Classifying Colon Cancer Colonoscopy Images Using Edge Histograms

Samy Dafir
Dominik Baumgartner
Sebastian Strumegger

17. Januar 2017

Content

Task - Overview

What?

- Colon cancer colonoscopy images
- Edge histograms
- KNN: K Nearest Neighbors - Classification

Task - Overview

How?

1. Preprocess images
2. Perform edge detection
3. Extract features (e.g. edge lengths)
4. Compute Edge Histogram
5. Classify with KNN
6. Analyze the results

Overview

- What are Edges
- Edge Detection
- First Derivative
- Second Derivative
- Canny Edge Detection

Edges:

Edges are pixels, in which the image intensity function changes its magnitude

(a) Original Image (b) Image after Edge Detection

Abbildung: Edge Detection using Canny

Edge Detection:

Almost every Edge Detector uses either the first derivative or the second derivative of the intensity function.

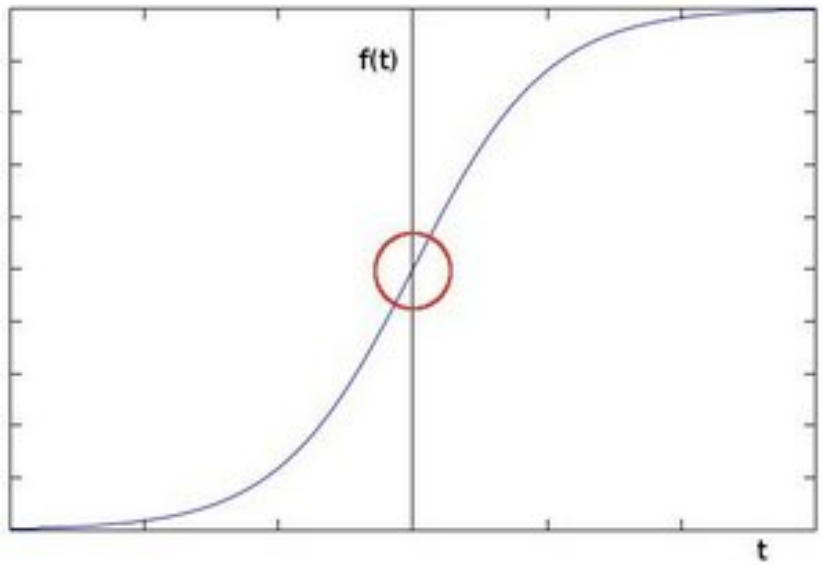


Abbildung: Intensity function

First Derivative:

Sobel-, Roberts-, Robinson-, Kirsch-Operator

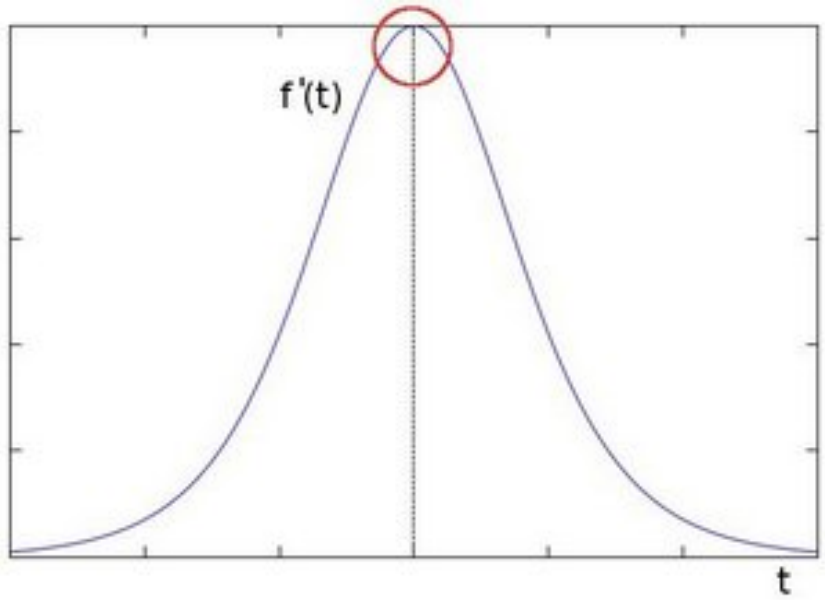


Abbildung: Intensity function - First derivative

Second Derivative:

Laplace-, Mexican-Hat-Operator

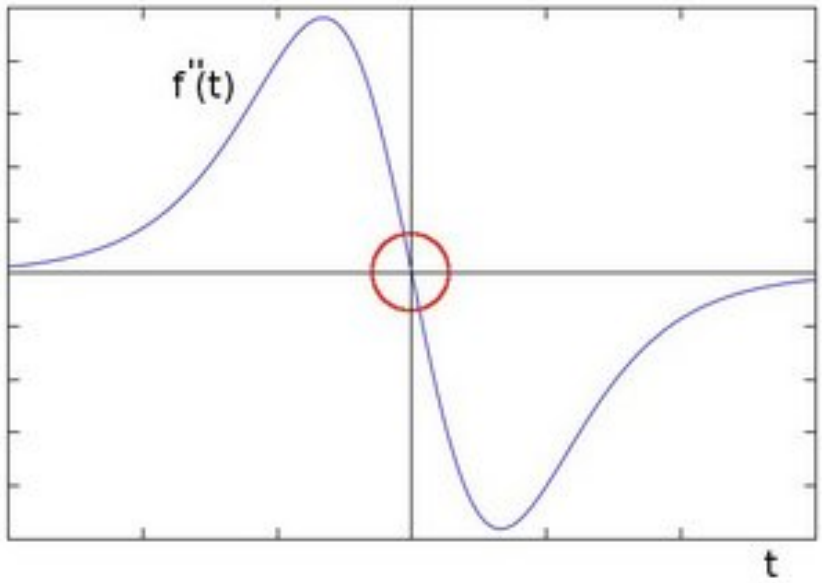


Abbildung: Intensity function - Second derivative

Canny Edge Detection:

- Low error rate
- Good localization
- Minimal response

Steps:

1. Filter out noise using Gaussian filter
2. Find the intensity gradient using Sobel-Operator
   $G = \sqrt{G_x^2 + G_y^2}$ or $G = |G_x| + |G_y|$
3. Non-maximum suppression
4. Hysteresis

Overview

- Development and Frameworks
- Image Enhancement
- Edge Detection
- Histograms
- Edge Lengths
- Edge Orientation
- Image Classification

Development and Frameworks

Developed using:

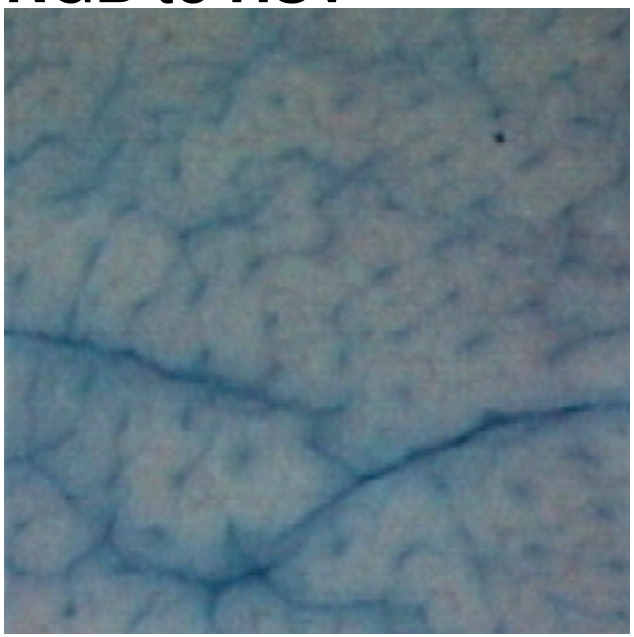
- Java 8
- OpenCV for Java
- Eclipse Neon

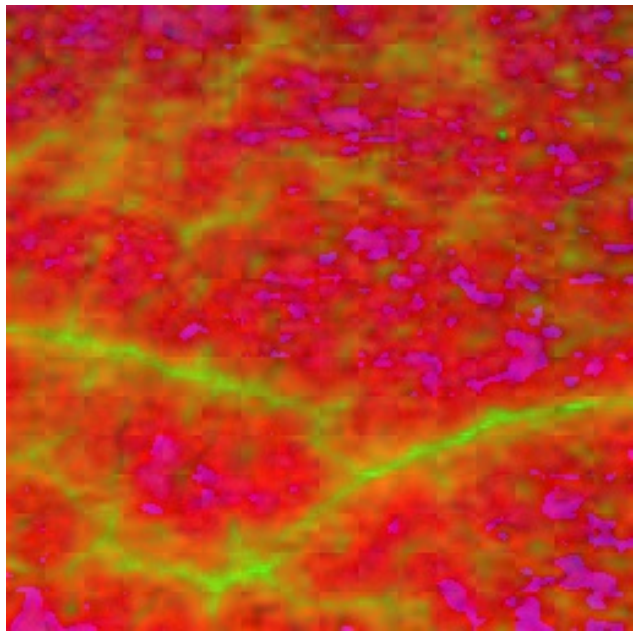
Image Enhancement and Conversion

- Color Space Conversion
- Normalization
- CLAHE
- All part of OpenCV

Image Enhancement and Conversion

RGB to HSV





```
Imgproc.cvtColor(matrix, matrix, colorSpace)
colorSpace: e.g. Imgproc.COLOR_RGB2HSV
```

Image Enhancement and Conversion

Normalization:

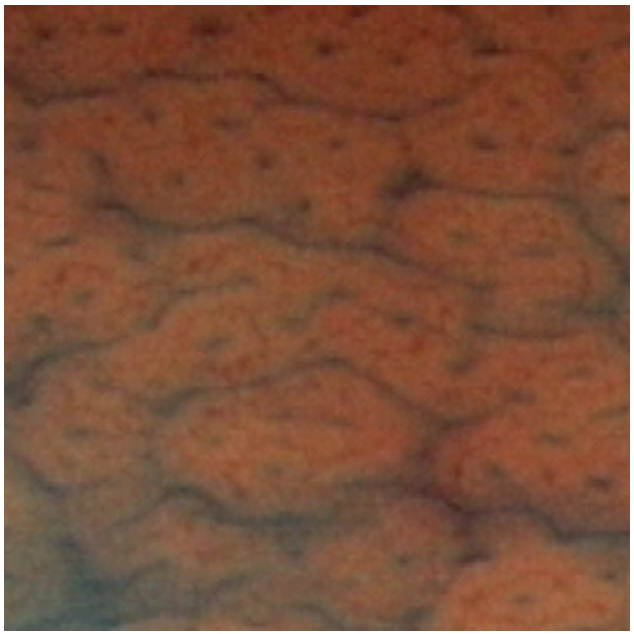
```
Core.normalize(matrix,matrix,255,0,
 Core.NORM_MINMAX);
```

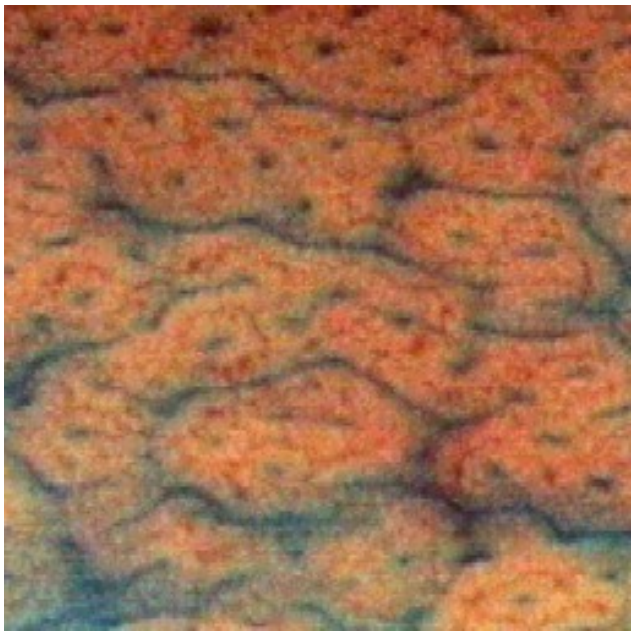
CLAHE:

```
Mat channel = new Mat();
Core.extractChannel(matrix, channel, i);
CLAHE clahe = Imgproc.createCLAHE();
clahe.apply(channel, channel);
Core.insertChannel(channel, matrix, i);
```

Image Enhancement and Conversion

Image Enhancement and Conversion

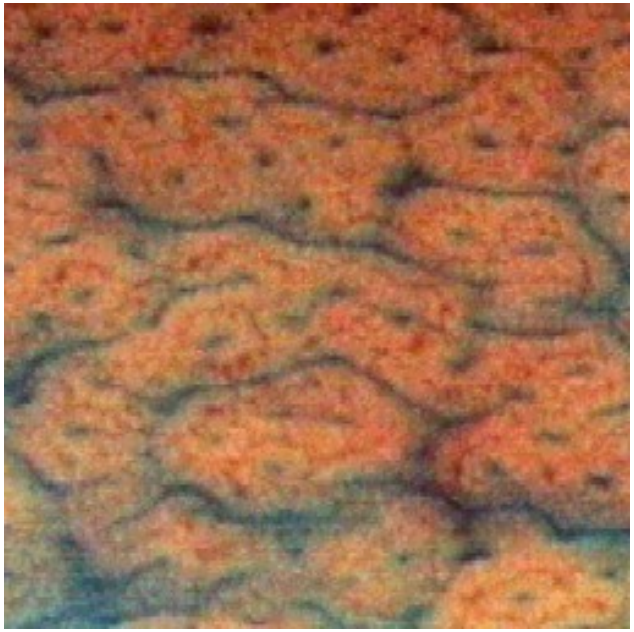


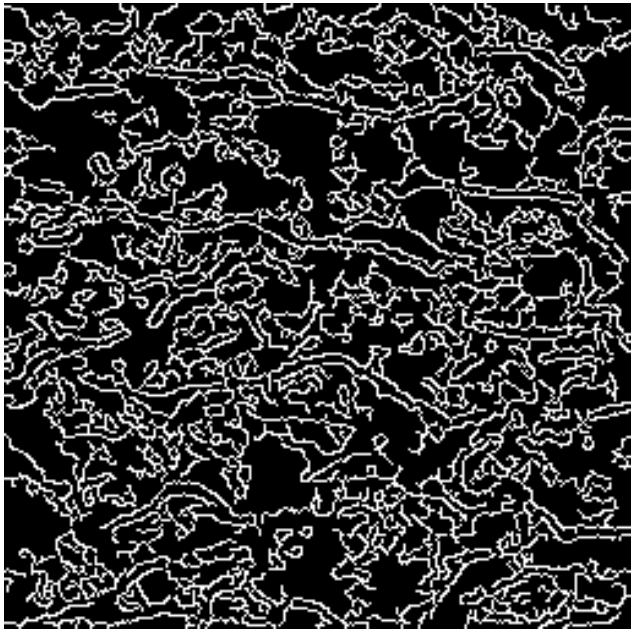


Edge Detection

- Grayscale Conversion
- Canny Edge Detector

Edge Detection





```
Imgproc.Canny(matrix, matrix,
 lowThresh, highThresh);
```

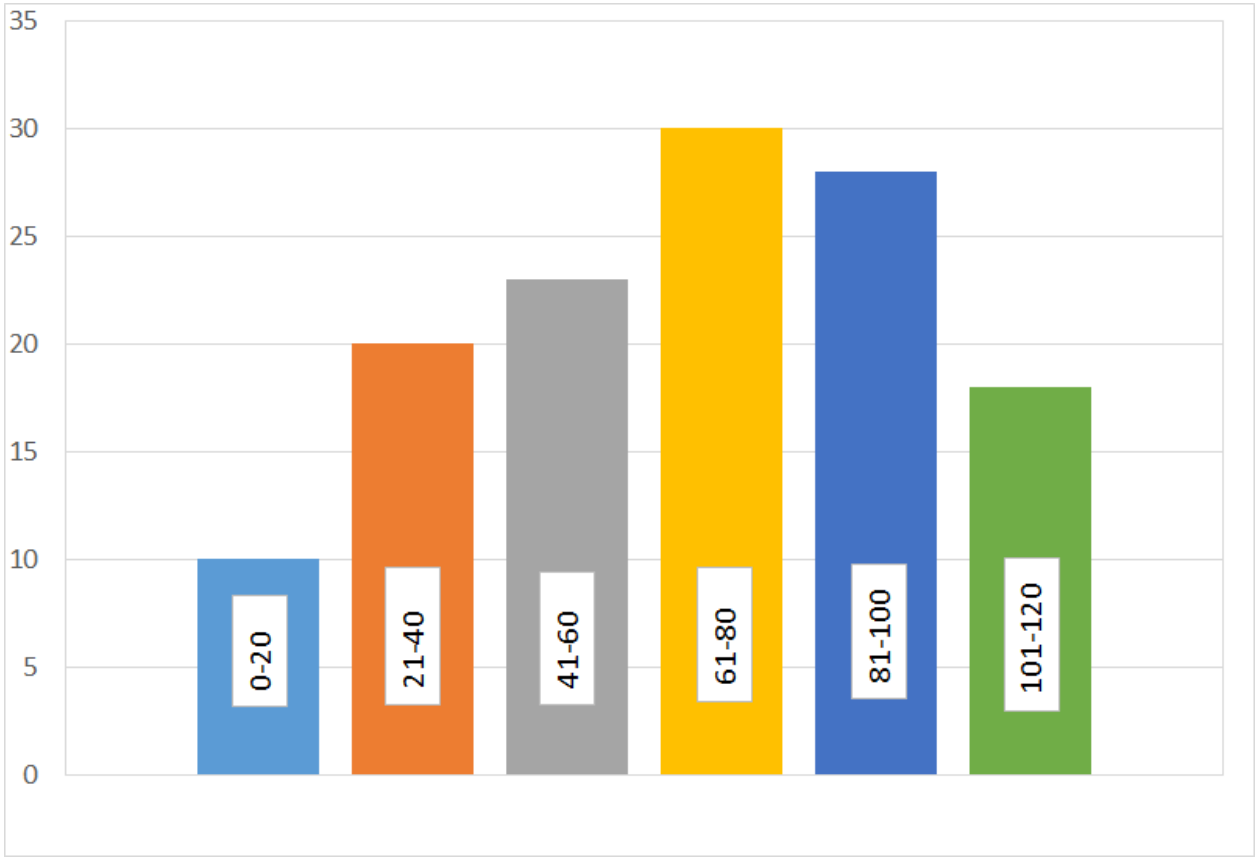
Edge Histograms

Definition

- Characteristics of an image e.g. edge lengths
- Partition characteristic attributes into bins
- In our case: Edge lengths & orientations
- Length: Image has
  5 edges of length 0 - 20 pixels
  20 edges of length 100 - 120 pixels

Edge Histograms

Simple Example:
10.0, 20.0, 23.0, 30.0, 28.0, 18.0, 1



Edge Histograms

- Histogram data for each example image collected in a hist-file.
- Specify Category. Here: Cancer stage
- Example:

```
210.0,3.0,170.0,142.0,126.0,93.0,32.0,16.0,1
192.0,2.0,181.0,139.0,119.0,87.0,32.0,17.0,1
143.0,1.0,172.0,147.0,128.0,91.0,30.0,16.0,1
```

Edge Lengths

Prequisites

- Image with detected Edges
- Edges white
- Rest black
- Not entirely given $\rightarrow$ Threshold set at grayscale 200

Edge Lengths

Algorithm

Iterate over all pixels

1. Check if pixel is white $\rightarrow$ new edge found
2. Check immediate neighbours: if white $\rightarrow$ add pixel to edge
3. Follow white path until no more connected white pixels
4. Add all passes pixels to Collection of used pixels
5. Add one to category with detected length
6. Continue iterating and start at 1

Edge Lengths

Algorithm

```
function measureEdge(pixel) {
    length = 1
    for each surrounding pixel p:
        if p == white
            length += measureEdge(p)

    return length
}
```

Edge Orientation

- Requires edge detected image
- Use sobel operator to detect edges
- Extract edge orientations from image (OpenCV)
- Partition edges into bins
- Bin content: pixels part of edge with certain orientation
- Category: range of angles

Image Classification

1. Classify all example images $\rightarrow$ create file with histograms
2. Create List of Feature Vectors
3. Enhance image to classify
4. Create Feature Vector
5. KNN: Compare new vector all vectors in list
   $\rightarrow$ euclidean distance
6. Select K vectors with smallest distance
7. Classification: category found most often

Paramter Optimization

Difference between a good and barely functional program

- Selection of input images
- Thresholds for edge detection
- Edge Lengths: number of bins, range of lengths in bins
- Weights of features: all equally significant
  Lengths: per edge
  Orientation: per pixel

Problems

Encountered Problems

- Fine tuning of parameters

Results

Colormodels compared

- enhanced RGB
- HSV
- Twice enhanced RGB

Results

Edgelength vs Edge Orientation vs both