INTRODUCTION

In archeology analyzing marbles and sandstones are regular tasks. The analysis sometimes aims for determining the provenance of marbles and the quality of sandstones. These attributes can be acquired by image processing techniques. The images are scanned thin section images about the cross-sections of a sample. Marbles and sandstones consist of grains. The provenance can be classified by the distribution of grain sizes and the quality depends on the average number of neighboring grains. Quality in this sense means how resistant is the sandstone against water. This is important to know during restoration works. In order to determine the sizes and the neighbors, the grain boundaries should be identified [1]. To summarize, the whole analysis process consists of the following steps consecutively: (1) identify the grains on the raw thin section images, (2) draw statistics about the grains like size histograms and (3) make inferences from statistics to provenance or quality. Traditionally this pipeline is fulfilled manually which tends to be time consuming due to the fact that right inferences requires representative statistical data. To gain representative statistics a lot of grains are necessary to be present on the image which induces big pictures with high resolution. There are two major challenges. One is on the algorithmic side, how to find the borders despite noise on the picture. The other is that processing large images can be computationally demanding. The project GrainAutLine addresses these challenges. The main goal is to make easier the analysis by automating the steps and providing helpful tools.

In this paper the project GrainAutLine is introduced from the architectural perspective and from the available user interface tools’ point of view. Two segmentation related tools are covered deeper in the paper as well. The first is a Monte Carlo based segmenting technique while the second is a reinforcement learning based smart drawing tool. The last one showcases the flexibility of the framework because it can incorporate a structurally very different learning method than supervised methods which are more common to use for image processing tasks. The development of these tools is still in-progress.

DOI: 10.26649/musci.2017.058
PROJECT GrainAutLine

GrainAutLine is software which aims for supporting experts with tools for marble and sandstone thin section image processing. From a higher vantage point this project is a subproject of a larger project called cv4sensorhub. The cv4sensorhub manages projects which deal with image processing on special fields. Currently two subprojects have been developed: tracking white blood cells on microscopy images and segmenting marble and sandstone thin section images. The framework has an abstract model which possesses the common requirements of the different projects. As a result the architecture provides a general environment for different image processing tasks. In this section the architecture is proposed and exemplified with GrainAutLine as an instance of the architecture.

Project GrainAutLine1

![A screenshot on the old version of GrainAutLine, called GrainAutLine1](image)

GrainAutLine1 does not use the cv4sensorhub architecture. It is an application written in C++ and uses the Cuda technology to enhance the performance of the image processing. Because of that users without NVidia GPU can suffer high performance loss. The UI was developed with qml. The low level approach provides good performance optimizations. The need for integrating the common image processing parts of different projects, the need for state-of-the-art UIs and the available
technologies offered by .Net made unnecessary the too low level technologies in this context. The project GrainAutLine2 uses state-of-the-art .Net technologies with better architecture by taking into account the well-tried solutions and incorporating the developed algorithms from GrainAutLine1.

The most important requirement is to separate the data model (image, result of the processing) and the algorithm (which processes the image). This is important because of two things. First, the framework provides general environment for image processing therefore it is vital to reuse the algorithms between applications. Second, the view on the UI depends on the data therefore the development of the UI should be independent from the development of the algorithms. As a consequence it is not required to implement data structures for each algorithm just uses the already developed data model and implement algorithms independently of the UI (view) part of the system.

Due to the separation of data model and algorithms (hereafter operations), each image processing tool is implemented as an operation. The provided operations are:

1. Operations for segmentation: adaptive threshold segmenting, Canny edge detector, Watershed, RJMCMC (see later) etc.
2. Operations for manual interventions: drawing closed shapes to draw around grains, merging, slicing etc.
3. Operations for statistics: querying special predefined properties from grains, show the neighborhood graph of the grains, smart drawing tool (see later) etc.

GrainAutLine currently provides more than 20 operations. Some of these operations are shared with the other subproject (the white blood cell tracker one).

*Project GrainAutLine2*

![Figure 2](image)

A screenshot on the new version of GrainAutLine, called GrainAutLine2
The project GrainAutLine2 is a domain specific instance in cv4sensorhub and was developed in C#.

![Domain Specific Application Diagram]

The abstract architecture of the cv4sensorhub framework

The Fig. 3 shows the abstract architecture of the framework. At the bottom level the framework uses .NET technology to implement the functionalities. The UI is based on WPF because of its advanced features. WPF provides advantages. WPF is able to transform the objects on the canvas by built-in functions, and delegating the task to GPU when necessary. This has huge impact on the performance of the UI which enhances the user experience during work. Developing complex UIs in WPF is easier and faster for an average programmer than in qml.

At the next level the common data model is defined which encompasses the most typical elements of the processed image. These are the followings:

1. The original image as the target of the processing.
2. Polygons which represents concrete formations. For instance in case of GrainAutLine these are the grains.
3. A special one, the so called seedpoint, which represents a single point. This is typically good for user interventions which useful for manually marking locations to guide the semi-automatic techniques.
4. Polylines to represent curves.

These are referred as entities too. Entities have tags to connect important values to them. Every tag has the form {name, value}. Each entity has a default tag “id”, to distinguish them. To speed up the searches by tags, a special indexer is defined for each tag. This is based on key-value pairs in a sorted dictionary. The key is the value of the tag; the value is the corresponding entity. The next level consists of four different modules:

1. The UI common gathers typical UI elements like image viewer, operation selector, appearance commander etc.
2. The Derived Data module collects objects which can calculate and store frequently used data from the entities, for instance the distances between the entities.

3. Operations contains all the algorithms which can manipulate on the data model. The input is an instance of the data model; the output is also an instance of the data model.

4. The project cv4sensorhub has the ability to access data collected by IoT systems and stored in databases. The persistence and communication module manages these functionalities.

The **domain specific application** is a concrete application like GrainAutLine2. Due to the separated data and operation approach the framework is highly flexible for incorporating new algorithms. Due to the MVVM architecture the UI can be developed independently as well.

**SEGMENTATION WITH MONTE CARLO**

![Figure 4](image)

**Figure 4**

Left: the segmented thin section image. This is a so called over-segmentation. The RJMCMC algorithm gets over-segmented pictures as input.

Right: the original input showing twin crystals. Twin crystals are formed under special physical conditions, when some smaller crystals melted into a bigger one.

In this section a special segmenting algorithm is proposed which is also part of GrainAutLine. The algorithm focuses on images overwhelmed by twin crystals. Twin crystals are formed under special temperature and pressure conditions as smaller crystals melt into a bigger one. In the thin section image a twin crystal has a lot of almost parallel lines inside its border (see Fig. 4). These lines and curves inside the grains make the identification of the real borders difficult. The lines are not perfect lines in the mathematical sense. The lines sometimes break before it reaches the border. Traditional image segmenting algorithms based on the gray scale thresholds
A SMART TOOL: LIVE-POLYLINE

After the automatic segmentation some errors usually remain. The faults should be repaired manually. This means that the wrong curves are deleted then the correct one is drawn. Furthermore, there are some images which are really challenging typically posed by sandstones and twin crystals.

The basic idea of the algorithm is that it optimizes an energy function [2] which describes how close the current segmentation is to the correct one. Then a monte carlo process is applied to change the segmentation by regrouping the segments (created during over-segmentation), this results a new configuration of the segments. The series of the configurations form a markov chain. Any change in the configuration is reversible because it can be redone by applying suitable regroupings. Each configuration has its energy value. The configuration with the lowest energy is the configuration where each group matches with the real grains.

The regroupings are incremental processes. After every regrouping the energy is expected to be lower.
To overcome this problem a special tool is under development. A live-polyline behaves in the following way. The user clicks on a point (start) somewhere on the borderline then moves the mouse to a next point (target) on the borderline. The algorithm recommends a curve on-the-fly to connect the start point with the target point. This can ease the manual segmentation because the recommended curve fits exactly the border and only clicks are required from the user to draw the curves. Actually two approaches are investigated to develop this algorithm.

**Naïve approach**

The outline of the algorithm:

1. Generate a weight map for the image. The weight map contains a weight value for each pixel in the picture. The weights are derived from the original pixel values, e.g.: luminance. Every weight has to be a positive value.
2. Find the path between the start and the target with minimum total weight. This can be done with e.g. A-star, BFS etc.
3. If the user clicks again at the target then finalize the recommended curve.

To express this in a picturesque way, a good weight map can be imagined as a landscape with valleys and hills. The path in the valley corresponds to the borderline. Two major problems arise in practice. First, to find a right weight map is difficult. Second, the search tends to be computationally demanding and it can cause some delay in the response time.
Reinforcement Learning-based approach

To enhance the performance of the algorithm behind live-polyline, the methods of reinforcement learning were considered.[4, 5] The motivation was the flexible and general model of reinforcement learning. It is based on agents which interact with the environment. The environment is described by its state. The agent can intervene in the environment by actions. Then the environment gives a feedback for the agent by sending a reward. The goal of the agent is to find a behavior (policy) to gather the most rewards in the long run.

Two different approaches are planned to create learning algorithms with this model. The first one learns an approximating function which can calculate the weights. If the function is good then the weight map would be able to reflect the borders. This technique can not eliminate the search but provides better weights. The second one learns to build incrementally, step-by-step, the path between the start and target. This approach eliminates the search and the weight map as well.

Reinforcement learning has a lot of successful applications and results nowadays and therefore we expect it to work on this domain as well.

CONCLUSION

The framework of cv4sensorhub provides a flexible environment for image processing related tasks on different application domains. As an example the GrainAutoLine was introduced. Two segmentation related tools were introduced: RJMCMC is an energy based segmentation approach, and live-polyline is a technique for semi-automatic drawing of grain boundaries. The paper demonstrated the current state of our research. The framework for general-purpose image processing was fully developed and applied on concrete domains as it was shown in the two examples above.

REFERENCES