Charles Yang

7 January 2021

Problem Statement

From a microscopy image, determine where flakes are and infer their thickness.

Problem Statement

From a microscopy image, determine where flakes are and infer their thickness.

Subproblems:

- The intensity of the flakes is very similar to the intensity of the background.

Problem Statement

From a microscopy image, determine where flakes are and infer their thickness.

Subproblems:

- The intensity of the flakes is very similar to the intensity of the background.
- The intensity of sensor noise is on the order of flake intensity.

Problem Statement

From a microscopy image, determine where flakes are and infer their thickness.

Subproblems:

- The intensity of the flakes is very similar to the intensity of the background.
- The intensity of sensor noise is on the order of flake intensity.
- Determining the location of flakes and distinguishing them from other objects.

General Algorithm

Subproblems:

The intensity of the flakes is very similar to the intensity of the background.

The intensity of sensor noise is on the order of flake intensity.

Determining the location of flakes and distinguishing them from other objects.

General Algorithm

Subproblems:

- The intensity of the flakes is very similar to the intensity of the background.
  - Flattening and Contrast increase
- The intensity of sensor noise is on the order of flake intensity.

- Determining the location of flakes and distinguishing them from other objects.

General Algorithm

Subproblems:

- The intensity of the flakes is very similar to the intensity of the background.
  - Flattening and Contrast increase
- The intensity of sensor noise is on the order of flake intensity.
  - Blurring and Morphological transformations
- Determining the location of flakes and distinguishing them from other objects.

General Algorithm

Subproblems:

- The intensity of the flakes is very similar to the intensity of the background.
  - Flattening and Contrast increase
- The intensity of sensor noise is on the order of flake intensity.
  - Blurring and Morphological transformations
- Determining the location of flakes and distinguishing them from other objects.
  - Edge Detection, Contouring, and Transmission Measurement (partial)

Flattening

Assuming elliptical symmetry, we define a distance metric

$$s^2 = (x - x_c)^2 + r^2(y - y_c)^2$$

Flattening

Assuming elliptical symmetry, we define a distance metric

$$s^2 = (x - x_c)^2 + r^2(y - y_c)^2$$

Then, we claim that we can approximate the background as

$$B(s) = \sum_n a_n s^{2n}$$

Flattening

Assuming elliptical symmetry, we define a distance metric

$$s^2 = (x - x_c)^2 + r^2(y - y_c)^2$$

Then, we claim that we can approximate the background as

$$B(s) = \sum_n a_n s^{2n}$$

Sampling multiple points, we then obtain a system of linear equations in a_n

$$B_i = \sum_n a_n s_i^{2n}$$

Flattening

Rewrite the previous as matrix equation

$$B = SA$$

with the vector of background values B_i being measured at points s_i to generate $S_{ij} = s_i^{2(j-1)}$, with coefficients $A_n = a_n$

Flattening

Rewrite the previous as matrix equation

$$B = SA$$

with the vector of background values B_i being measured at points s_i to generate $S_{ij} = s_i^{2(j-1)}$, with coefficients $A_n = a_n$
We strategically chose s_i to make computation easy:

$$s_i^2 = (i+1) \cdot s^2$$

Flattening

Performing row reduction on S, we obtain a nice pattern:

$$[S'|I] = \left[\begin{array}{ccccc|ccccc} 1 & 1 & 1 & 1 & \cdots & 1 & 0 & 0 & 0 & \cdots \\ 1 & 2 & 4 & 8 & \cdots & 0 & 1 & 0 & 0 & \cdots \\ 1 & 3 & 9 & 27 & \cdots & 0 & 0 & 1 & 0 & \cdots \\ 1 & 4 & 16 & 64 & \cdots & 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right]$$

$$\downarrow$$

$$[T|C] = \left[\begin{array}{ccccc|ccccc} 1 & 1 & 1 & 1 & \cdots & 1 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 3 & 7 & \cdots & -1 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 2 & 12 & \cdots & 1 & -2 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 6 & \cdots & -1 & 3 & -3 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right]$$

This pattern was calculated and verified to hold until at least $n = 10$

Flattening

Thus, we can obtain A by instead solving the equation

$$CB = CSA$$

using back substitution.

Flattening

Thus, we can obtain A by instead solving the equation

$$CB = CSA$$

using back substitution.

Once A is determined, the baseline approximation can be computed recursively as

$$f_0 = a_N$$

$$f_n = s^2 f_{n-1} + a_{N-n}$$

$$B_N(s) = f_N$$

This baseline is then subtracted from the total image.

Flattening Results

Blurring and Morphological Tranforms

To reduce noise, first a gaussian blur was used to "average out" the initial image, before the process of flattening.

Blurring and Morphological Tranforms

To reduce noise, first a gaussian blur was used to "average out" the initial image, before the process of flattening.

After flattening, the noise was removed using a combination of the morphological transformations Open, Close, Erode, Dilate included in the OpenCV API.

Blurring and Morphological Tranforms

To reduce noise, first a gaussian blur was used to "average out" the initial image, before the process of flattening.

After flattening, the noise was removed using a combination of the morphological transformations Open, Close, Erode, Dilate included in the OpenCV API.

Erosion returns 1 only if all pixels in the kernel are 1

Blurring and Morphological Tranforms

To reduce noise, first a gaussian blur was used to "average out" the initial image, before the process of flattening.

After flattening, the noise was removed using a combination of the morphological transformations Open, Close, Erode, Dilate included in the OpenCV API.

- Erosion returns 1 only if all pixels in the kernel are 1
- Dilation returns 1 if any pixel in the kernel is 1

Blurring and Morphological Tranforms

To reduce noise, first a gaussian blur was used to “average out” the initial image, before the process of flattening.

After flattening, the noise was removed using a combination of the morphological transformations Open, Close, Erode, Dilate included in the OpenCV API.

- Erosion returns 1 only if all pixels in the kernel are 1
- Dilation returns 1 if any pixel in the kernel is 1
- Opening and Closure are both compositions of the Erosion and Dilation

Blurring and Morphological Tranforms

To reduce noise, first a gaussian blur was used to "average out" the initial image, before the process of flattening.

After flattening, the noise was removed using a combination of the morphological transformations Open, Close, Erode, Dilate included in the OpenCV API.

- Erosion returns 1 only if all pixels in the kernel are 1
- Dilation returns 1 if any pixel in the kernel is 1
- Opening and Closure are both compositions of the Erosion and Dilation

The morphological transforms were determined by trial and error, adjusting the order of transformation and size of the kernel.

Denoising Results

Edge Detection, Contouring, and Transmission Measurement (partial)

Edge detection is can be done using a gradient method, such as Sobel Derivatives and the Laplacian, to detect where the intensity of an image changes quickly—an "edge"

Edge Detection, Contouring, and Transmission Measurement (partial)

- Edge detection is can be done using a gradient method, such as Sobel Derivatives and the Laplacian, to detect where the intensity of an image changes quickly—an "edge"
- After the edge is detected, the *contour* method of OpenCV generates sets of isolines—contours—as the sets of boundaries

Edge Detection, Contouring, and Transmission Measurement (partial)

- Edge detection is can be done using a gradient method, such as Sobel Derivatives and the Laplacian, to detect where the intensity of an image changes quickly—an "edge"
- After the edge is detected, the *contour* method of OpenCV generates sets of isolines—contours—as the sets of boundaries
- The flakes can be inferred to be regions contained within closed boundaries.

Edge Detection, Contouring, and Transmission Measurement (partial)

- Edge detection is can be done using a gradient method, such as Sobel Derivatives and the Laplacian, to detect where the intensity of an image changes quickly—an "edge"
- After the edge is detected, the *contour* method of OpenCV generates sets of isolines—contours—as the sets of boundaries
- The flakes can be inferred to be regions contained within closed boundaries.
- The flakes can be filtered from noise and other objects by measuring the transmission relative to the approximated baseline.

Edge Detection, Contouring, and Transmission Measurement (partial)

- Edge detection is can be done using a gradient method, such as Sobel Derivatives and the Laplacian, to detect where the intensity of an image changes quickly—an “edge”
- After the edge is detected, the *contour* method of OpenCV generates sets of isolines—contours—as the sets of boundaries
- The flakes can be inferred to be regions contained within closed boundaries.
- The flakes can be filtered from noise and other objects by measuring the transmission relative to the approximated baseline.
- While I have proposed these methods, I haven’t been able to test them too much.

Flake Determination Results

Issues and Directions for Future Work

Background isn't perfectly centred or uniform as assumed.

Issues and Directions for Future Work

Background isn't perfectly centred or uniform as assumed.

Very low signal to noise ratio—Image stacking?

Issues and Directions for Future Work

Background isn't perfectly centred or uniform as assumed.

Very low signal to noise ratio—Image stacking?

Many of the transformations need specified kernel sizes—introduction of "magic numbers"

Issues and Directions for Future Work

Background isn't perfectly centred or uniform as assumed.

Very low signal to noise ratio—Image stacking?

Many of the transformations need specified kernel sizes—introduction of "magic numbers"

The latter half of the program could not be implemented due to issues with the flattening algorithm.

Issues and Directions for Future Work

Background isn't perfectly centred or uniform as assumed.

Very low signal to noise ratio—Image stacking?

Many of the transformations need specified kernel sizes—introduction of "magic numbers"

The latter half of the program could not be implemented due to issues with the flattening algorithm.

Alternative colour spaces may be better suited for analysis than the default BGR colourspace.

Code

https://github.com/daedalus1235/FlakeAutoFind.git (private repo)

Written in C++ using OpenCV, compiled with CMake and g++.