TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

“Image analysis to automated drill hole logging”

Georgii Burdin – 66259

General data review and definition of the terms of reference

Given: An array of images of drilling rock samples

Task: Get description of the core samples

Rock Description	Qz vein material. Qz mosaics. Unknown acicular mineral. Severan unknown hypogene and supergene Cu sulfide									
Specimen #	**Mineral**	**Texture**	**Shape**	**Grain Size**	**Replacement**	**%**	**Associated Minerals**	**Observations**	**Metal**	**Texture**
CHA25-275.7	Pg		subh	fg		7			Py	
10107402	Qz		anh	fg		5			Cp	
2012	Ca		anh	vfg		3			Sph	
GGG	Gy		euh	fg		1	Ca, Ct, sulfides	vnlet	Ga	
	Se		anh	vfg		65	Pg, matrix		Bn? Orange	
	clay?					5			Bn? Purple	
	glass?					2		shards	Cc?	
	Ct		anh	vfg		2	Gy, Ca, sulfides	vnlet		

Overview of results

Algorithms have been developed that solve the necessary tasks to prepare a dataset of rock sample descriptions:

- obtaining the properties of image segments
- Several ways to localize and extract the segments of interest during preprocessing were proposed
- Main segmentation algorithms were tested
- The key problem to be solved for further development is outlined.

The created algorithms can be easily combined in a package or used separated in other projects

Standard OpenCV sklearn.cluster .KMeans and skimage libraries are used

https://colab.research.google.com/gist/geoburdin/606dbd1285f5ba1f8063cc87685afc49/segmentation_demonstration.ipynb

Structure of the solution

Preprocessing

- Image normalization
- Defining the image analysis region
- (optional) Localization

Segmentation

- Choice of segmentation algorithm

Merging

- Similar segment integration

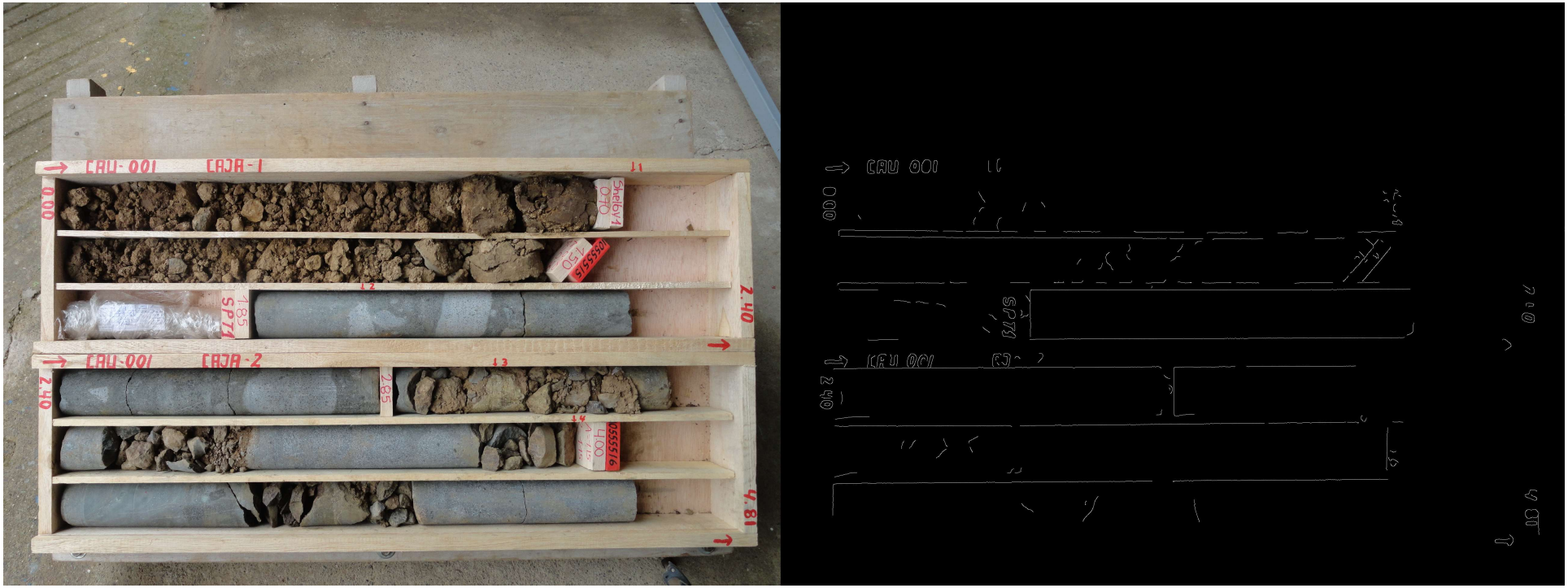
Feature extraction

- Choosing the necessary properties
- Obtaining properties using standard methods
- Obtaining properties by a self-written method

Canny edges

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

Canny edges



Localization using the contour method

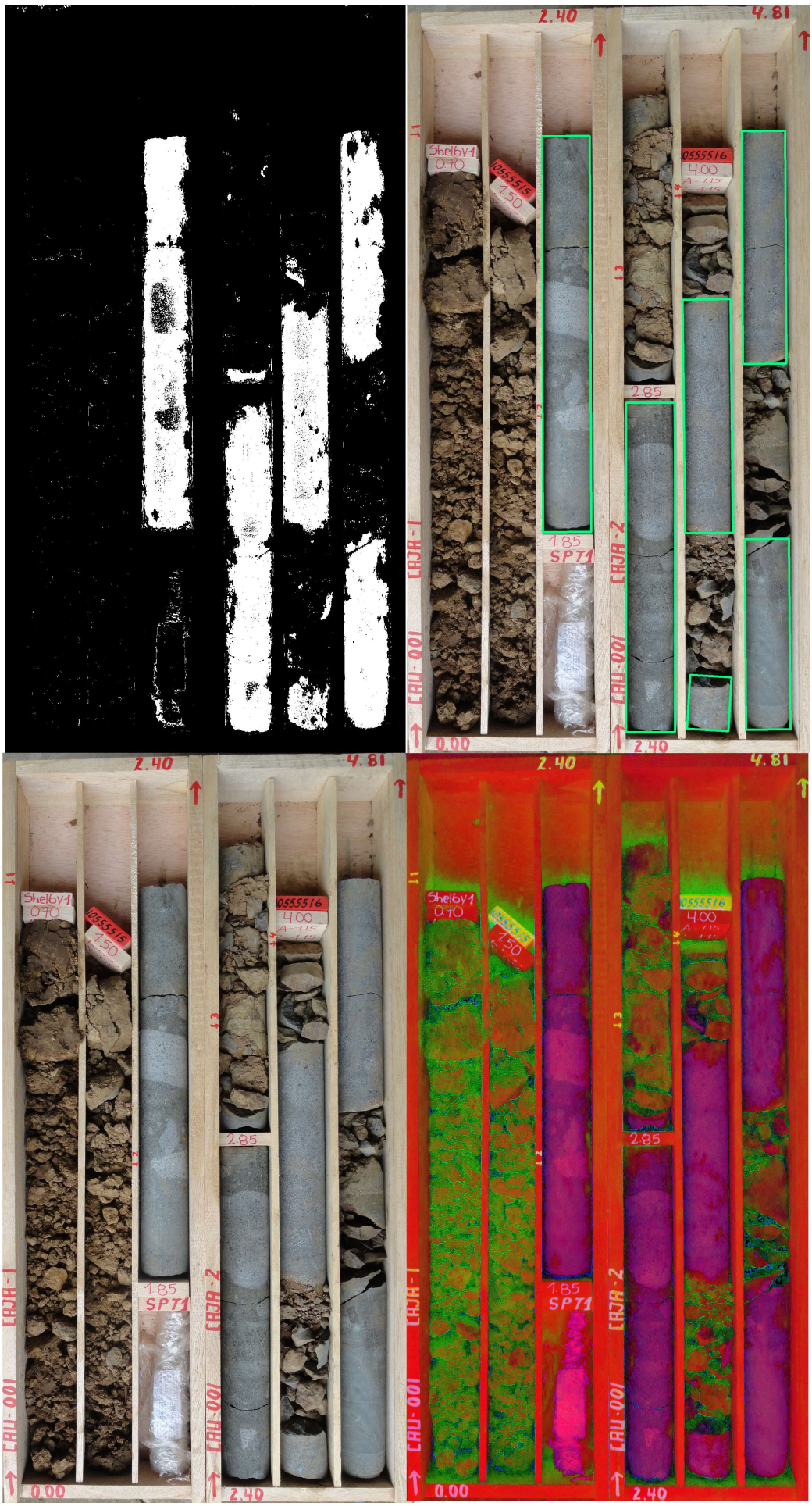
Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.
•For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.

•In OpenCV, finding contours is like finding white object from black background. So remember, object to be found should be white and background should be black.

1. Convert to HSV
2. Threshold
3. Find the contours

```
contours0, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

4. Filter the contours by area

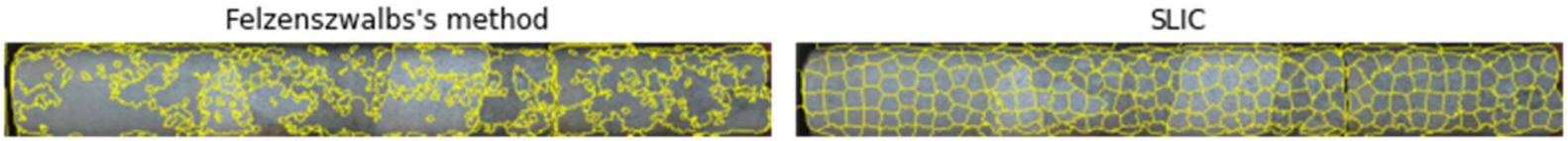

2.40
4.81
Shelby1
0.70
0555515
1.50
1.85
SPT1
0555516
4.00
2.85
CAJA-1
CAJA-2
CAU-001
0.00
2.40

Segmentation

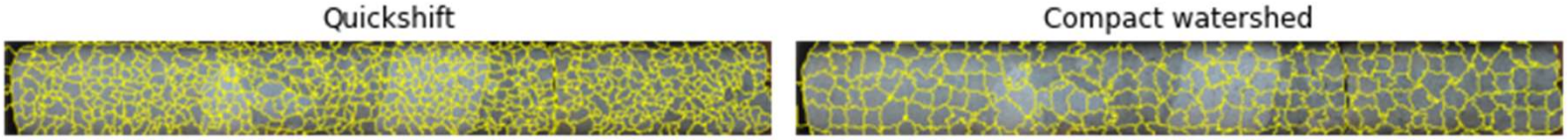
In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

```
segments = skimage.segmentation.felzenszwalb(img, min_size=500)
```

Comparison of segmentation algorithms



This example compares four popular low-level image segmentation methods. As it is difficult to obtain good segmentations, and the definition of “good” often depends on the application, these methods are usually used for obtaining an oversegmentation, also known as superpixels. These superpixels then serve as a basis for more sophisticated algorithms such as conditional random fields (CRF).



FELZENSZWALB'S EFFICIENT GRAPH BASED SEGMENTATION

This fast 2D image segmentation algorithm, proposed in 1 is popular in the computer vision community. The algorithm has a single scale parameter that influences the segment size. The actual size and number of segments can vary greatly, depending on local contrast.

QUICKSHIFT IMAGE SEGMENTATION

Quickshift is a relatively recent 2D image segmentation algorithm, based on an approximation of kernelized mean-shift. Therefore it belongs to the family of local mode-seeking algorithms and is applied to the 5D space consisting of color information and image location 2.
One of the benefits of quickshift is that it actually computes a hierarchical segmentation on multiple scales simultaneously.

SLIC - K-MEANS BASED IMAGE SEGMENTATION

This algorithm simply performs K-means in the 5d space of color information and image location and is therefore closely related to quickshift. As the clustering method is simpler, it is very efficient. It is essential for this algorithm to work in Lab color space to obtain good results. The algorithm quickly gained momentum and is now widely used. See 3 for details. The compactness parameter trades off color-similarity and proximity, as in the case of Quickshift, while n_segments chooses the number of centers for kmeans.

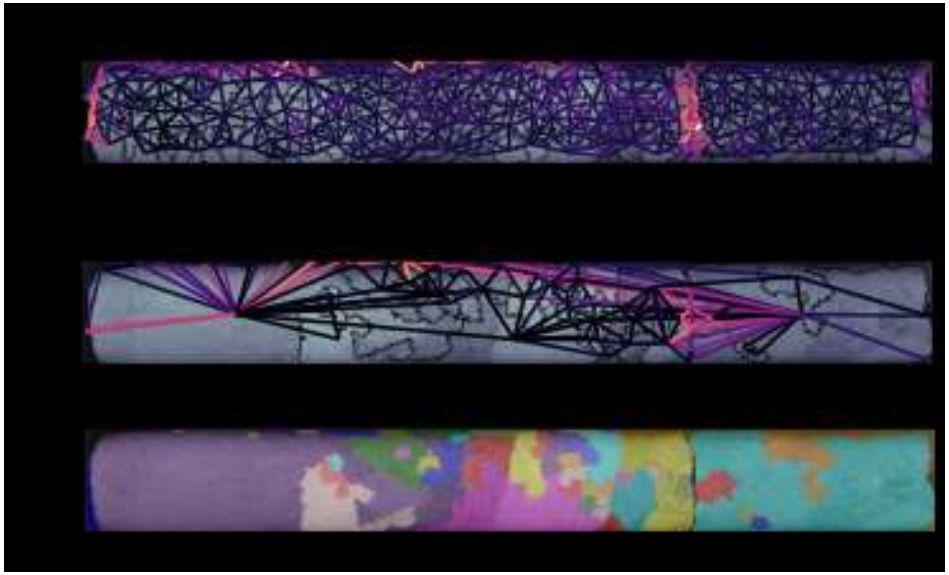
COMPACT WATERSHED SEGMENTATION OF GRADIENT IMAGES

Instead of taking a color image as input, watershed requires a grayscale gradient image, where bright pixels denote a boundary between regions. The algorithm views the image as a landscape, with bright pixels forming high peaks. This landscape is then flooded from the given markers, until separate flood basins meet at the peaks. Each distinct basin then forms a different image segment.

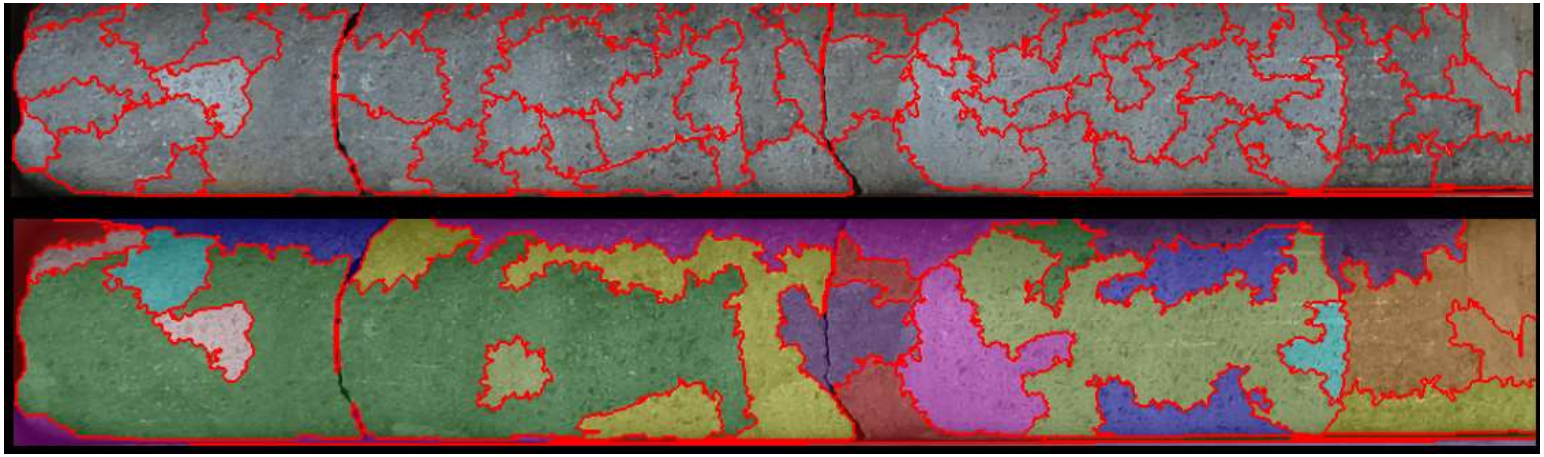
RAG merging

After image segmentation, the resulting segments are often not very different from each other in characteristics, so it would be useful to create a super-segment combining several segments obviously similar by a key criterion

```
g = graph.rag_mean_color(img, segments)
labels2 = graph.merge_hierarchical(segments, g,
                                   thresh=20,
                                   rag_copy=False,
                                    in_place_merge=True,
                              merge_func=merge_mean_color,
                              weight_func=_weight_mean_color)
```



This example constructs a Region Adjacency Graph (RAG) and progressively merges regions that are similar in color. Merging two adjacent regions produces a new region with all the pixels from the merged regions. Regions are merged until no highly similar region pairs remain.



Getting features

Selection of specific features that may be needed in the future

I took the following properties from the standard library:

centroid, orientation,
major_axis_length, minor_axis_length,
area, mean_intensity, perimeter, eccentricity.

As well as additional property of the dominant colors, the color palette

```
labels2 = labels2+1
props = measure.regionprops_table(label_image = labels2, intensity_image = gray_img,
                                  properties=('centroid‘, 'orientation‘,
                                  'major_axis_length‘, 'minor_axis_length',
                                  'area','mean_intensity', 'perimeter', 'eccentricity'),)
```

label_image : (N, M) ndarray
Labeled input image. Labels with value 0 are ignored.

For anyone who is going to work with this function, it should be noted that segment number 0 is considered a background and therefore is not included in the table

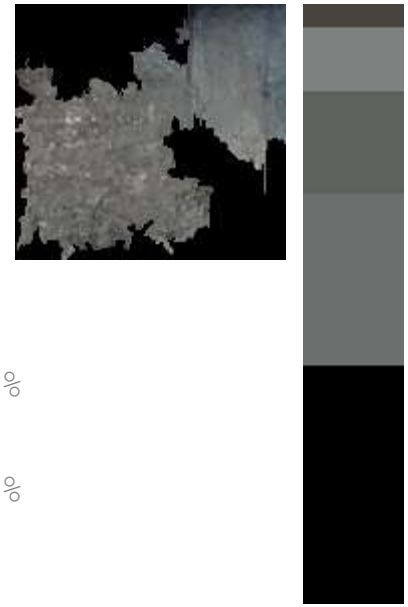
The danger is that on big data it is easy to overlook this and lose some of the data and mix up the rest

This bug\feature has several solutions:

1. Rename labels so that the zero segment does not exist (label = label+1)
2. Using a self-written function

Algorithm dominant colors

In addition to the standard functions, it would be useful to have additional knowledge of what colors a segment contains and how these most common colors are distributed.



```
from sklearn.cluster import Kmeans
cluster = KMeans(n_clusters=5).fit(reshape)
```

```
Result:
[71.60141844 68.34893617 63.39148936] 4.10%
[125.51082251 130.37878788 128.46590909] 10.70%
[94.3429738  96.1531133  94.91629806] 16.99%
[108.44725141 111.77303961 109.89773646] 28.63%
[0.08377193 0.07280702 0.06388889] 39.58%
```

Results

	centroid-0	centroid-1	orientation	major_axis_length	minor_axis_length	area	mean_intensity	perimeter	eccentricity	pallete colours	procent of color
0	17	76	-1,41452	331,3253	83,7866	3097	35,15467	804,3259	0,967497	[[2.83704472e+01 2.68004112e+01 2.72180614e+01] [1.31139284e-04 1.04911436e-04 1.15402574e-04] [8.06455002e+01 7.80952377e+01 7.89312134e+01] [4.72273293e+01 4.38236008e+01 4.30074539e+01] [1.26243423e+02 1.21388161e+02 1.22223686e+02]]	[50.89%, 27.37%, 14.20%, 5.51%, 2.02%]

The output contains a table with segment descriptions and their features

RECOMMENDATIONS

Preprocessing:

- Paint the box in a bright contrasting color. Then we can take the thresholding and draw the contours of the exact rock samples clearly.

Segmentation:

- Depending on the task it is necessary to choose an algorithm and its parameters so as not to lose information and to optimize the time of calculations

Merging:

- This will reduce the area that the rest of the segmentation and RAG algorithms will process, I think will speed up the process.

Feature extraction:

- Together with geologists we need to find the right and meaningful parameters

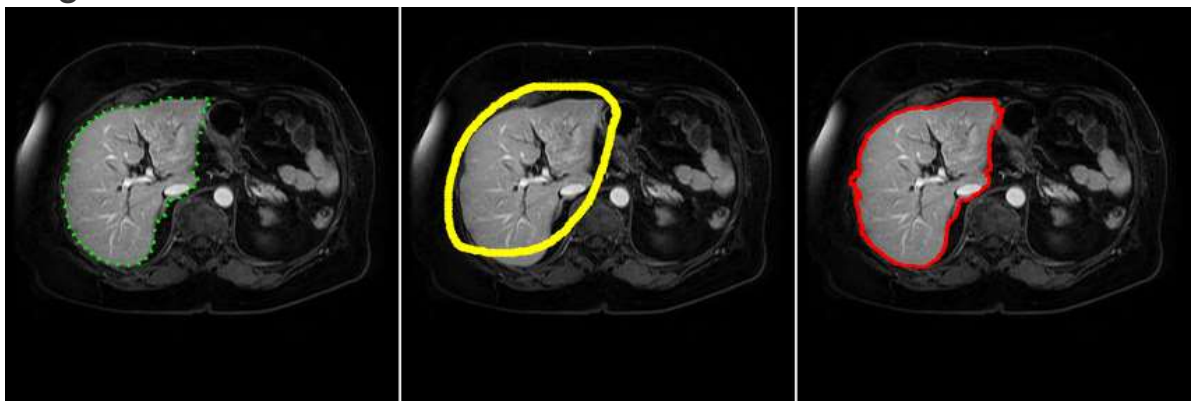
RECOMMENDATIONS

We have several images of the same samples, moreover, in different states, wet, dried and normal.

It would be interesting to crossvalidate the segmentation and combine the analysis of all three samples to give some kind of overall analysis.

RECOMMENDATIONS

- I would replace RAG for segmentation merging with the random walker algorithm.
- Example of segmentation using the random walker with our approach: (left) ground truth image, (middle) rough contour drawn by the user, and (right) segmentation results:



So, the idea is simple:

- get the marked images
- Then train the model based on them
- and the rest of the images would be marked by trained model

How can be used? What to do next?

- This could be useful as a starting point for building an automatic logging system that could detect the presence of certain rocks/minerals during drilling
- As I described before, we need a geologist to show that this image shows, for example, hematite at this location to get a set of labeled data
- Having the necessary dataset, I would recommend building a convolutional neural network, which from my experience is well suited and often used for image analysis

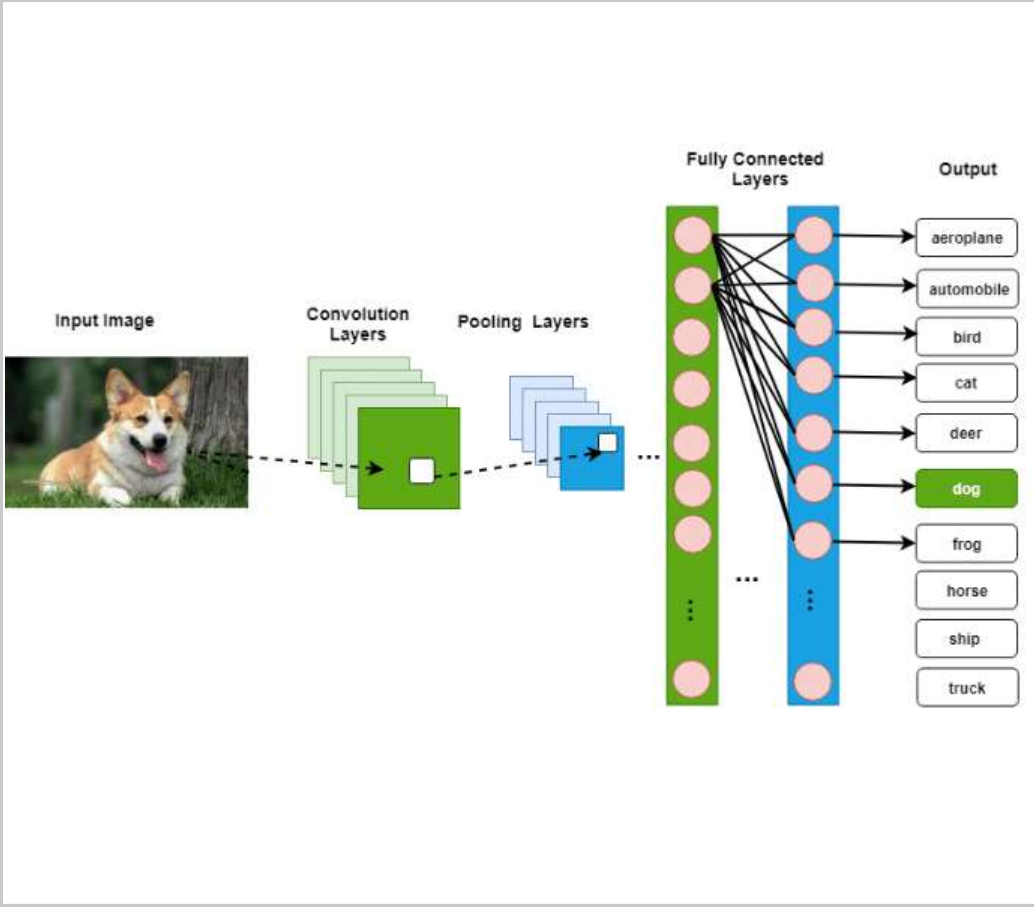
CNN

Convolutional neural networks (CNNs) are very different from other types of networks.

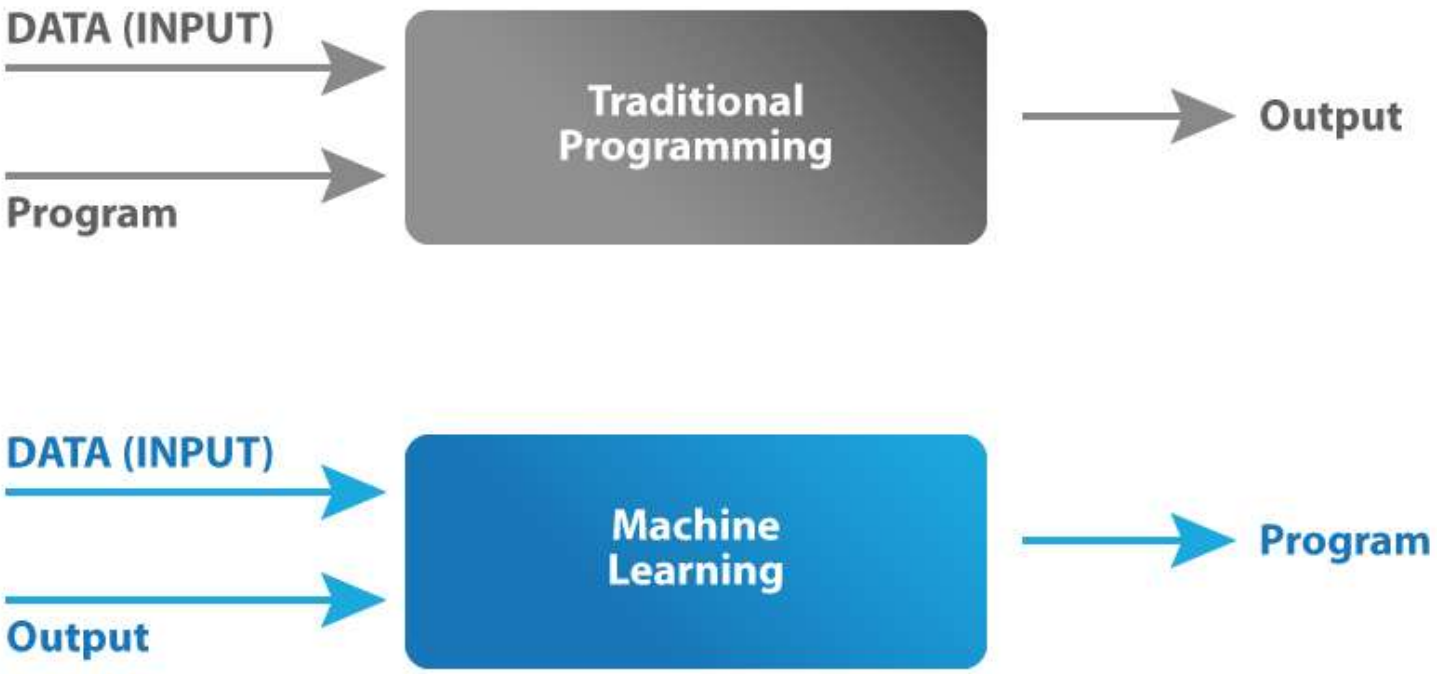
A typical application of CNNs is to classify images: if there is a cat in the image, the network will say "cat"; if there is a dog, it will say "dog". Such networks usually use a "scanner" that doesn't parse all the data at once.

For example, if you have a 200×200 image, you won't process all 40,000 pixels at once. Instead, the network counts a 20 x 20 square (usually from the top left corner), then shifts by 1 pixel and counts a new square, and so on. This input is then passed through convolutional layers, in which not all nodes are connected to each other.

In practice, a FFNN (feed forward neural networks) is attached to the end of the CNN for further data processing. Such networks are called depth networks (DCNN)



Main problem of the ML approach



Sources

Development:

https://docs.opencv.org/4.5.2/index.html

https://scikit-image.org/docs/dev/api/skimage.html

https://github.com/geoburdin/core_samples

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://colab.research.google.com/gist/geoburdin/606dbd1285f5ba1f8063cc87685afc49/segmentation_demonstration.ipynb

Presentation:

https://www.researchgate.net/figure/Examples-of-segmentation-using-the-random-walker-with-our-approach-left-ground-truth_fig9_303321652

https://www.researchgate.net/figure/A-simplified-CNN-architecture-for-dog-Classification_fig1_336304381

https://eduspiral.com/2021/03/04/best-course-study-malaysia-machine-learning/what-is-machine-learning-machine-learning-model-vs-traditional-model/?amp