Patrick Schwarz

Microscope Tracking

Master Project Information Systems

Graz, 28.04.2018

Abstract

This thesis will enlight certain aspects on improving camera based measurements. The project started with the idea, to simplify the teaching of pathology students. A pathologist uses a microscope to observe organic slices. The application should measure the movements of the pathologist and record it for the students to learn. In order to measure the movements, two methods where used. One uses a computer mouse laser to measure the movement, while the other tracks a dot via a camera system.

The main topic of this thesis will be the combination of those two methods. The camera input corrects the steady growing mouse input error which grows with each measurement.

Theoretically it should be possible to get measurements with a precision of 12,4 μm. The maximal error, which adds up with each mouse measurement, can be reduced to 3032,9 μm. Even though the project could not be completed, it shows that it is possible to create this method and that the results could be reasonable.

Contents

1. Background

In this section of the thesis we will describe knowledge that is needed to understand the rest of the topic.

1.1. Microscope

This part will describe the microscope its functions and its components.

1.1.1. Microscope components

First of all we look at the external appearance of the microscope in figure 1. It is important to notice that not all microscopes look the same. The model presented in the figure was the one used for this thesis in the end. In the preparation steps we used a different microscope which was smaller and lighter.

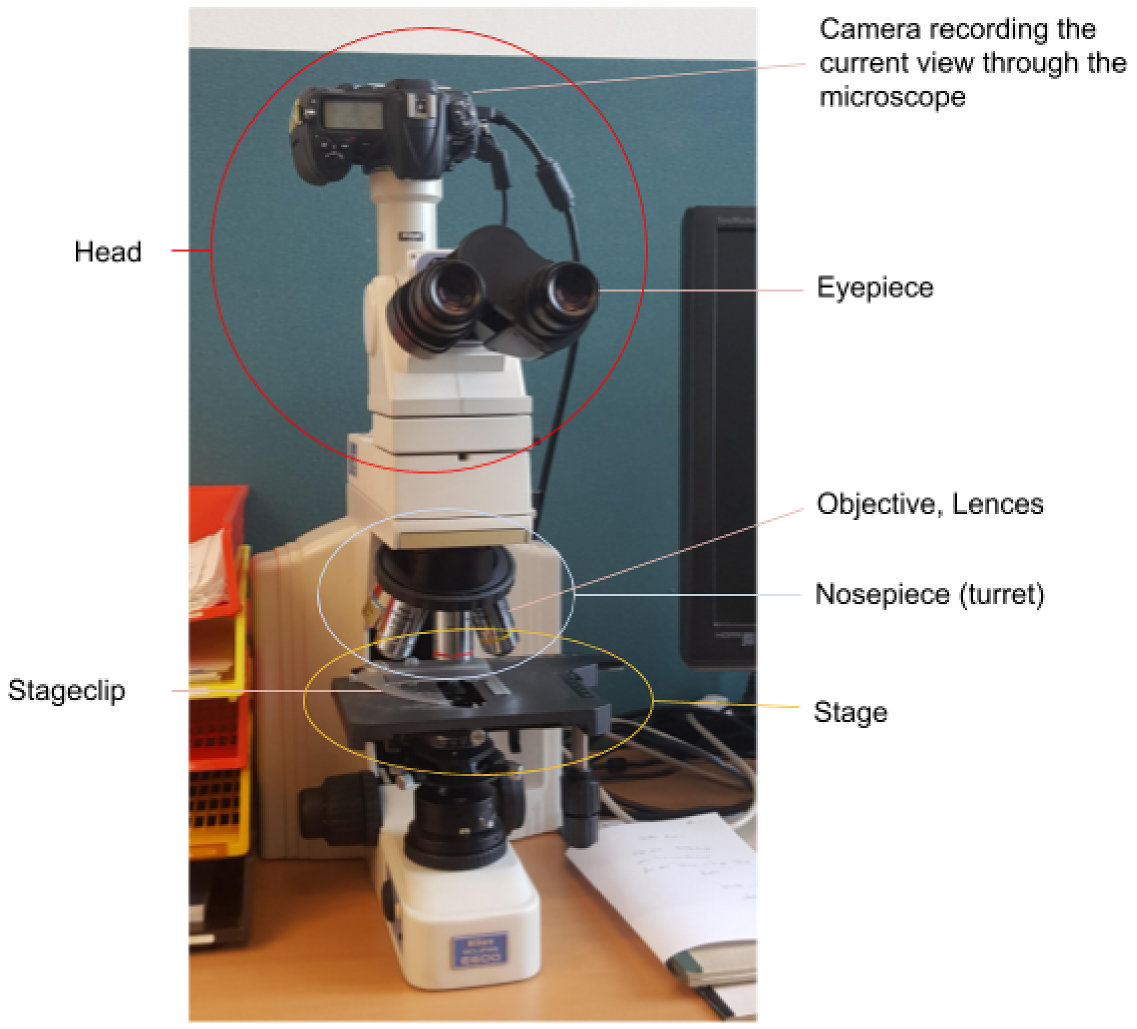


Figure 1: External components of microscope [1]

In order to explain the components of the microscope in a little bit more detail figure 2 shows us the internal parts of the microscope.

In order to make the magnification possible there are two main parts. The eyepiece and the objective have to collaborate. The objective, located close to the object you want to observe, relays the real image onto the eyepiece. The main magnification takes place in the objective. Typically the eyepiece has a magnification as well which lays between 1-30.

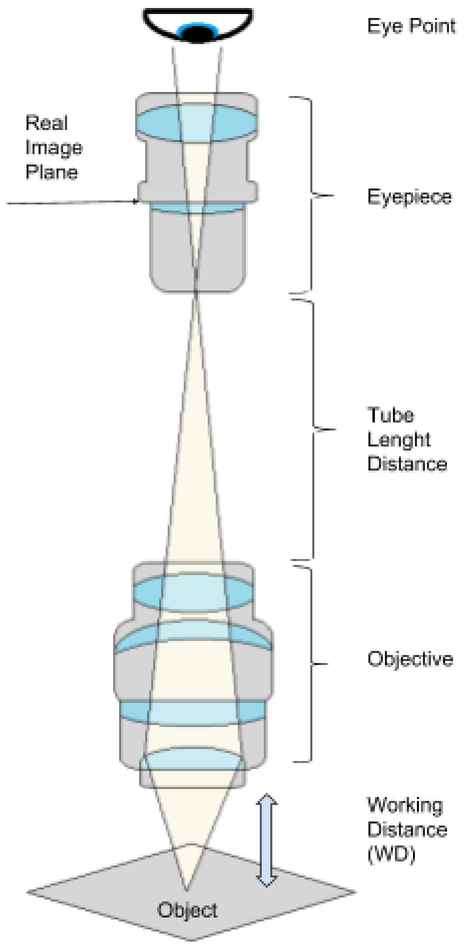


Figure 2: Internal Components of microscope [2]

The magnification of the objective and the eyepiece have to be combined with equation 1 in order to get the total magnification.

$$Magnification_{Total} = Magnification_{Objective} \times Magnification_{Eyepiece}$$

Equation 1: Total Magnification [2]

1.1.2. Objective specifications

Most of the specifications for an objective can be found on its body. There are multiple specification that can be written onto a objective like objective design/standard, numerical aperture, working distance, lens to image distance and many more. In our case we do just need the magnification. The figure 3 shows how to read the specifications of a objective.

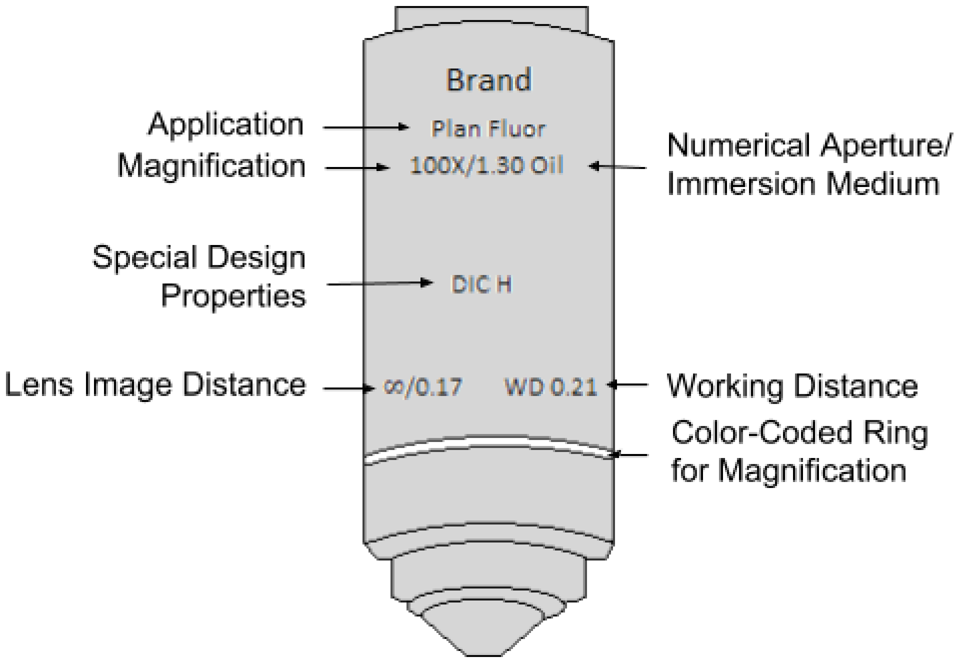


Figure 3: Objective specifications

In order to identify the magnification not only the written form is used but also a color ring. This makes it possible in our project to observer the magnification by color via a camera. Table 1 shows the magnification and its corresponding color.

Magnification	**1X**	**2X**	**3X**	**4X**	**10X**	**20X**	**40X**	**60X**	**100X**
Color Code	Black	Brown	Red	Red	Yellow	Green	Light Blue	Cobalt Blue	White

Table 1: Magnification Color Codes [2]

1.1.3. Field of View

The field of view describes the diameter which is visible by looking through the eyepiece. In most cases the diameter depends on the magnification of the objective. To calculate the field of view we have to determine our parameters. These parameters can be found on the side of the nosepiece and eyepiece. If a camera is used we have to look up the sensor size. In order to calculate the field of view we can use equation 2. The total magnification can be derived by equation 1.[3]

$$Field\ of\ View_{Camera} = \frac{Camera\ Sensor\ Size}{Magnification_{Objective}}$$

$$Field\ of\ View_{Eyepiece} = \frac{Fieldnumber}{Magnification_{Total}}$$

Equation 2: Field of View [4]

1.2. Whole Slide Imaging

Whole slide imaging is a technique born in the pathology. In order to form a whole slide image a scanner is used which scans a slide of biometric material between two glass plates. The scanner generates different magnifications of the object in order to reproduce the same results a microscope would do. [5]
In figure 4 one can see how the scanner constructs a whole slide image in several loops over the image. The scanner divides the whole image in rows and columns.
In order to view such images the software displaying the image has to adapt the zoom to the image with the right magnification. In this thesis the whole slide images do not get used but they are a prerequisite to produce the end result of all combined projects.

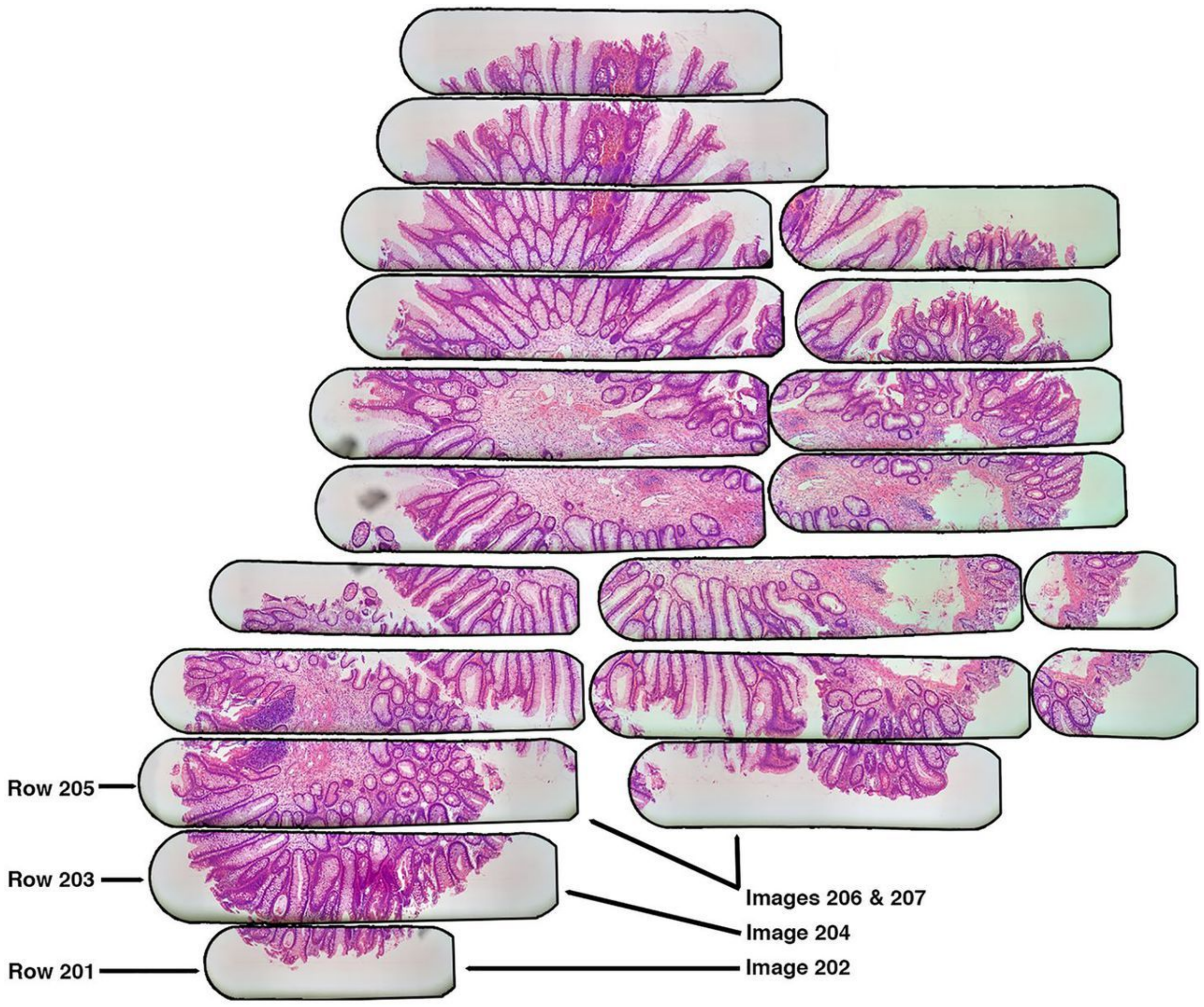


Figure 4: Whole slide imaging [6]

2. Measuring

This part of the thesis will explain the different methods we used measure the input. This is the main part of the thesis and should enlight certain advantages and disadvantages of each measuring method.

2.1. X-Y Axis

The first part of the measurement is about the measurement of the two axis. Therefore we need to determine in which direction the stage of the microscope has moved.

2.1.1. Precision

First we have to determine the precision needed for the recordings. The highest possible resolution for the scanning of a whole slide image are 0.121267361111111 micrometer per Pixel. In figure 5 we can see that the tolerance for a whole slide image on a monitor with the highest resolution would be 145 µm.
If we want to show the student the same picture as the pathologist we have to be smaller than this maximal error. The center point on the pathologists screen would then be at least on the screen of the student.

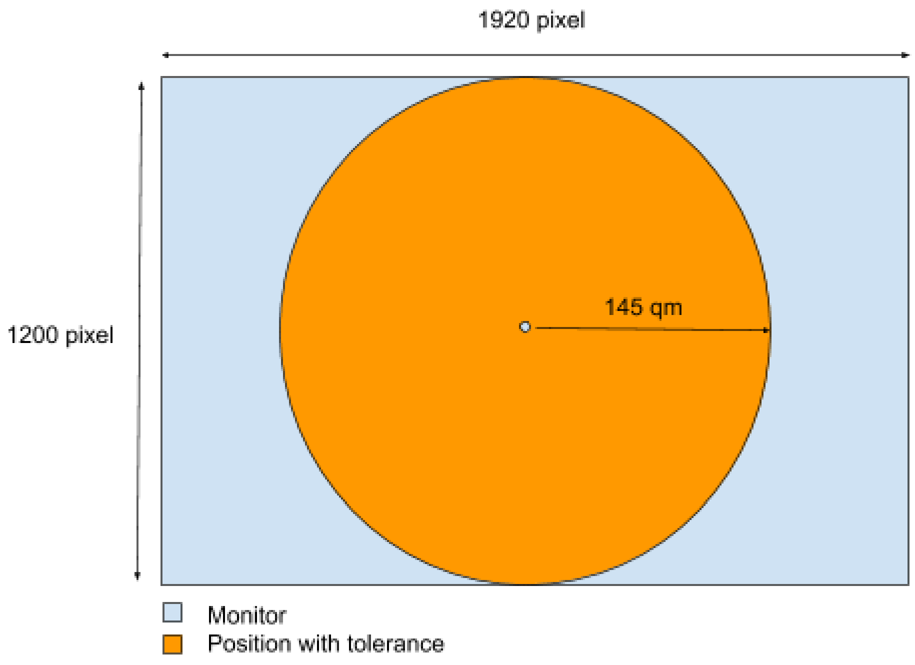


Figure 5: Monitor comparison

2.1.2. Microscope Movement

Figure 6 and figure 7 show how the different parts of the microscope move in the X and Y Axis. It is important to mention that it is different for each microscope and we had to construct our solution in a way each microscope type can use it.

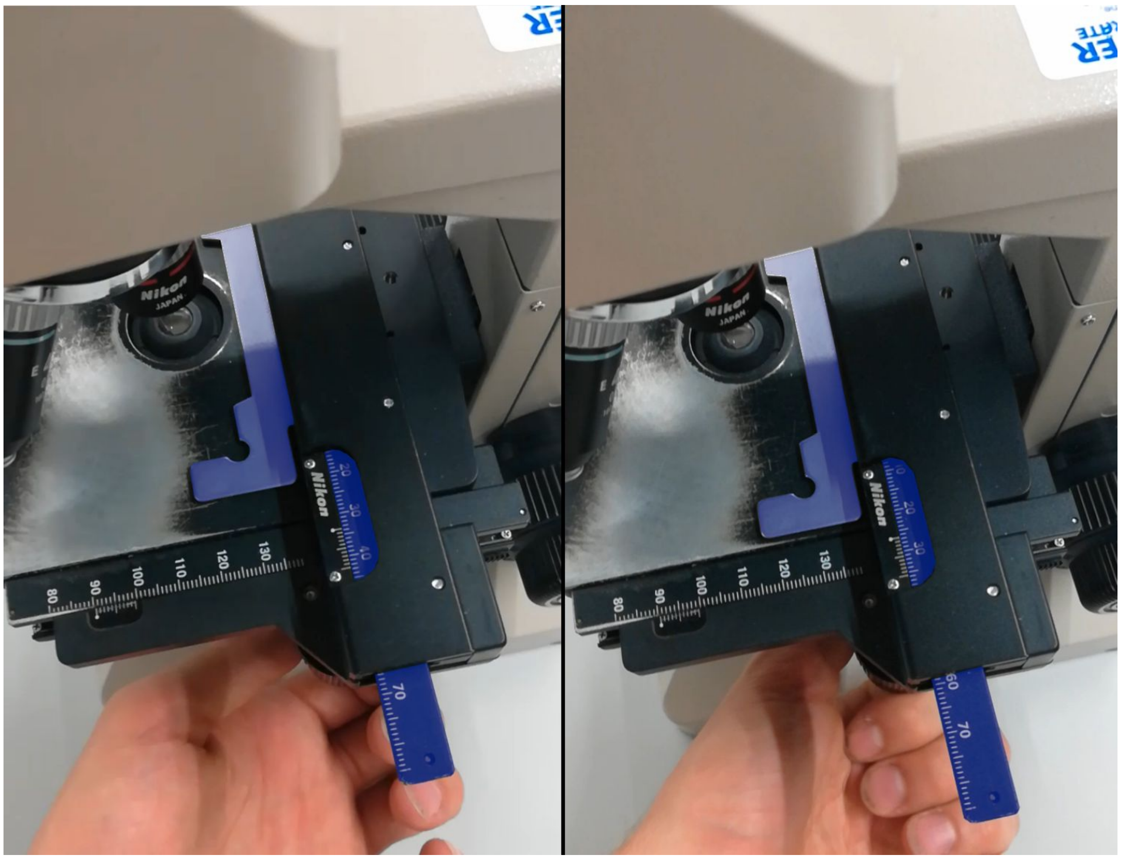


Figure 6: Moving parts in X Axis

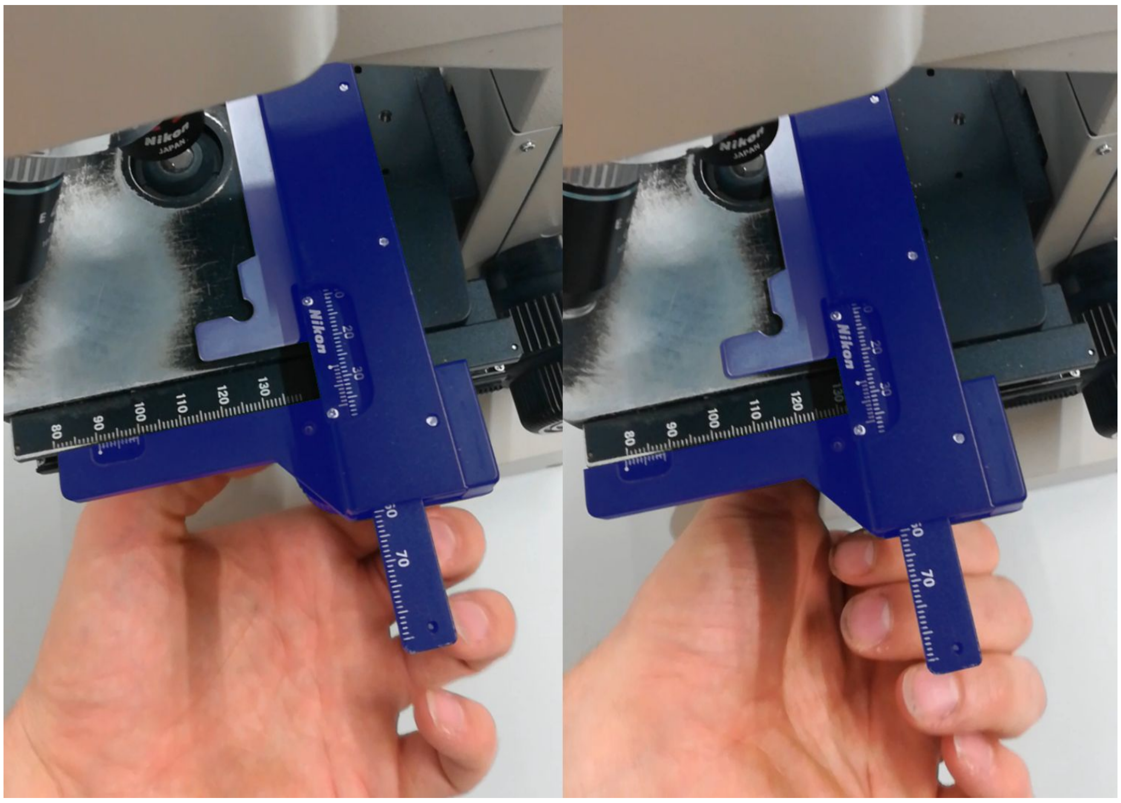


Figure 7: Moving parts in Y Axis

All blue colored parts will move in the direction stated in the description. It is the movement of these parts that we want to observe. Only a view of these parts are moving in both directions. These parts we could use for our observation.

2.1.3. Camera-based measuring

Our goal is to use the OpenCV Python library, a microscope and some colored stickers to capture the image feed from our camera, identify which magnification is currently active and find the position of the slide relative to the microscope.
Here are the steps to get to this goal.

Checkerboard calibration
For the very first time using the program we must calibrate with our checkerboard. We can save this calibration to a file to avoid having to do it another time (unless the camera moves). We can use the use_file option in the config file to enable or disable that.
The config file also lets us specify the information about this calibration such as the width and height of the checkerboard and the size of a tile.
The checkerboard has to be on the same plane as our slide, so typically we will place it on the table where the microscope will be.
To actually do the calibration we use the findChessboardCorners function from OpenCV from which we get the corners (intersections of tiles) pixel position.
We associate to the corner pixel points a 3d point in space, which we say are simply on the Z = 0 plane, and the X and Y are integers representing the position of the corner on

the checkerboard. This way we have reference axes that is entirely based on the checkerboard.
We calculate the reprojection error and only confirm the calibration if it is below 1 pixel.

Fish eye calibration
If it is necessary, we can calibrate the camera to compensate for any fish-eye effect.
If we wish do do this, we must turn on the correct_fisheye option in the config file.
This will reproject the image points according to the camera matrix and distortion coefficients found from the checkerboard calibration.

Magnifications calibration
The next step is to tell our program to which magnification corresponds which marker.
To do this we just click on the picture where the marker is and then type in the terminal the value of the active magnification.
In the background, the mouse pixel position is found, and the pixel color is calculated by averaging the color at this pixel over a few frames. This makes detection more solid in case the lighting changes very slightly every other frames. The number of frame to average over can be changed in the config file.
Once the color is calculated it is associated to the magnification value and stored for later.
The position of the click (and thus the marker) is also stored, so that when we look for the marker of that color, we only look in a small radius around that position (the magnification marker will always be in that position). Since the markers for each magnification are not placed exactly in the same position, every time we calibrate a magnification, we recalculate the marker position to average it between every click positions.

Slide calibration
Next, we click on the marker attached to the mobile part of the microscope (the part that will move with the slide).
Using the same technique as for the magnifications calibration, we store the color of the marker. We however don’t store the position, as the marker will be moving so we need to look for it in the entire picture.
Then we click on the first fixed marker (doesn’t move with the slide) and store the color and the position of the marker (to limit our search around that position), and then we do the same for the second fixed marker.
The distance between our 2 fixed markers is calculated in pixels and is compared to the fixed_distance option in the config file, which is equal to the real distance in millimeters between the 2 fixed markers. This allows us to have a correspondence pixels to millimeters in the plane of the slide.

Axis calibration
Now we need to know what the axes of the slide are, as they may differ from the ones from the checkerboard.
To know this, we move the mobile marker on a single axis, which will correspond to our X axis after calibration.

Once done, we click anywhere on the image and the program will now look for the new position of the marker.
If found, we calculate the angle between our absolute X axis and the slide's X axis created by the 2 positions of the mobile marker.
We store this angle as our reference for finding the position of the slide in the axis of the slide.

Location detection
Now we have all the tools to express the position of the slide on the microscope in its own reference frame.
First we detect the mobile marker in the image. We then transpose the point in the right axes by moving it and rotating it, and finally we scale it using the reference distance to get the position in millimeters.

Theoretical Precision

Camera resolution in pixel	Field of view in cm	Precision in µm
3840x2160	10x10	46,296
3840x2160	5x5	23,148
3840x2160	3x3	13,888

Practical Precision

Camera resolution in pixel	Field of view in cm	Average error after 50 samples in µm
3840x2160	10x10	5513,21
3840x2160	5x5	3032,9
3840x2160	3x3	2876,45

Table 2: Theoretical versus practical results camera-based method

The results from table 2 show a standard deviation of 0,5 mm. Increasing the field of view does not improve the results. This effect is due to problems with the focus, if the camera is placed closer to the observed object.
Even though the camera is calibrated the edges do contain much worse results than the center of the image.

2.1.4. Laser-based measuring

In this section we will explain a laser based method for measuring the x and y axis of the microscope. The idea is to use a computer mouse laser in order to recognize movements in the two axises. In order to make measurements possible the first problem is the mounting of the laser.

We have to ensure the laser sits on a position where it can measure each direction. First we tried to use the same approach for camera-based and laser-based measuring but the gooseneck mounting device did not work out. The problem was that it wiggles while measuring which influences the results. The second approach was much more convenient. In figure 8 you can see that we fixed the laser to the stage clip described in figure 1. This allows the mouse to move the same way the slide does.
The second problem we encountered were transmission problems. At first we used a wireless mouse (Logitech m-rbr125) , because we did not want to have any wires which would influence the movement of the mouse. After a while we discovered that the wireless mouse fails to send all the movements to the computer. The failures are not visible in the cursors movement but they are a problem for the measurement. Another problem with the wireless mouse we used were the difference between fast and slow movements. We could not tell if it is only this wireless mouse behaving wrong or if it is true for all wireless mize.

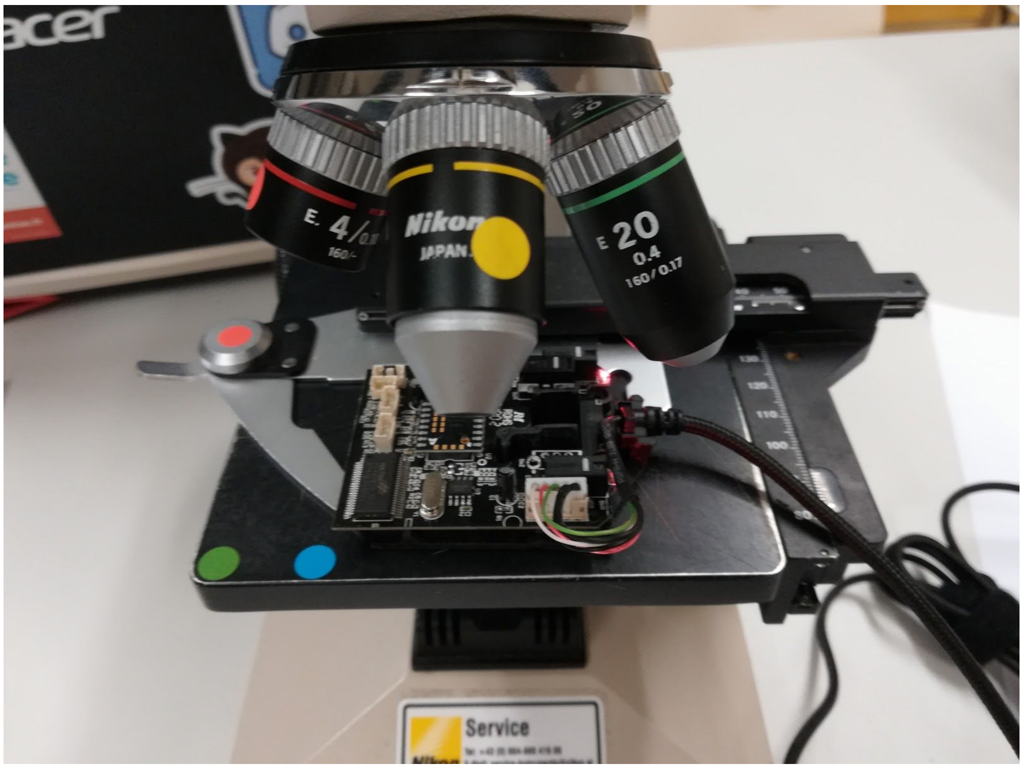


Figure 8: Mounting mouse onto microscope

In order to solve this problem we started to use a wired mouse. The results showed that we have precise measurements and any of the failures from before.
To increase the accuracy we used the Utechsmart UK Venus. The mouse had a resolution of 16400 dpi which gives us a theoretical precision of 1,46 micrometers.

Theoretical Precision

Points per inch	Points per centimeter	Precision in µm
2400	1000	10
10800	4500	2,2222

16400	6833,333333	1,4634

Practical Precision

Points per inch	Average error over 50 samples in µm
2400	145,2
10800	32,1
16400	12,4

Figure 9: Theoretical versus practical results laser-based method

The results in figure 9 show that the mouse has a standard deviation of 12,4 µm which is much more precise than the camera-based measuring but since the error adds up while using the mouse it rises to above 0.5 mm after 400 samples. This means it is impossible to get a good results because of the error summing up. Maybe it would be possible to overcome them by directly speaking to the chip itself and generating a map of the underground itself. Since the mouse consists of a camera which tracks the movement this should be possible.

2.1.5. Combined Measuring

In this section we try to improve the calibration of the coordinate system by combining both methods. We use the input from the mouse in order to calibrate the camera. And we use the camera input to reduce the error of the mouse measurements.

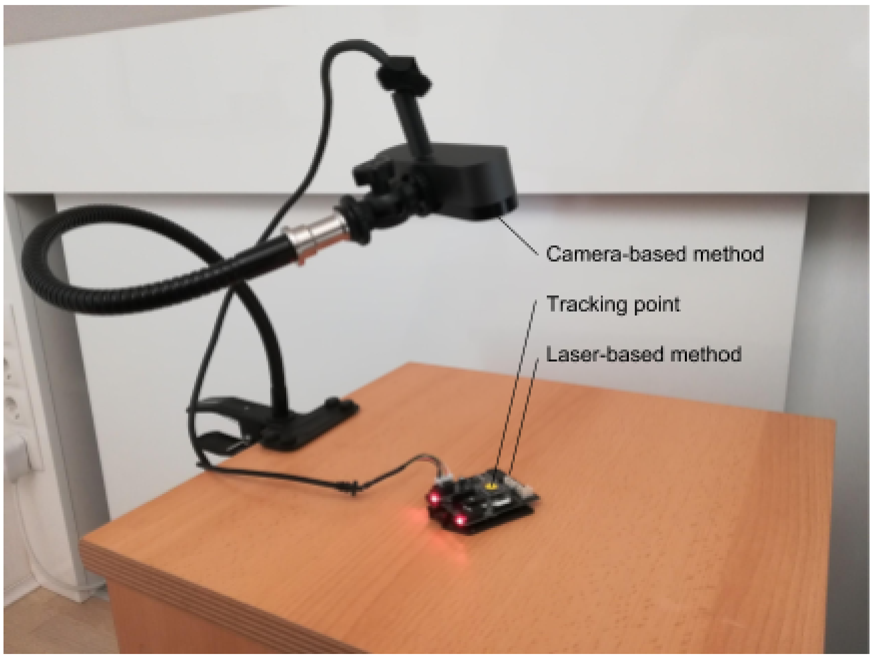


Figure 10: Picture of the combined solution

We can use the absolute measurement of the camera to find errors in the laser-based measurements. Every time the mouse inputs feature a higher error than the cameras standard deviation we are able to correct this error. In order to calculate some theoretical numbers we take the mouse with the highest DPI which are 16400 and a moderate field of view which is 5 by 5 centimeters. The precision of the mouse is 12,4 µm and the standard deviation for 50 samples is 3032,9 µm.

Theoretical Precision [DPI = 16400,FOV 5x5]

Samples	Maximal error in µm
10	124
200	2480
3000	3032,9
10000	3032,9

Figure 11: Theoretical results combined method

The results in figure 11 resembles the assumptions for the combined method. The practical results are not shown because the project itself could not be finished. In the table we can see that the error first depends on the laser-based methods error alone. After 200 samples the camera-based methods error corrects the mouse inputs error. This gives us a maximal error around 0,3 millimeters in Total.

2.2. Magnitude

The magnification will be measured with the camera. We had several ideas. First we had the idea to look with a camera through the microscope and recognize the movement of the objectives in the microscope.The idea can be seen in figure 12.

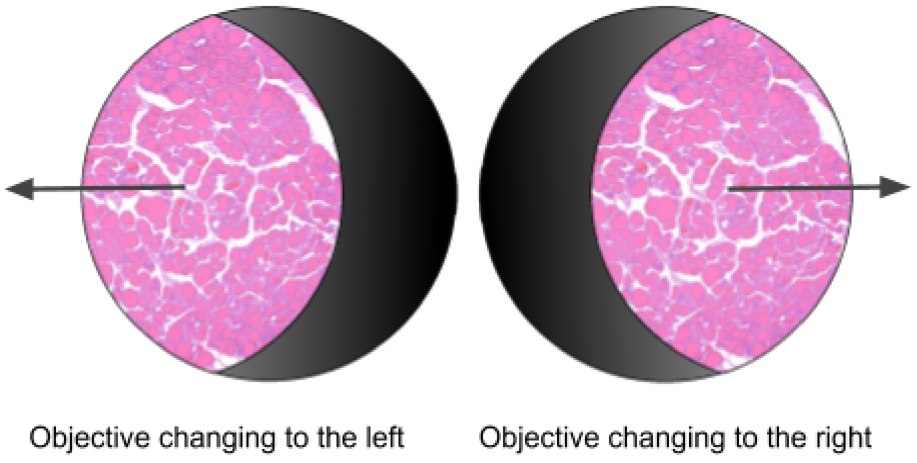

Objective changing to the left
Objective changing to the right

Figure 12: Field of view movement

The next ideas was to stick on little QR-Codes and read out the magnification. We could not find a good enough printer in order to print small QR-Codes readable by a webcam therefore we decided for another solution.

In the last step we only used colors for recognizing the magnification. This seemed to be very promising at first because the microscope objectives have their own color code but as we can see in table 1 there are multiple magnifications for the same color. In the end we decided to use our own color scheme that can be calibrated for each magnification. In the figure 14 you can see our solution on the actual microscope.

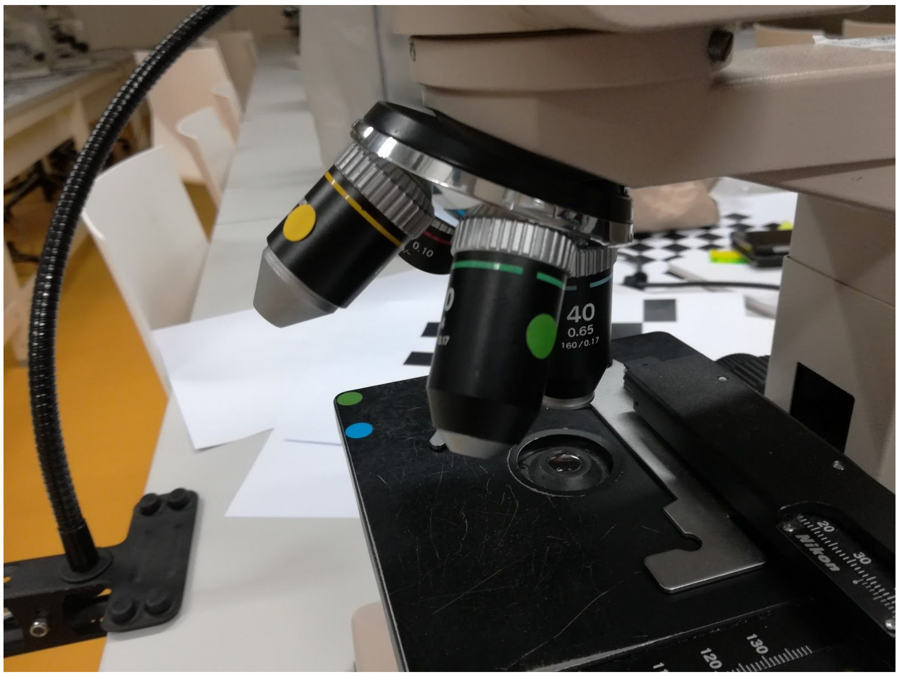


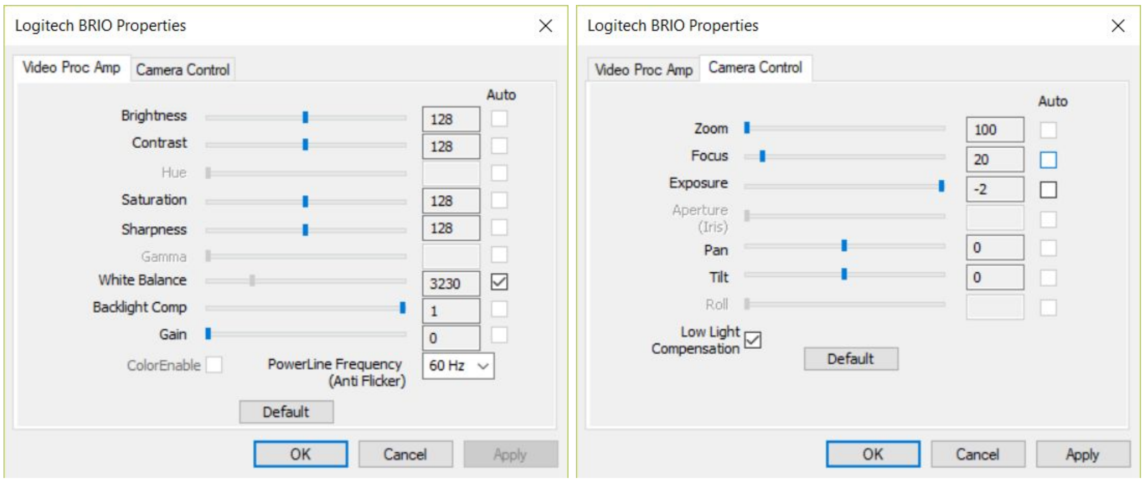
Figure 13: Moving parts in Y Axis

3. Application

In this section we show how the application works and its components. The application itself is a combination of C++ and Python.

Overview

First the the windows in figure 14 emerge. By disabling the Auto-Focus we reduce the interferences with the tracking. If the focus changes while tracking the position changes even though we are not moving. Auto-White-Balance could have the same effect if there is not enough light.



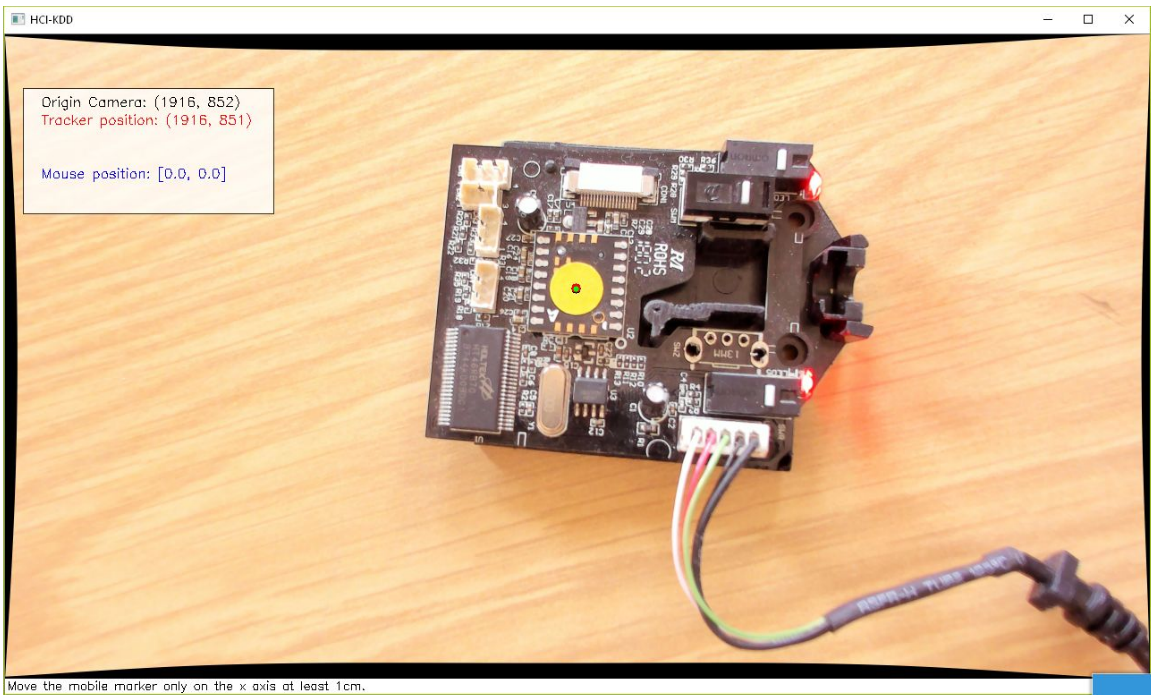
	Value	Auto
Brightness	128	☐
Contrast	128	☐
Hue		☐
Saturation	128	☐
Sharpness	128	☐
Gamma		☐
White Balance	3230	☑
Backlight Comp	1	☐
Gain	0	☐

ColorEnable ☐ PowerLine Frequency (Anti Flicker) 60 Hz

	Value	Auto
Zoom	100	☐
Focus	20	☐
Exposure	-2	☐
Aperture (Iris)		☐
Pan	0	☐
Tilt	0	☐
Roll		☐

Low Light Compensation ☑

Figure 14: Set Focus and disable Auto-Focus



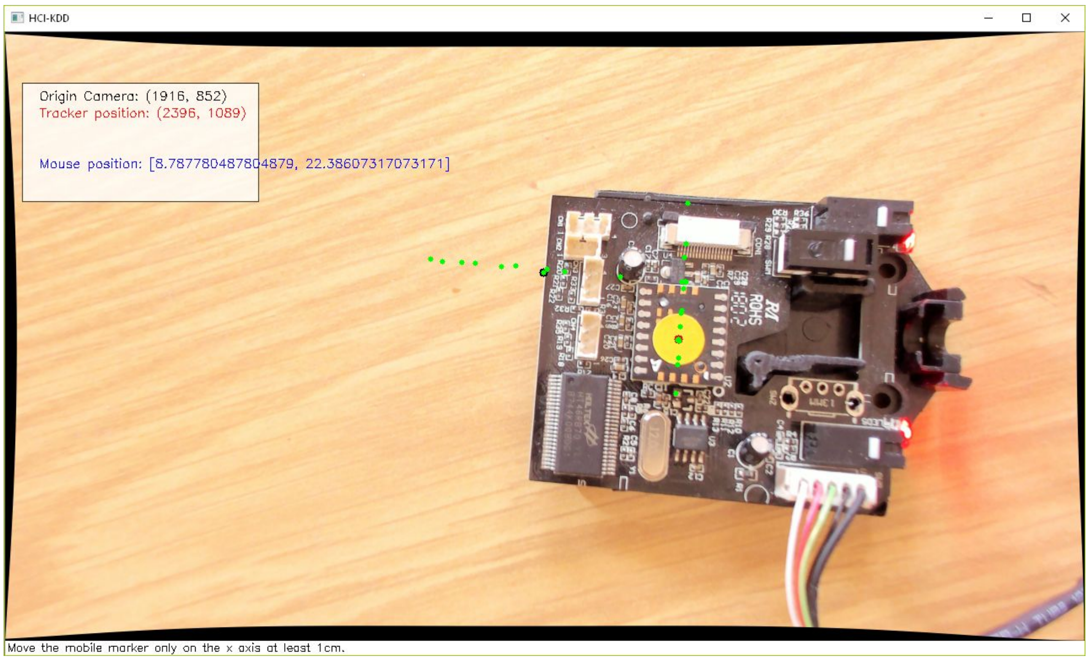
Origin Camera: (1916, 852)
Tracker position: (1916, 851)

Mouse position: [0.0, 0.0]

Move the mobile marker only on the x axis at least 1cm.

Figure 15: Select tracking point

In figure 16 we have to click on the tracking point. The color of the point we clicked will be saved. This color then will be used to track the marker.
The point itself will only be tracked if it does not move more than a certain amount of pixels. This limitation results from improving the interference between our tracking point and other points with the same color. We only search for the tracking point around the last tracked position.
Since Python does not perform that well you have to move very slowly. The frame rate we get from Python OpenCV are five frames per second. Therefore the maximal moving speed is five times the search radius of our tracking function per second.



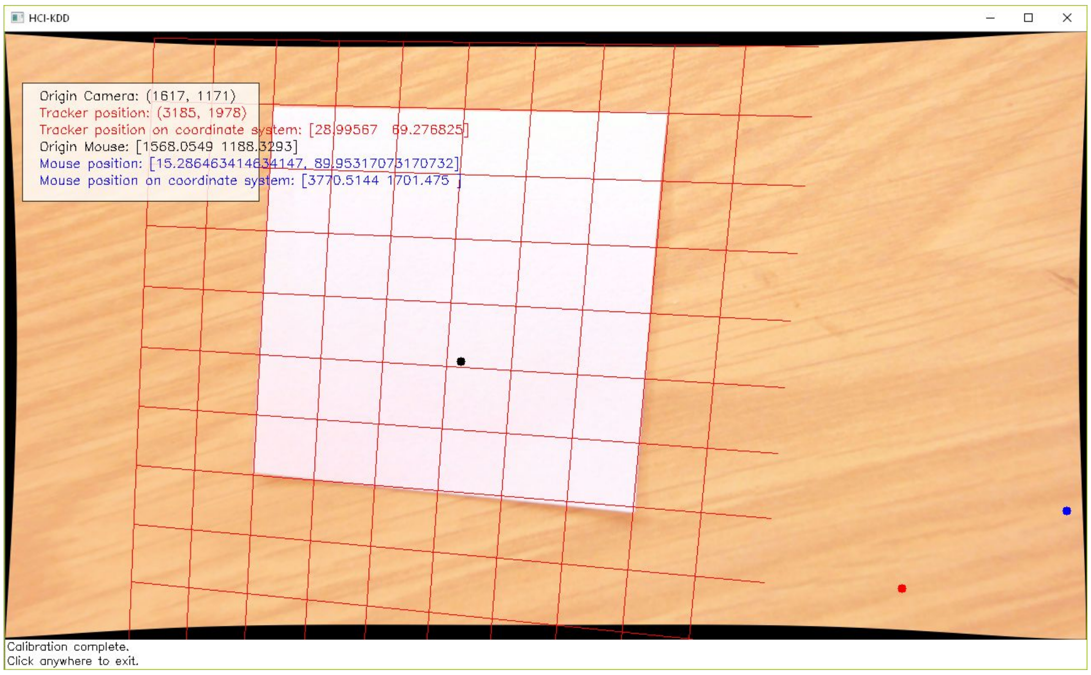
```
Origin Camera: (1916, 852)
Tracker position: (2396, 1089)

Mouse position: [8.787780487804879, 22.38607317073171]
```

```
Move the mobile marker only on the x axis at least 1cm.
```

Figure 16: Recording points for calibration

In figure 16 the green dots mark the recorded points. On each green dot both camera position and mouse position is saved. The recorded points will than be used to find a homography for the both coordinate systems.
In figure 17 we can see how the mouse points are mapped according to the homography. The grid shows the x and y coordinate system of the mouse on the camera picture.



```
Origin Camera: (1617, 1171)
Tracker position: (3185, 1978)
Tracker position on coordinate system: [28.99567  69.276825]
Origin Mouse: [1568.0549 1188.3293]
Mouse position: [15.286463414634147, 89.95317073170732]
Mouse position on coordinate system: [3770.5144 1701.475 ]
```

```
Calibration complete.
Click anywhere to exit.
```

Figure 17: Calibrated grid with 6x6 cm paper on the table

Languages

The application consists of two different languages. The reason for this decision was the limitation of reading raw mouse inputs. Normally the system will take the raw input an translate it to movement on the screen. The maximal precision then is the resolution of the screen and not the Point Per Inch. You have to use native code in order to access the data for the laser-based method directly. Therefore C++ is used to communicate with the system in order to get the raw input.
The pybind11 framework is used to combine our C++ code with the python application. [7] The framework allows us to communicate between our native code. We use the OpenCV library to learn our camera parameters in order to reduce the diffusion each camera. In the next steps the framework evaluates and learns a homography from the recorded points. Here we use a learning algorithm in order find a good solution.

4. References

[1] "AK HCI Selected Topics of HCI." [Online]. Available: https://hci-kdd.org/wordpress/wp-content/uploads/2018/03/AK-HCI-2018.pdf. [Accessed: 28-Apr-2018].

[2] "Understanding Microscopes and Objectives | Edmund Optics." [Online]. Available: https://www.edmundoptics.eu/resources/application-notes/microscopy/understanding-microscopes-and-objectives/. [Accessed: 30-Apr-2018].

[3] "Anatomy of a Microscope - Eyepieces (Oculars)." [Online]. Available: https://www.olympus-lifescience.com/en/microscope-resource/primer/anatomy/oculars/. [Accessed: 30-Apr-2018].

[4] "Field of View," *Nikon's MicroscopyU*. [Online]. Available: https://www.microscopyu.com/microscopy-basics/field-of-view. [Accessed: 30-Apr-2018].

[5] N. Farahani, A. V. Parwani, and L. Pantanowitz, "Whole slide imaging in pathology: advantages, limitations, and emerging perspectives," *Pathol. Lab. Med. Int.*, vol. 7, pp. 23–33, Jun. 2015.

[6] L. Auguste and D. Palsana, "Mobile Whole Slide Imaging (mWSI): a low resource acquisition and transport technique for microscopic pathological specimens," *BMJ Innov*, vol. 1, no. 3, pp. 137–143, Jul. 2015.

[7] "pybind11 — Seamless operability between C++11 and Python — pybind11 2.2.3 documentation." [Online]. Available: https://pybind11.readthedocs.io/en/stable/. [Accessed: 24-Mar-2019].