10 shows the result of the segmentation for one of the images, the stone material in this image is denoted D and is of light rock type. The stones in the images have been rolling in the bottle for 72 hours before they were placed on the paper and photographed. The left image shows a survey picture and the right image shows a detail from the image. The red lines shows the boundary

	trial 1	trial 2	trial 3
degree of bitumen coverage	42.4 %	42.4 %	43.4 %

Table A.1: The degree of bitumen coverage for material D calculated from three different images. The stones are the same in all images but placed differently.

between background and foreground and the blue lines shows the boundary between stones and bitumen.

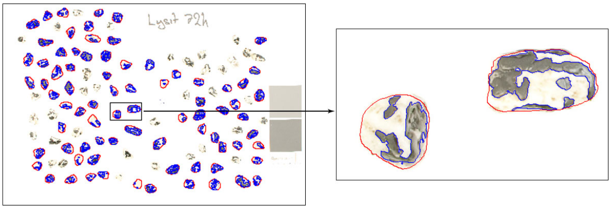


Figure A.10: Result after 72 hours for stone material D. The red lines show the border between stones and background and the blue lines shows the border between bitumen and stone.

For this image the degree of bitumen was calculated to 38.6 %. At the same time two independent laboratory assistants estimated the degree of bitumen coverage to 35 and 40 %. The degree of bitumen coverage calculated with the image analysis method lies in between them. Together with visual study of the result image one can conclude that the image analysis gives accurate result.

Since we can only see one side of the stones we want to investigate the repeatability of the method. To do this we collected all the stones and placed it randomly on the paper again. New images were taken and the degree of bitumen was calculated again. This was repeated a few times. The result of the different trials can be seen in Table A.1.

The result after 72 hours for a different type of stone is shown in Figure A.11. The stone material, denoted by B, is reddish and a bit darker than the previous material.

The degree of bitumen coverage was calculated to 13.6 %. The laboratory assistants both estimated the coverage to 10 %. By studying the result image one can see that the segmentation algorithm seems to segment the bitumen from the stone in a correct way. The result is also close to the estimation by the laboratory assistants.

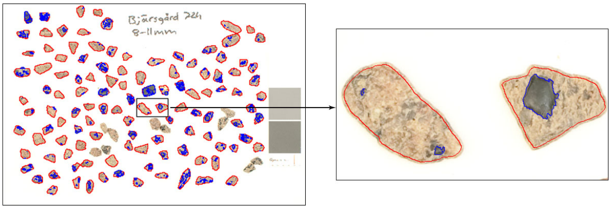


Figure A.11: Result after 72 hours for stone material B. The red lines show the border between stones and background and the blue lines shows the border between bitumen and stone

	trial 1	trial 2	trial 3
degree of bitumen coverage	13.6 %	12.7 %	14.1 %

Table A.2: The degree of bitumen coverage for material B calculated from three different images. The stones are the same in all images but placed differently.

In the same way as the other lighter stones these stones were also collected, again placed on the paper and photographed. Then the new images were analyzed to again investigate how the result changes depending on the placement of the stones. The result can be seen in Table A.2.

	degree of bitumen coverage		
	image analysis	assistant 1	assistant 2
material D after 4 h	92.6 %	95 %	95 %
material D after 24 h	57.5 %	60 %	60 %
material D after 48 h	46.4 %	45 %	45 %
material D after 72 h	38.6 %	40 %	35 %
material B after 72 h	13.6 %	10 %	10 %

Table A.3: The degree of bitumen coverage for the different stone materials and times estimated both by the image analysis system and by two independent laboratory assistants.

Table A.3 shows a summary of the results for two different kinds of stones. For the first material the degree of the bitumen coverage has been calculated at four different times, but for the second stone material only one time is considered. In all cases the automatic image analysis method gives result that is close to the visual estimations by the laboratory assistants. Since the results from the image analysis is close to the results from the traditional method, it indicates that our method gives result close to the true degree of bitumen coverage even though the results from the laboratory assistants should not be considered as the ground truth.

5 Conclusions and Future Work

Using image analysis to estimate the degree of bitumen coverage of the stones works well for light and middle dark rock types. The method presented in this paper gives accurate results and the repeatability is good.

The method has also been tested on darker, almost black, stones. For these stones the color and intensity of the stones are almost the same as for the bitumen, and the described method is not suitable. To solve the problem with darker stones we need to use a different photo and lightening arrangement.

References

[1] Ben Appleton and Hugues Talbot. Globally minimal surfaces by continuous maximal flow. *pami*, 28(1):106–118, Jauary 2006.

[2] Y. Boykov and G. Funka-Lea. Graph cuts and efficient nd image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.

[3] Y. Boykov and V. Kolmogorov. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124 –1137, sept. 2004.

[5] V. Caselles, F. Catte, T. Coll, and F. Dibos. A Geometric Model for Active Contours. *Numerische Mathematik*, 66:1–31, 1993.

[6] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[7] T. Chan and L. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[8] Tony F. Chan, Selim Esedoglu, and Mila Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal of Applied Mathematics*, 66(5):1632–1648, 2006.

[9] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Computer Vision*, 1(4):321–331, 1987.

[10] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[11] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

[12] F. Merusi, A. Caruso, R. Roncella, and F. Giuliani. Moisture susceptibility and stripping resistance of asphalt mixtures modified with different synthetic waxes. *Transportation Research Record: Journal of the Transportation Research Board*, 2180(-1):110–120, 2010.

[13] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[14] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[15] N. Paragios and R. Deriche. Geodesic active regions: a new paradigm to deal with frame partition problems in computer vision. *International Journal of Visual Communication and Image Representation, Special Issue on Partial Differential Equations in Image Processing, Computer Vision and Computer Graphics*, 2002.

[16] J.A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93(4):1591–1595, 1996.

Paper B

Measuring Bitumen Coverage of Stones using a Turntable and Specular Reflections

Hanna Källén[†], Anders Heyden[†] and Per Lindh[‡]
[†] *Centre for Mathematical Sciences, Lund University*
[‡] *Peab, Peab Sverige, Helsingborg, Sweden*

Abstract: The durability of a road is among other factors dependent on the affinity between stones in the top layer and bitumen that holds the stones together. Poor adherence will cause stones to detach from the surface of the road more easily. The rolling bottle method is the standard way to determine the affinity between stones and bitumen. In this test a number of stones covered in bitumen are put in a rolling bottle filled with water. After rolling a number of hours the bitumen coverage are estimated by visually investigating the stones. This paper describes a method for automatic estimation of the degree of bitumen coverage using image analysis instead of manual inspection. The proposed method is based on the observation that bitumen reflects light much better than raw stones. In this paper we propose a method based on the reflections to estimate the degree of bitumen coverage. The stones are put on a turntable which is illuminated and a camera is placed straight above the stones. Turning the table will illuminate different sides of the stones and cause reflections on different part of the images.

1 Introduction

When building roads one wants them to be as lasting as possible to avoid expensive repairs. Usually the surface of the road consists of a mixture of stones of different sizes and a petroleum-based material called bitumen. To avoid that stones get loose from the pavement the affinity between the stones and bitumen has to be as good as possible. The affinity is measured by the rolling bottle method. The goal with this paper is to improve the manual analysis in this method using digital image analysis techniques.

1.1 Rolling Bottle Method

The rolling bottle method is a method to investigate the affinity between stones and bitumen. The stones are first mixed with bitumen so that they are completely covered in bitumen. After they have been stored for a few days the stones covered in bitumen are put in a glass bottle filled with distilled water.

The glass bottles are then put on a bottle rolling machine, see Figure B.1. On this machine the bottles are rolling for a couple of hours so that some of the bitumen gets teared off from the stones. After rolling a few hours the bottle is removed from the machine to estimate the degree of bitumen coverage. The stones are put on a piece of silicon coated paper and two experienced laboratory assistants are visually observing the stones in order to estimate the degree of bitumen coverage.

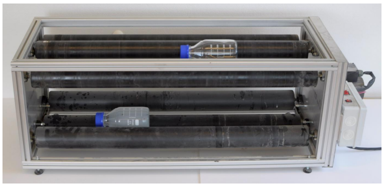


Figure B.1: A bottle rolling machine.

A problem with current state of the art is that it is not objective, two different labs can get different result since the degree of bitumen coverage is estimated by different laboratory assistants in different labs. It is also very hard to make a correct estimation and the accuracy of the estimations are not sufficient. The purpose of this project is to improve the estimation by taking photographs of the rolled stones and then use digital image analysis techniques to analyze the stones. This would make the method more objective since the same computer program can be used in different labs.

1.2 Previous Work

In [8], an algorithm for trying to estimate the degree of bitumen coverage by using image analysis has been developed. In the proposed method, a cyan-colored background for easy segmentation of the background has been used. To avoid sparkles and reflections in the image a cyan-colored truncated cone, with the camera in one of the bases, is used. To classify pixels either as stones or bitumen, a principal component analysis was implemented. Using the first component the images were thresholded and pixels below the threshold were classified as bitumen.

A more advanced method for estimating the degree of bitumen coverage was suggested by [12]. To avoid reflections in the bitumen surface, the stones are put in a crystallization dish where they were covered with distilled water. A plastic cylinder were put around the aggregates and illuminated from outside to ensure diffuse lightening to prevent shadows to occur. A probability based segmentation method was used for segmenting the images. To train parameters in the classifier, reference images on the background, the raw aggregates and aggregates completely covered in bitumen were used.

Both these methods rely on a difference in appearance between the aggregates and bitumen. In this paper we focus on the more difficult problem when the color of the stones are very similar to the color of bitumen.

Concerning segmentation there is a vast literature describing several different segmentation methods. The first methods were based on thresholding and region growing techniques. Also methods from mathematical morphology were frequently used (opening, closing, etc.) in order to smoothen out the contours. The starting point of modern segmentation methods, based on variational formulations, was the introduction of active contours, so called snakes, see [6].

A development of active contours to more general level-sets was done by Osher and Sethian in [10] and [9]. The main advantage of the level-set representation is the flexibility to change topology and improved numerical methods. A faster version of level-sets, so called fast marching, was presented in [11].

Another approach to segmentation based on variational methods is the so called area based methods. The pioneering work, the Chan-Vese method, is based on the Mumford-Shah functional, see [4]. Yet, the main drawback of those methods is the existence of local minima due to non-convexity of the energy functionals. Minimizing those functionals by gradient descent methods makes the initialization critical. A number of methods have been proposed to find global minima such as [1, 5].

A new development into discrete methods, based on graph-theory, is the so called graph-cut methods, introduced by Boykov, Kolmogorov and others, [2, 3, 7]. The main advantage of these methods is that they can guarantee that the solution reaches the global minimum and they are usually very fast.

2 Methods for Estimating the Degree of Bitumen Coverage

A problem when trying to take images of stones covered in bitumen is that we often get specular reflections in the bitumen. The idea in this paper is to instead of trying to avoid the specular reflections we try to use it for segmenting the images. For that reason we want to take several images, typically 20-30, with light from all possible directions. In practice it turns up to be more practical to place stones on a turntable which we turned a bit between images than to place a high number of light sources around the scene.

Our system for analyzing the images then consists of three parts. First we have to register the images to each other. After registration we segment the foreground, stones, from the background using all images. Last, for the pixels classified as foreground we estimate the degree of bitumen coverage by using a probability based classification method.

2.1 Experimental Setup

The setup used to take images can be seen in Figure B.2. In the setup we have one camera, one light source and one turntable. The camera is placed straight above the turntable and facing downwards, looking at the stones from above. Beside the camera we have a light source that illuminates the stones from one direction. By turning the turntable we get light from many more directions. To easier segment the stones from the background we use a blue background on the turntable.

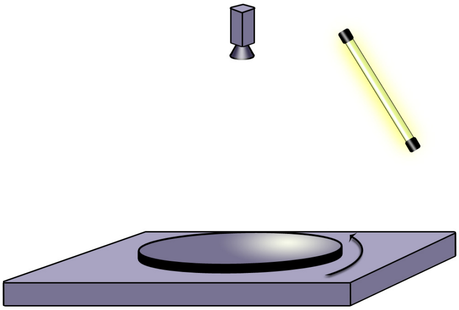


Figure B.2: The experimental setup for taking the pictures. The camera is looking straight down to the turntable, the lamp gives light from one direction but turning the turntable different sides of the stones will be illuminated.

Figure B.3 shows some examples of images that we get from our setup, these stones are completely covered in bitumen.

2.2 Registration and Segmentation of Stones from Background

To be able to use the images we have to register them to each other. This is done by extracting some corresponding key points in all images and compute a homography from all images to some reference image. The homography is a 3×3

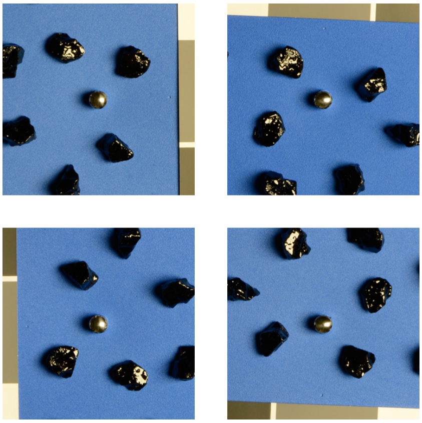


Figure B.3: Example of images, the original images before transformation.

matrix H so that

$$\lambda \mathbf{y} = H\mathbf{x}, \tag{B.1}$$

where H is the homography, λ a scaling factor, $\mathbf{x}$ is the point in the reference image and $\mathbf{y}$ is the corresponding point in the image that we want to transform, $\mathbf{x}$ and $\mathbf{y}$ are given in homogeneous coordinates.

Then the images are transformed according to the homography associated with the current image. Figure B.4 shows the same images as Figure B.3 after the transformations.

When the images are transformed we want to find out which part of the image that is stone and which part is background. Since the shadows are quite sharp in the images we take a mean image of all the images and use that for segmentation. The mean image can be seen in Figure B.5, now the shadows are much smoother. The segmentation is done by thresholding in the blue channel of the image. The threshold is chosen manually, which is not crucial for the segmentation result.

2.3 Estimation of the Degree of Bitumen Coverage

To estimate the degree of bitumen coverage we look at the difference between the highest value for a pixel through all images and the lowest value. If there are any specular reflections in any of the images this difference will be high. The difference image for stones completely covered in bitumen can be seen in Figure B.6,

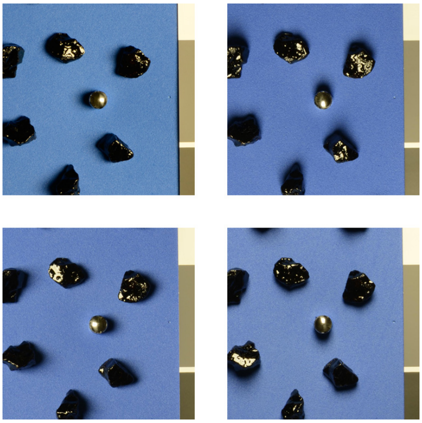


Figure B.4: Example of images, the images after transformation.

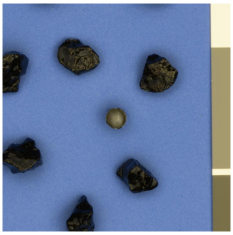


Figure B.5: The mean image used to segment foreground, stones, from background.

as can be seen in the image we do not get reflections everywhere. We use some reference images with stones covered in bitumen and the raw stones to build his-

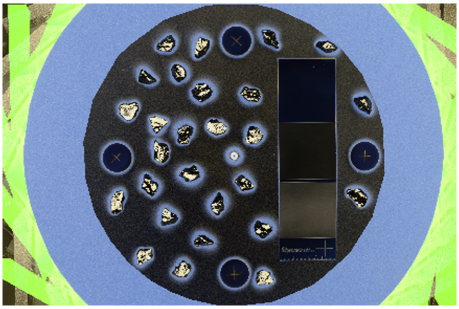


Figure B.6: The difference image.

tograms for the difference of the highest and the lowest value. These histograms are then used to find a probability function, that tells how likely a pixel with a certain difference is to be bitumen and stone respectively. The histograms are normalized so that they sum up to 1. The probability that a pixel with intensity i is bitumen can be calculated by

$$P_b(i) = \frac{h_b(i)}{h_b(i) + h_s(i)}, \tag{B.2}$$

where $P_b(i)$ is the probability that a pixel with intensity i is bitumen, $h_b(i)$ is the value of the histogram for bitumen pixels with intensity i and $h_s(i)$ is the is the value of the histogram for stone pixels with intensity i.

Figure B.7 shows the histograms and the probability functions for two different stone materials. The blue curves show the curves for bitumen and the red curves show the curves for stone.

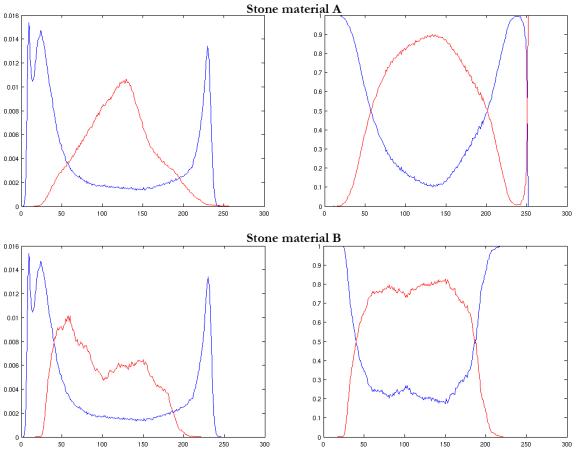


Figure B.7: Histograms and probability functions for stone material A and B.

To estimate the degree of bitumen coverage for stones that are partly covered in bitumen the differences for all pixels are computed. The image is also segmented into foreground and background. For all the foreground pixels, the probability that a pixel is bitumen is calculated. Figure B.8 shows an image of the probabilities that pixels is bitumen, white means that a pixels is very likely to be bitumen and black pixels are very unlikely to be bitumen. Then the degree of

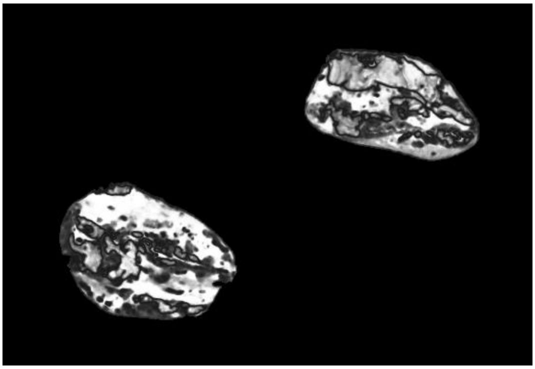


Figure B.8: Probability image for being bitumen, white indicates high probability and black low.

bitumen coverage is estimated by

$$dbc = \frac{\sum_i P(i \text{ is bitumen})}{N}, \tag{B.3}$$

where dbc is the degree of bitumen coverage, $P(i \text{ is bitumen})$ is the probability that pixel number i is bitumen and N is the total number of pixels. Only pixels that were classified as foreground are considered.

3 Experiments and Results

The method has been tested for two different stone materials, one dark and one lighter. The results from the image analysis have been compared with the visual investigation by experienced laboratory personnel. Table B.1 shows the result for the two materials. The results are close to the visual estimations by the laboratory assistant, but the visual estimation could also deviate from the true answer.

	degree of bitumen coverage	
	image analysis	manual inspection
material A	46.8 %	50 %
material B	29.2 %	35 %

Table B.1: The degree of bitumen coverage for the different stone materials estimated both by the image analysis system and by visual inspection.

4 Conclusions and Future Work

With this method we can automatically compute the degree of bitumen coverage even for stone materials with a darker color close to the color of bitumen. We still have to work a bit on the lightening arrangement to ensure to get more specular reflections in the images.

Acknowledgements

This work was founded by SBUF. We also want to thank PEAB for supplying images and stone material.

References

[1] Ben Appleton and Hugues Talbot. Globally minimal surfaces by continuous maximal flow. *pami*, 28(1):106–118, Jauary 2006.

[2] Y. Boykov and V. Kolmogorov. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124 –1137, sept. 2004.

[4] T. Chan and L. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[5] Tony F. Chan, Selim Esedoglu, and Mila Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal of Applied Mathematics*, 66(5):1632–1648, 2006.

[6] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Computer Vision*, 1(4):321–331, 1987.

[7] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[8] F. Merusi, A. Caruso, R. Roncella, and F. Giuliani. Moisture susceptibility and stripping resistance of asphalt mixtures modified with different synthetic waxes. *Transportation Research Record: Journal of the Transportation Research Board*, 2180(-1):110–120, 2010.

[9] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[10] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[11] J.A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93(4):1591–1595, 1996.

[12] F. Wellner, S. Kayser, L. Marschke, D. Schlesinger, A. Morgenstern, and C. Schulze. Optimierung der affinitätsprüfung - verbesserung der präzision der prüfung zur besimmung des haftverhaltens zwischen groben gesteinskörnen und bitumen. *Forschung Straßenbau und Straßenverkehrstechnik*, 2011.

Paper C

Estimation of Grain Size in Asphalt Samples using Digital Image Analysis

HANNA KÄLLÉN[†], ANDERS HEYDEN[†] AND PER LINDH[‡]
[†] *Centre for Mathematical Sciences, Lund University*
[‡] *Peab, Peab Sverige, Helsingborg, Sweden*

Abstract: Asphalt is made of a mixture of stones of different sizes and a binder called bitumen, the size distribution of the stones is determined by the recipe of the asphalt. One quality check of asphalt is to see if the real size distribution of asphalt samples is consistent with the recipe. This is usually done by first extract the binder using methylenchloride and the sieving the stones and see how much that pass every sieve size. Methylenchloride is highly toxic and it is desirable to find the size distribution in some other way. In this paper we find the size distribution by slicing up the asphalt sample and use image analysis techniques on the cross-sections. First the stones are segmented from the background, bitumen, and then rectangles are fit to the detected stones. We then get the sizes of the stones by taking the with of the rectangle.

1 Introduction

1.1 Background

One of the quality control of hot mix asphalt (HMA) is to check the recipe of the mixture. This check could be performed on HMA directly from the plant or on samples that are drilled from the finished pavement. The quality check includes determination of particle size distribution via sieving. The test consists of the determination of the particle size distribution of the aggregates in the bituminous mixture by sieving and weighing. A granulometric analysis of the aggregate is performed after binder extraction. The binder extraction is often performed with methylenchloride (dichloromethane). Methylenchloride is toxic and the European Union has decided to reduce the usages of methylenchloride.

After the binder distraction the material is sieved in order to determine the particle size distribution. The method proposed in this paper is an alternative to binder extraction in combination with sieving. The asphalt sample to be analyzed is sliced into a few slices. Then each two-dimensional slice is analyzed by image analysis methods to find the size distribution for all slices.

1.2 Related Work

The article [1] combine a canny edge detection with thresholding to find an initially segmentation of the image. They also use the watershed algorithm to deal with the problem of grains being too close to each other and therefore segmented as one segment. The watershed algorithm is performed on a distance image, the binary image of the foreground after applying boolean distance to it. In this way the separate segments with concavities along the contours.

In [4] they use the L*a*b color space to segment the grains from the bitumen background. After converting to the L*a*b color space they easily find a good threshold and then they perform thresholding. The same problem as mentioned in the earlier article with undersegmented segments they also find and solve it in the same way with the watershed algorithm.

Both articles compare to the true aggregate gradation and both show good correlations.

2 Methods

To estimate the size distribution of the stones in one slice of the sample we first segment the image by using the fast marching method. In the resulting segmentation it often happens that stones lying too close to each other belongs to the same foreground segment even if they are not supposed to. In order to get correct size estimations we need to separate these segments. This is done by applying some morphological operations. After this correction rectangles are fit to the segments and the sizes is estimated as the width of the fitted rectangle for each stone.

2.1 Fast Marching

The fast marching algorithm was presented by J. A. Sethian in [2] and [3]. It is a numerical technique that follows the evolution of an interface. It numerically solves the Eikonal equation

$$|\nabla T|F = 1, \tag{C.1}$$

where T is the arrival time for the curve and F is the speed function for the curve. For solving the Eikonal equation at a grid point we use the following scheme for updating the arrival times

$$\begin{pmatrix} \max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 \\ + \max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2 \end{pmatrix}^{1/2} = \frac{1}{F_{ij}}, \tag{C.2}$$

where D_{ij}^{-x} is the one sided derivative in the negative x-direction defined by $D_{ij}^{-x} = \frac{T(x)-T(x-h)}{h}$.

The grid points are put in one of the following classes: *Known*, *Trial* or *Far*. *Known* consist of the grid points on the boundary or points already passed. These points have already been assigned with an arrival time. *Trial* consist of the points that are neighbors to the boundary and not in *Known*. For these points a temporary arrival time can be computed, and the points are also put in a min-heap to easily and efficiently find the element with the smallest arrival time. *Far* consist of all other points.

The fast marching algorithm goes as follows:

1. Initialize, add the points at the initial boundary to *Known*, calculate the temporary arrival time for the neighbors, not in *Known*, according to equation C.2, add them to *Trial* and to the heap.

2. Take out the first element from the heap which is the point with the smallest arrival time, add it to *Known* and remove it from *Trial*.

3. For all neighbors not in *Known*: update arrival time according to equation C.2, add the ones not already in *Trial* to *Trial* and to the heap.
   While updating the arrival times, also update the heap.

4. Repeat 2-3 until the heap is empty or until the smallest of the arrival times is larger than some threshold.

To start the fast marching algorithm we need to initialize it at some points, points we know belongs to the background. To do this we threshold the image with a low threshold. The chosen threshold should be so low that points below it can certainly not be stone. Then we let the curve propagate from these points. For larger areas below the threshold we start the curve at the boundary and tag the points inside as *Known*.

To know at what speed the curve should advance at a certain point we need a speed function. We choose to use a logistic function defined by

$$F(x, y) = \frac{1}{1 + e^{I(x,y)/v}}, \tag{C.3}$$

where $F(x, y)$ is the speed at point (x, y), $I(x, y)$ is the intensity at the same point in the input image and v is a scaling factor that controls the steepness of the function. The function for three different values on v can be seen in Figure C.1

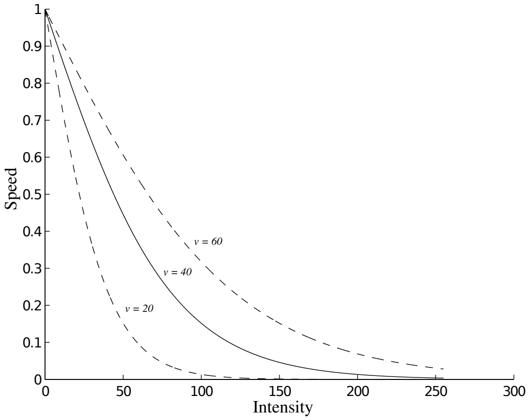


Figure C.1: The speed function used with three different values on the steepness factor v. On the y-axis is the speed and on the x-axis is the pixel intensity.

The speed function is then applied to the original image converted to grayscale. The original image and the speed image can be seen in Figure C.2.

The fast marching algorithm produce the arrival times for every grid points, pixels in this case. The image is then segmented by thresholding the arrival times. Pixels with arrival times smaller than the threshold are set to background and the rest to foreground.

2.2 Refining the Segmentation

Sometimes when stones are very close to each other in the sample it is not enough bitumen between them, and the fast marching algorithm have a hard time separating those stones into different segments. That would require a high threshold,

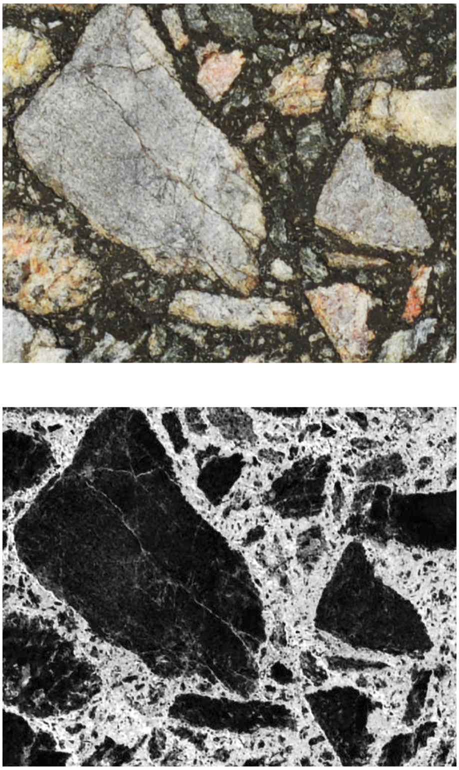


Figure C.2: The original image on top and the speed image at the bottom.

but using a too high threshold would make it impossible to find the smaller stones in the sample. Therefore we choose a lower threshold and try to deal with the undersegmented segments in another way. Figure C.3 shows such a segment.

For this purpose we use the morphological methods erosion and dilation in a clever way. First we perform binary erosion on the segments, this will, if the segments have concavities, cause the segment to eventually fall apart into two or more segments. If the size of the segment, after performing erosion repeatedly without the segment falling apart, is less than half of the original size, we guess that the shape is good and take back the original segment. When it does fall apart we continue the process on the new smaller segments keeping count on how many times we perform the erosion. Afterwards we perform dilation the same number of times that we performed erosion until the last separation of the

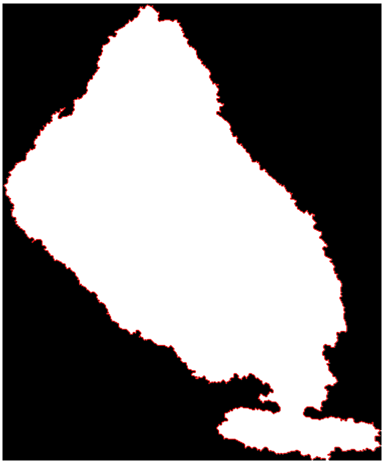


Figure C.3: A segment with strange shape that probably should be two different segments. The red line shows the border of the segment.

segment. In this way all non-convex segments get separated in two or more new segments. Some smoothing of the contour will also occur, but since we are not interested in the shape itself only the size it will not cause any problem for the overall method. Both erosion and dilation are performed with a disc of radius 4 pixels as structuring element.

The algorithm can be summarized as:

```
For all segments:
    until the segment is too small:
        perform binary erosion
        if separated:
            run the algorithm again on the new segments
    dilate if the segment was split earlier
```

This will in some cases cause some overlap between different segments. Since we do not want that, we find these intersections of segments, and assign all those pixels to belong to either of the overlapping segments. This is done by first subtracting the overlap from all the interesting segments and then iteratively alternate between first dilate one of the two segment, assign the pixels in the intersection between the overlap and the dilated segment to that segment and then perform the same thing with the other segment. This continues until all points in the overlap is assigned to one of the segments.

After applying this algorithm to the segments in Figure C.3 we get the result shown in Figure C.4. The image shows the original segment with the border of the new one showed with the red contour.

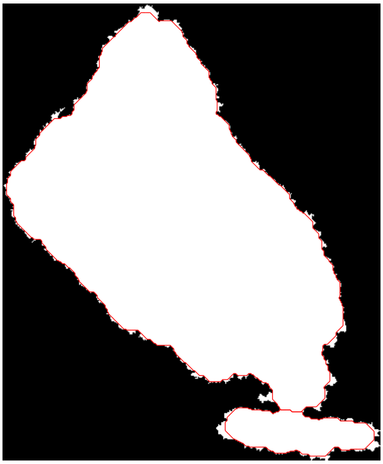


Figure C.4: The segment after separation to two new smaller segments. The red lines indicates where the border of the separated segments are.

2.3 Size Estimation

To estimate the size of the stones use the method described in [5], slightly adjusted, to find the rectangle that fit the segments best. The best-fit rectangle is the rectangle with the smallest width that the segments fit into.

The method consists of two main steps, first the orientation of the segment is estimated using the least-second moment method. Then the rectangle is fit using the Multiple Ferret method.

The idea to find the orientation is to find the line through the object that minimizes the total moment. We want to minimize the integral

$$E = \iint_I R^2 \, dx \, dy, \tag{C.4}$$

where R is the perpendicular distance from the point (x, y) to the line we seek for and I is the integration area, the segment we want to find the orientation of.

A segment with the line giving the least moment is shown in Figure C.5.

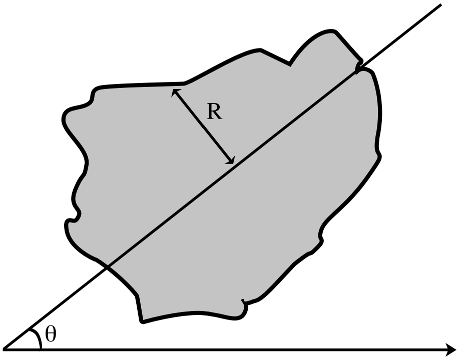


Figure C.5: The main rotation axis.

The integral in equation C.4 can also, with θ as shown in Figure C.5, be written as

$$E = \frac{1}{2}(I_x + I_y) + \frac{1}{2}(I_x + I_y)\cos 2\theta - \frac{1}{2}\sin 2\theta \tag{C.5}$$

$$= I_y \sin^2\theta - I_{xy} sin\theta \cos\theta + I_x \cos^2\theta, \tag{C.6}$$

where $I_x = \iint_{I'} (x')^2\, dx'\, dy'$, $I_{xy} = \iint_{I'} (x'y')\, dx'\, dy'$, $I_y = \iint_{I'} (y')^2\, dx'\, dy'$. By moving the center of the segment, $(\bar{x}, \bar{y})$, to the origin we obtain the new coordinates $x' = x - \bar{x}$ and $y' = y - \bar{y}$.

Minimizing E then gives:

$$\sin 2\theta = \frac{I_{xy}}{\sqrt{I_{xy}^2 + (I_x - I_y)^2}} \tag{C.7}$$

$$\cos 2\theta = \frac{I_y - I_x}{\sqrt{I_{xy}^2 + (I_x - I_y)^2}} \tag{C.8}$$

Knowing $\sin 2\theta$ and $\cos 2\theta$ we can easily compute θ and in that way get the orientation of the segment.

If we know the angle of the main rotation axis in the figure we can find a box around the segment by using dot products. We start by extracting all boundary points of the segment. For all these boundary points, $\boldsymbol{x} = (x, y)$, we calculate the dot product between the point vector and an vector, $\boldsymbol{u}$, which is of unit length and at an angle θ from the x-axis. This vector is simply

$$\boldsymbol{u} = (\cos\theta, \sin\theta), \tag{C.9}$$

where θ is the angle for the rotation axis as before.

By taking the dot products between a point vector, $\boldsymbol{x}$, and the vector $\boldsymbol{u}$ we get the length of the orthogonal projection of $\boldsymbol{x}$ on $\boldsymbol{u}$. If we save the points that gives the highest and lowest value of the dot product we get the points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ shown in Figure C.6. The points together with the direction v gives us the two lines l_1 and l_2 that both are tangents to the segment. The vector v is perpendicular to $\boldsymbol{u}$ and given by $\boldsymbol{v} = (-\sin\theta, \cos\theta)$.

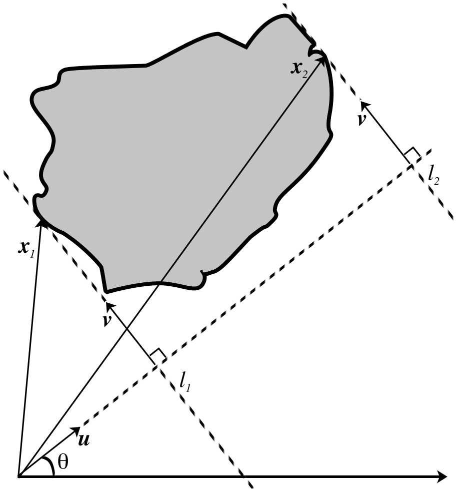


Figure C.6: The points that gives the smallest and largest dot product between itself and u. The vector v is perpendicular to u and shows the direction of the lines l_1 and l_2.

In the same manner we can calculate the dot product between the boundary points and the vector $\boldsymbol{v}$, find the maximum and the minimum and receive the lines l_3 and l_4, shown in Figure C.7. The corners of the rectangle are then achieved by finding the four intersections of the lines.

The orientation of the segment that we get from minimizing the total moment will just give us an approximative estimation of the angle of the rectangle. To find the rectangle that fit the segments the best we try different angles around

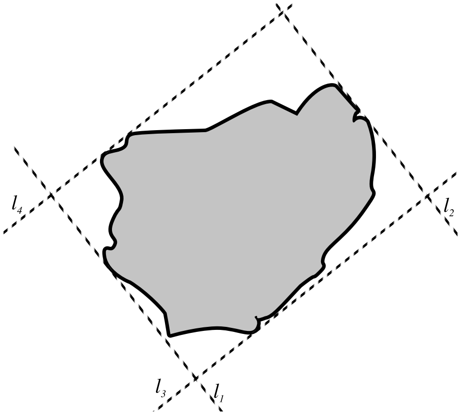


Figure C.7: The four lines that surround the segment.

θ, calculate a rectangle for all these angles as described, and choose the one with the smallest width. Figure C.8 shows the best fit rectangle for some segments, the red lines shows the rectangles using the angle estimated by the total moment method and the cyan colored lines shows the rectangle after optimization.

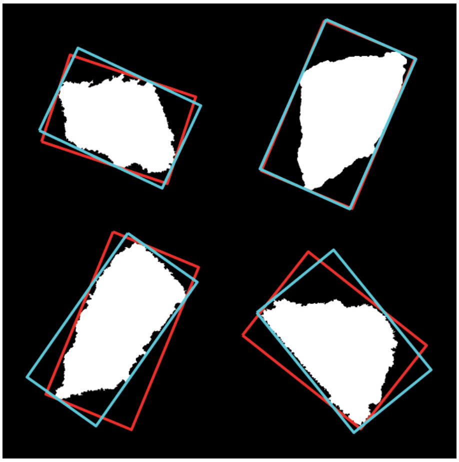


Figure C.8: Four segments with the best fit rectangle in cyan colored lines and the rectangle obtained with the angle estimated by the total moment method shown in red lines.

3 Results

Figure C.9 shows an asphalt sample with the best-fit rectangle for the detected stones in cyan-colored lines. The bottom image shows a close up of some part of the top image. By looking in the image we see that the boxes seems to fit what we would expect is the stones in the sample quite well. All the visible stones in this

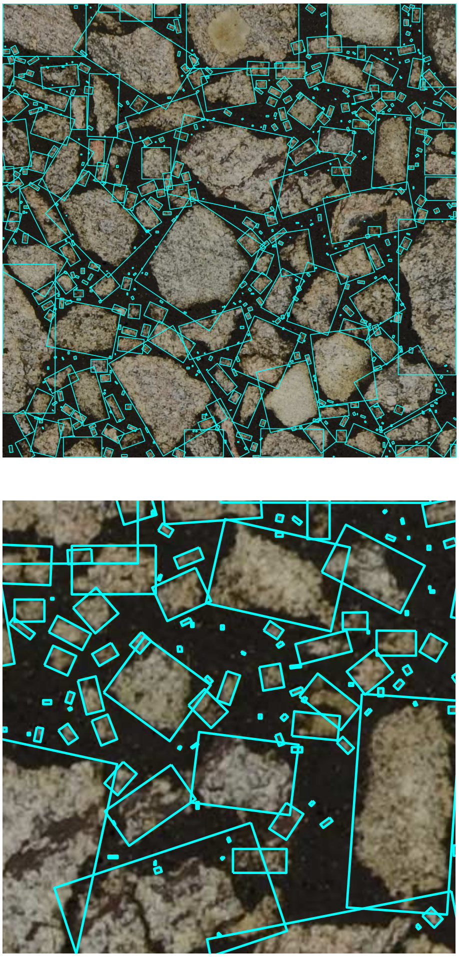


Figure C.9: Original image with the best-fit rectangle marked with cyan colored lines, the bottom image is a close up of a part of the top image.

sample also looks to be surrounded by a box indicating that the method works well to find the stones in the two dimensional image. The size distribution of the grains can be seen in Figure C.10. On the y-axis is the percentage of stone mass passing the sieve for different sieve sizes on the x-axis.

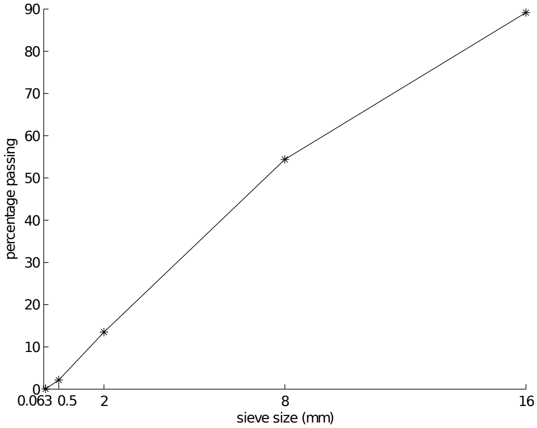


Figure C.10: Size distribution of the stones in the asphalt sample. The sieve size in millimeters is on the x-axis and the percentage passing at a given sieve size is on the y-axis.

4 Conclusions and Future Work

The method shows promising results since the rectangles fit the grains in a good way. However, no comparison with the real size distribution has been done yet.

References

[1] Leonardo Bruno, Giuseppe Parla, and Clara Celauro. Image analysis for detecting aggregate gradation in asphalt mixture from planar images. *Construction and Building Materials*, 28(1):21–30, 2012.

[2] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):pp. 1591–1595, 1996.

[3] J.A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1999.

[4] Morteza Vadood, Majid Safar Johari, and Ali Reza Rahaei. Introducing a simple method to determine aggregate gradation of hot mix asphalt using image processing. *International Journal of Pavement Engineering*, 0(0):1–9, 0.

[5] Weixing Wang. Image analysis of particles by modified ferret method âĂŤ best-fit rectangle. *Powder Technology*, 165(1):1 – 10, 2006.