SRFP 2022 EIGHT-WEEK REPORT

Realization of Functionalities of the CytoCube

Putul Kumari – ENGS3011

Guide: Dr. Sai Siva Gorthi

IISc, Bangalore

INDEX

Acknowledgements

I would like to thank several people for their help and support during this internship.

I am extremely thankful for the cooperation and assistance by my guide Dr.Sai Siva Gorthi, I would like to acknowledge his experience, and support.

I am grateful to all of those with whom I have had the pleasure to work during this internship. Each of the members on this project has provided me extensive personal and professional guidance and taught me a great deal about both scientific research and life in general, My sincere thanks to everyone those who have guided me and helped me in this research internship "Realization of Functionalities of the CytoCube". I would also like to thank them for their efforts and contributions to this work.

I would like to thank my parents, whose love and guidance are with me in whatever I pursue. They are the ultimate role models, I thank my colleagues Prateek, Biplab, Neelam, Aditya, Rajesh, Gayathri, Garvita, Adithya, Santosh, Sujith, Arun, and Anshul for their support. It was partly because of them that there was never a dull moment in the laboratory. A special thanks to the administrative staff, including Akhila for her kind service.

CytoCube

Introduction

Cytocube is a portable automated microscopy (refer Figure 1) whole slide imaging system used for telepathology. It enables digitization of glass slides image capture from blood smears, wet mounts, and microfluidic devices for medical diagnosis with further processing and analysis of glass slides, It's image processing is based on different image techniques especially for overcoming defocus and smear surfaces waves, it uses post-acquisition image enhancement method. This device has a high-resolution imaging capability with precise motorized modules. The global shutter camera helps in continuous motion imaging by which we can get well-focused images at high throughput. The existing system is being controlled using the Arduino Mega2560 Rev3 for uploading codes.

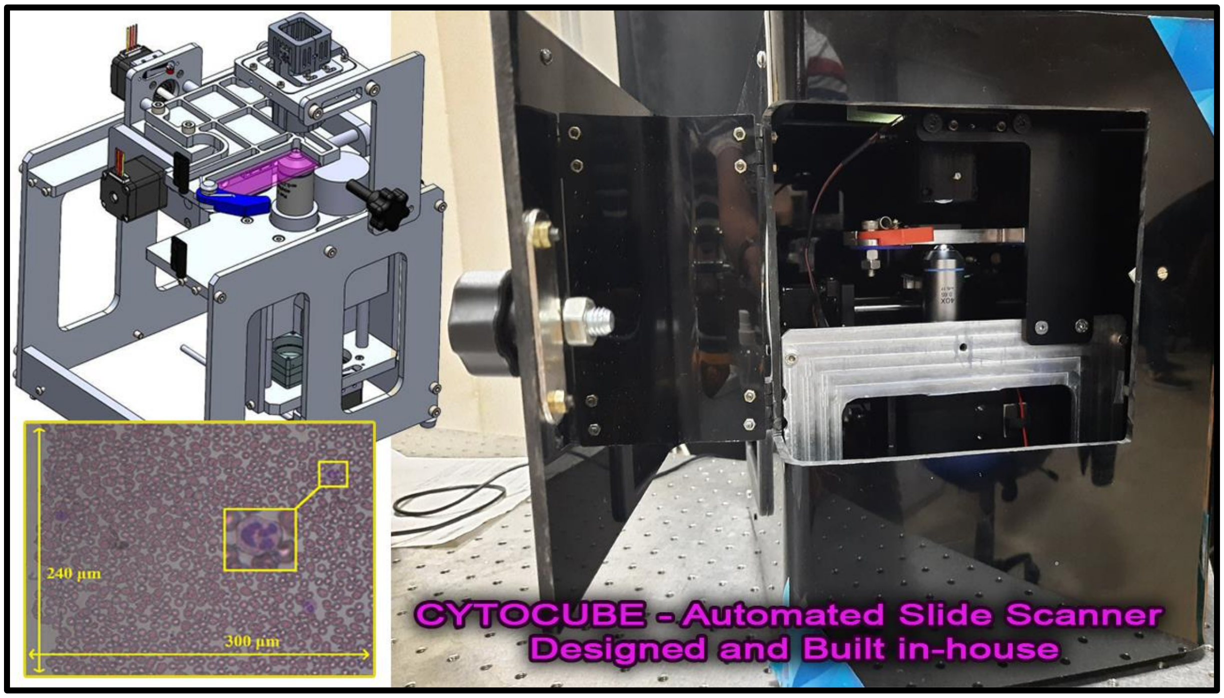

240 µm
300 µm
CYTOCUBE - Automated Slide Scanner
Designed and Built in-house

Figure 1: (a) Physical configuration and structure (b) photograph of the fabricated imaging system with external prototype casing

Specifications and Construction

It is a portable whole slide imaging system. It enables digitizing of glass slides, for display, further processing and analysis. It’s specifications are defined as follow

Whole Slide: 75mm X 25mm

Magnification: Optical 40X Sensor:

CMOS Number of pixels: 1.3 MP

Pixel Dimensions: 5.3 um x 5.3 um pixel

Mechanical Stage and movement:

Multi-axis stage with stepper motors

Speed range: 2-4 mm/s

X-Y movement resolution: 0.2 µm

Z movement resolution: 0.1 µm

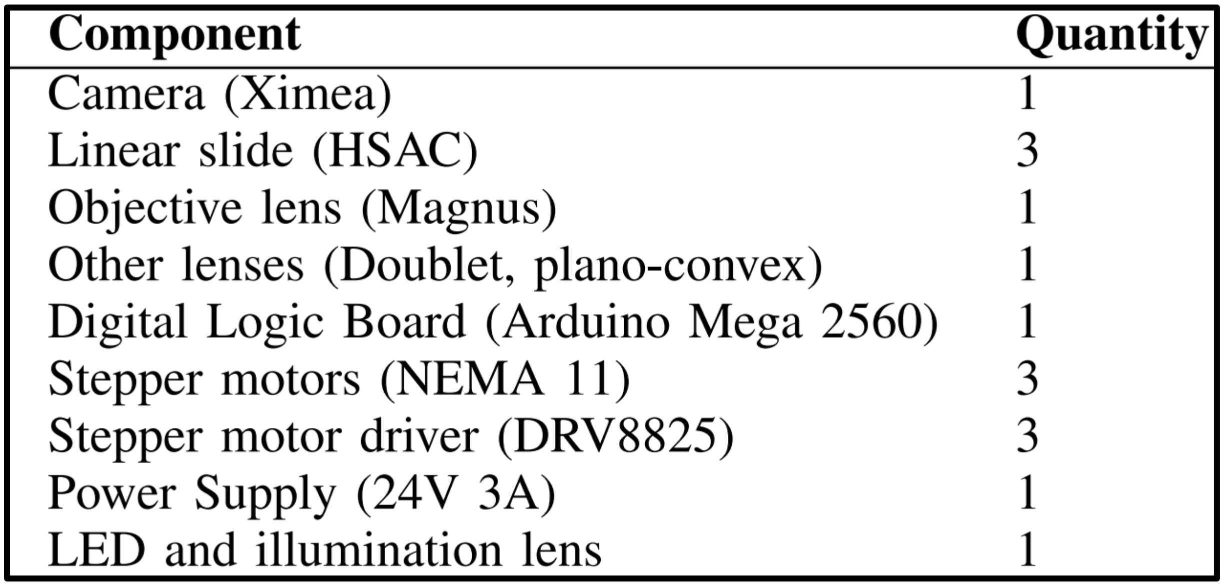
Output Image Formats: JPG

Power Consumption: < 50 Watts

Operating Temp. Range: 5 - 70 ºC

Weight – 10 Kg

Dimensions – 460L*300W*430H



Component	Quantity
Camera (Ximea)	1
Linear slide (HSAC)	3
Objective lens (Magnus)	1
Other lenses (Doublet, plano-convex)	1
Digital Logic Board (Arduino Mega 2560)	1
Stepper motors (NEMA 11)	3
Stepper motor driver (DRV8825)	3
Power Supply (24V 3A)	1
LED and illumination lens	1

Working

a)Global Shutter Camera: Camera used here is in contrast to the commonly used Rolling Shutter cameras, which apply a line-by-line shutter on the image exposure causing blur while relative motion with slide smear. Also, exposure times are usually in the millisecond regime, which means there is a large time window for environmental vibrations to affect the image focus. All these problems can be avoided by using a global shutter camera.

b)Optimized optomechanical design: Works on a set of innovations and some optimizations.

-The innovations consist of a stable mechanical design and spatial focal map-based dynamic focus adjustment capability.

-The optimizations pertain to fast and accurate autofocusing, and automatic smear location via an 8-point scan.

Imaging workflow begins with an 8-point scan to identify the area to be imaged.

i) Stabilized Imaging: decoupling horizontal and vertical axes

The optical (Z) axis containing microscopy objective lens is designed to separate from horizontal (X, Y) translation as it imparts vibrations to the system. The change in momentum during slide translation has only a minimum effect on focus which leads to improved image quality.

ii)Autofocus functionality: three step autofocus

As slide and smears vary in thickness, it's necessary to sweep through large range of focal plane to establish the best focus. Doing this through highest steeping/linear travel resolution throughput the thickness would waste a lot of time.

A three-level autofocus is implemented through the first step being a crude focus step to find an approximate focus plane and then two finer adjustment focus levels help arrive at best focus. The crude focus has a travel size of 10 µm then the finer levels have 5 µm and 0.1 µm with Z range of 300 µm. This can be scanned in about 10 seconds to arrive at best focus.

iii)Focal mapping: predicting best focus on fly

The need for focal mapping arises since the slide and smear are not perfectly planar and there are undulations across length and breadth and the mounting mechanism of lens and slide are not parallelly aligned with each other.

Plane of best focus at one point is usually located at a different height than the one at any other point.

To achieve this, three points of best focus are established first and then the tilt in both x and y axes is calculated.

Thereafter, given the location of any point of the slide, its distance from the origin can be found and multiplied by the tilt amount in both directions to arrive at the final, best focus Z position. This is termed as focal mapping since the process is essentially building a focal map of the entire slide by using best-focus coordinates from three points

iv)Tiled scan architecture:

The accuracy of focal map can be affected by undulation in smear and glass slide. In such case the imaging area is divided into tiles and separate focal map for each tile is built. The system allows to build tiled scan with user-defined tile size.

v)8-point scanning:

As the smear location varies from slide to slide, 8-point scan is implemented where from a set of eight predefined points, three are chosen based on highest value of variance. The eight points are spread to cover maximum area so that even if smear is absent from some portion the system still finds three points where it is present and builds a focal map from them.

Area with dense cell population is ignored algorithmically and only useful cell monolayers are mapped.

c)Post-acquisition image enhancement: After focal mapping and tiled scan if there is any leftover de-focus on the slide it can be corrected by image processing algorithm. We have used local algorithm for faster approach, using guided filter providing functionality such as de-noising, image enhancement, image fusion and it avoids gradient reversal artifacts(which disturbs colour tone). This would also cover positioning errors due to the open-loop operation of the actuators. There exist multiple image processing algorithms for this purpose.

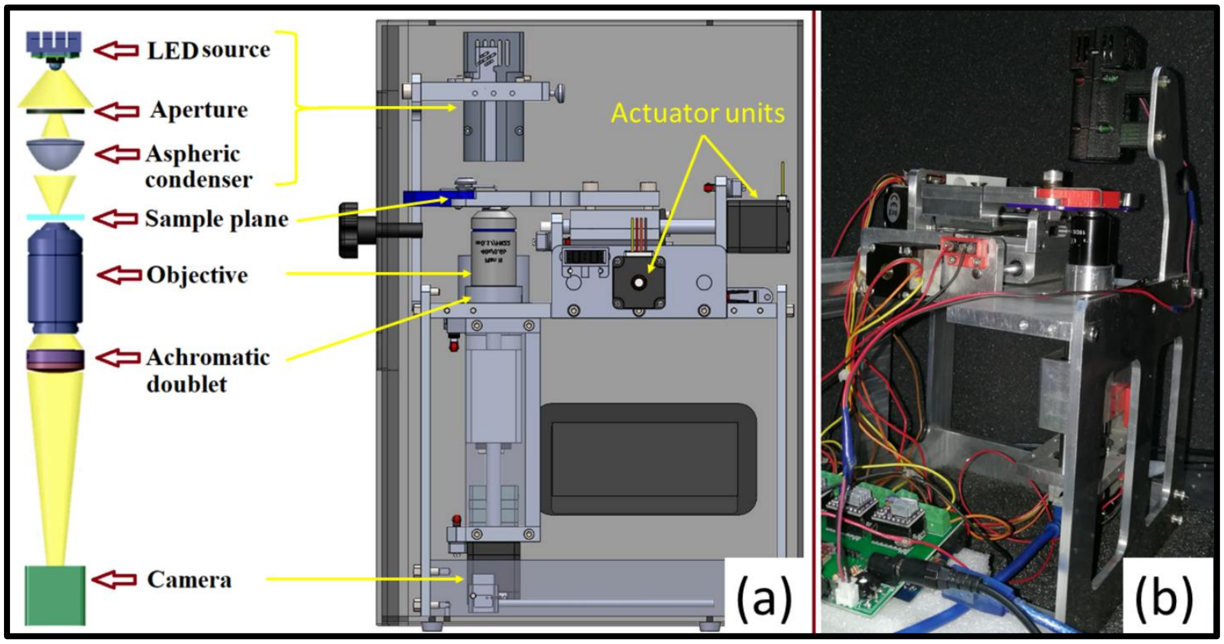


Figure 2: (a) Optical configuration and CAD model, and (b) photograph of the fabricated imaging system including the digital control board with and without the external prototype casing.

Blood Smear Preparation

i) Preparation of glass slides (without staining)

Five glass slides (depicted in Figure 3) were prepared with blood samples having haemoglobin (Hb 15.8 g/dl) without lysis or any staining agent. One specimen was prepared using the concentrated blood sample. Remaining 4 specimens were prepared by mixing blood sample with varying quantities of isotonic solution of 1X PBS (Phosphate-buffered saline). The specimens were in the ratio of 1:1, 1:5, 1:10, and 1:20 Blood to 4 PBS ratio. The prepared specimen was loaded onto the glass slide using the pipette (2-10 µL) and smear was created by placing cover slip at any angle of 45◦ .

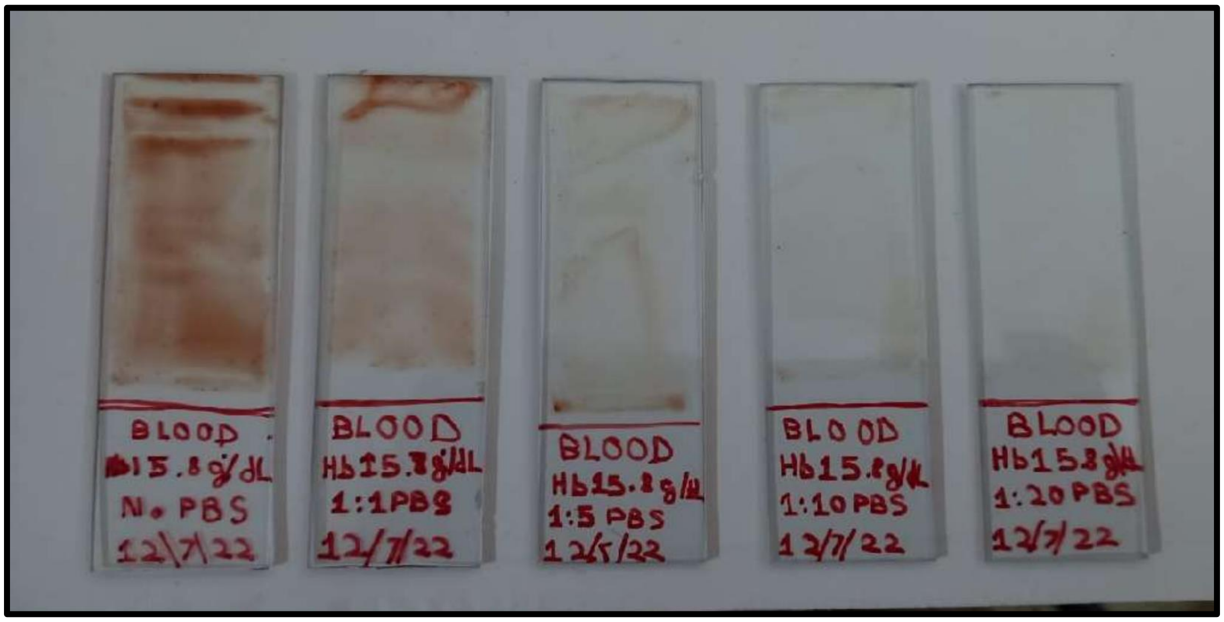


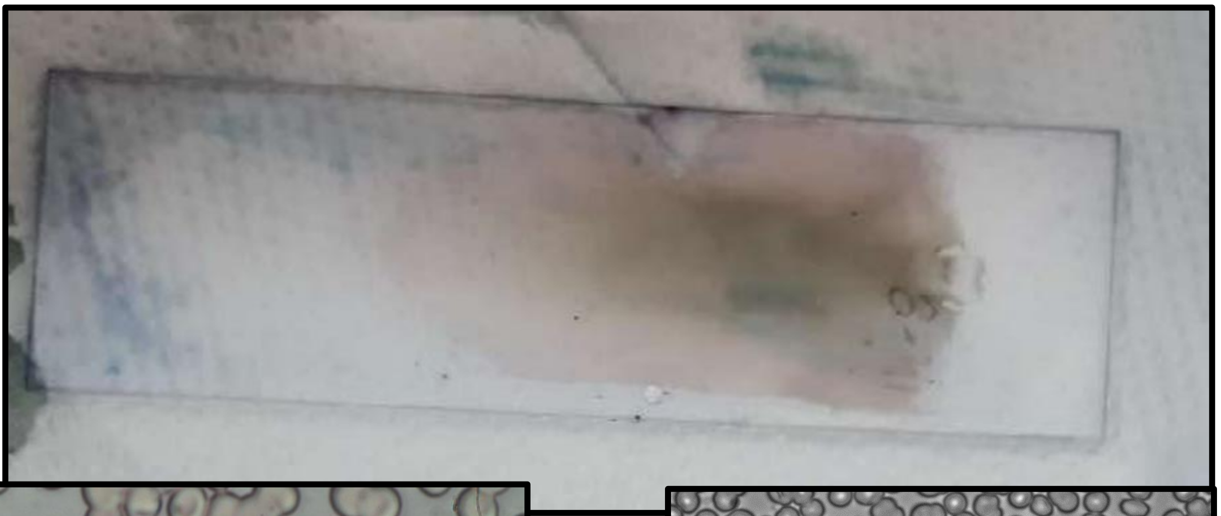
Figure 3: Preparation of 5 Biological Specimens using a combination of blood and PBS

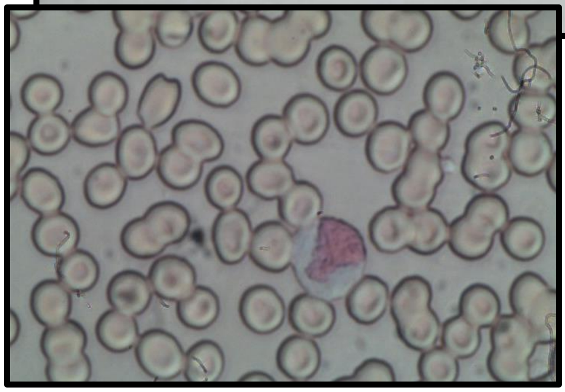
ii) Preparation of glass slides (with staining)

A single glass slides (depicted in Figure 4) was prepared with blood samples having haemoglobin (Hb 14.7 g/dl) and stained with freshy prepared Leishman stain as the staining agent. Leishman Stain (S018S) is a mixture of Methylene Blue and Eosin in Alcohol. It is generally used to stain blood slides to identify WBCs and malaria parasites. To prepare the stain 100ml of methanol was added to 5 g of Leishman stain powder in a falcon tube and shaken for a few minutes and kept aside. Next the freshly prepared air-dried smear was placed at a 45◦ angle in a container and few drops of the freshly prepared staining agent was added to the smear. This was allowed to rest for 1 minute. The methanol in the stain causes dehydration of the blood cells hence resulting in fixing of cells to the slide surface. After 1 minute it was washed off with de-ionised water. Next few drops of PBS 6.8 pH was added onto the slide to improve absorption of stain by blood cells upon rehydration. This was rested for 2-3 minutes and then again washed off with de-ionised water, this time at a slower rate to avoid peeling of cells from the slide. The stained slide was then observed under a 20x (depicted in Figure 4) and 40x magnification light microscope.

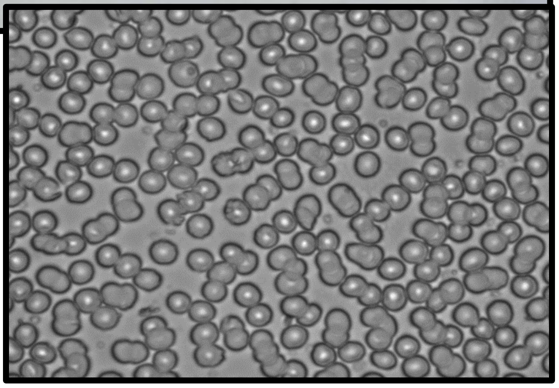
Figure 4: Image depicts blood smear stained with leishman stain

Figure 5: (a) Image depicts red blood cells and a pink coloured nucleus of WBC under 40x magnification





of



microscope. (b) Image depicts RBCs as observed under 20x magnification.

<u>PyQt5</u>

PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language. Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets. Qt classes employ a signal/slot mechanism for communicating between objects that is type safe but loosely coupled making it easy to create re-usable software components.

Qt also includes Qt Designer, a graphical user interface designer. PyQt is able to generate Python code from Qt Designer. It is also possible to add new GUI controls written in Python to Qt Designer. Python is a simple but powerful object-orientated language. Its simplicity makes it easy to learn, but its power means that large and complex applications can be created. Its interpreted nature means that Python programmers are very productive because there is no edit/compile/link/run development cycle.

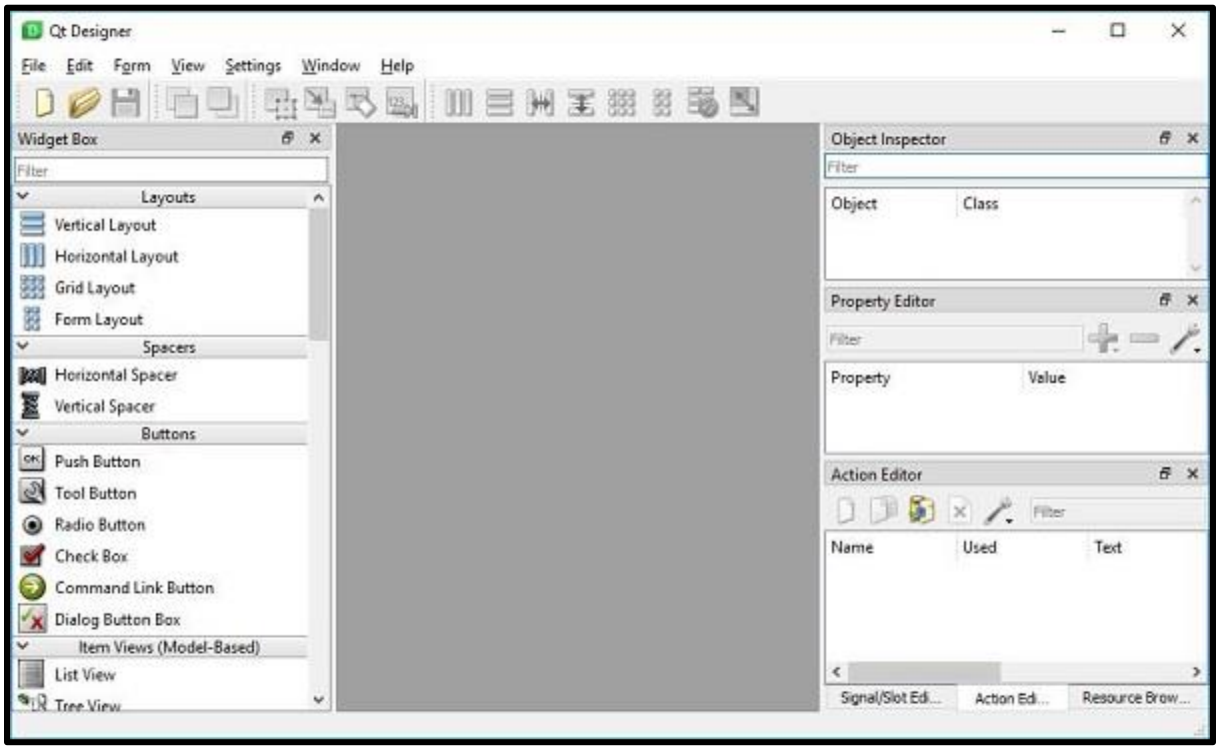


Figure 6: PyQt5 Window

GUI

The previous GUI was developed using MATLAB. MATLAB GUI contains several user interface tools like radio buttons,axes,check box,tables.sliders,list box,pannels..etc. Adding appropriate components we can create a GUI design for any application. GUI interface is an event driven programming. Flow of the application is based on events. Events may be click,double click,key press etc. Callback functions will be executed once an event occurs. We can edit the properties of each callback functions for making suitable response from the GUI as the user ineracts.

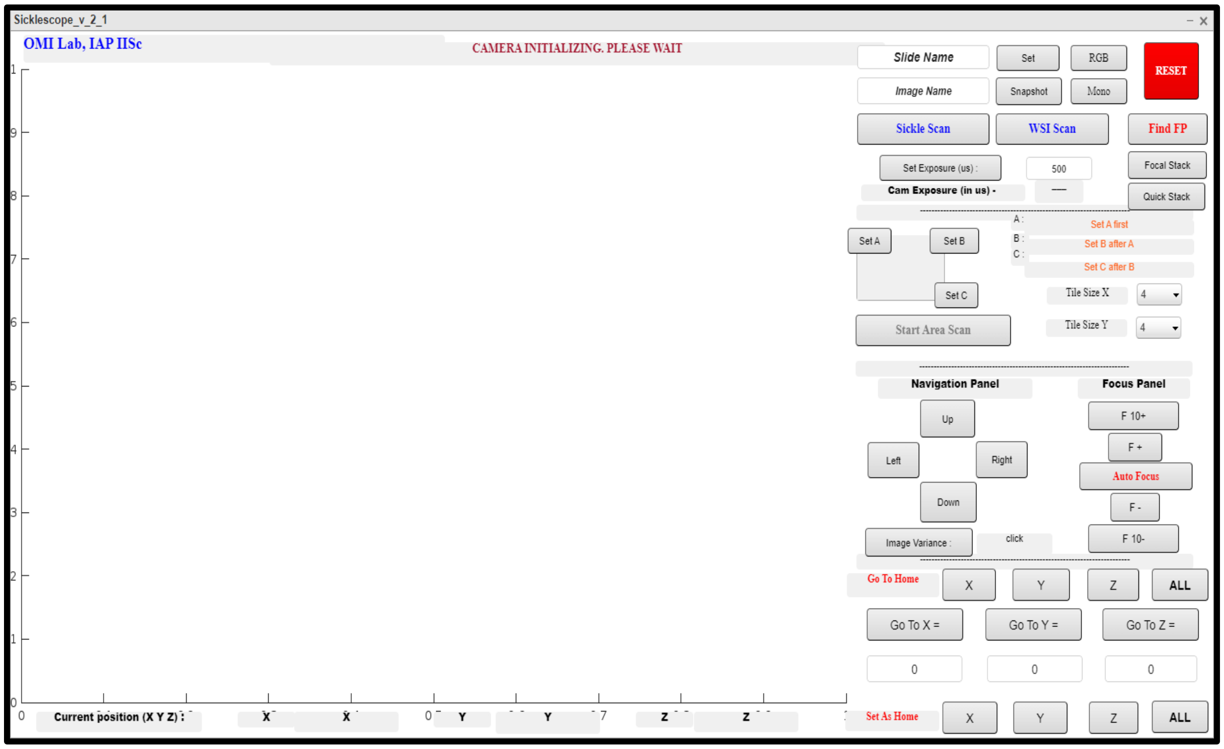
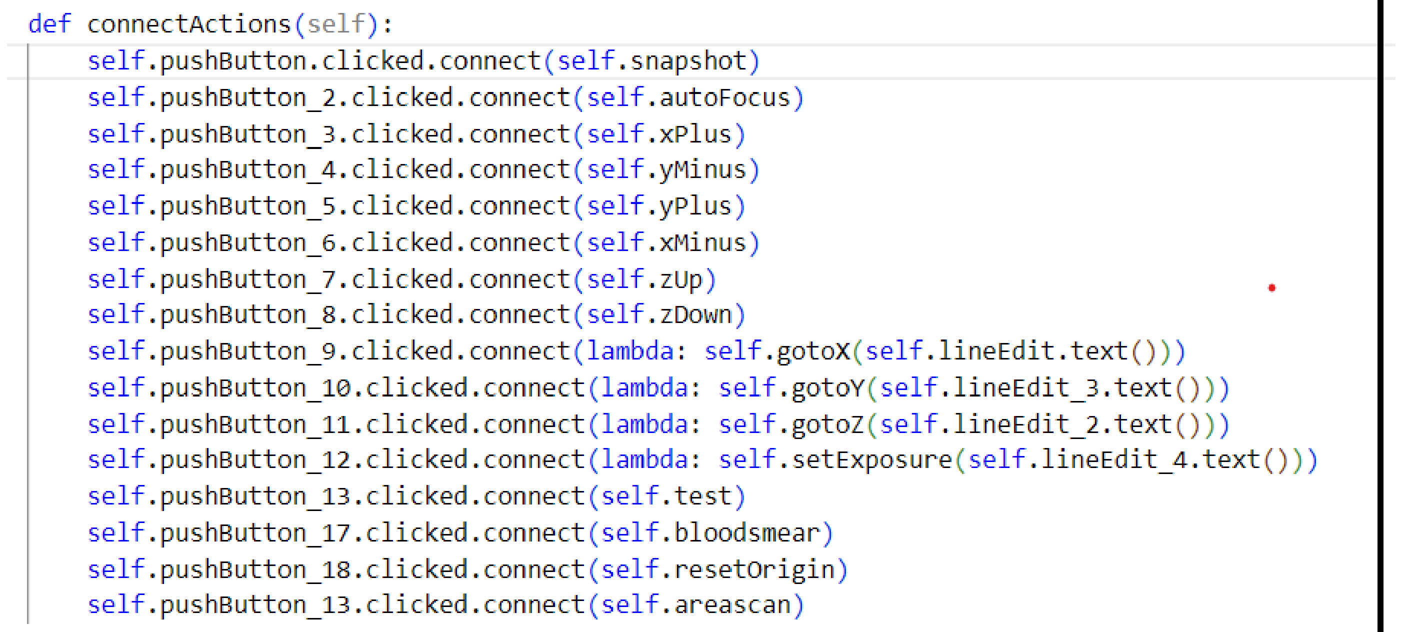


Figure 7: MATLAB GUI

New GUI

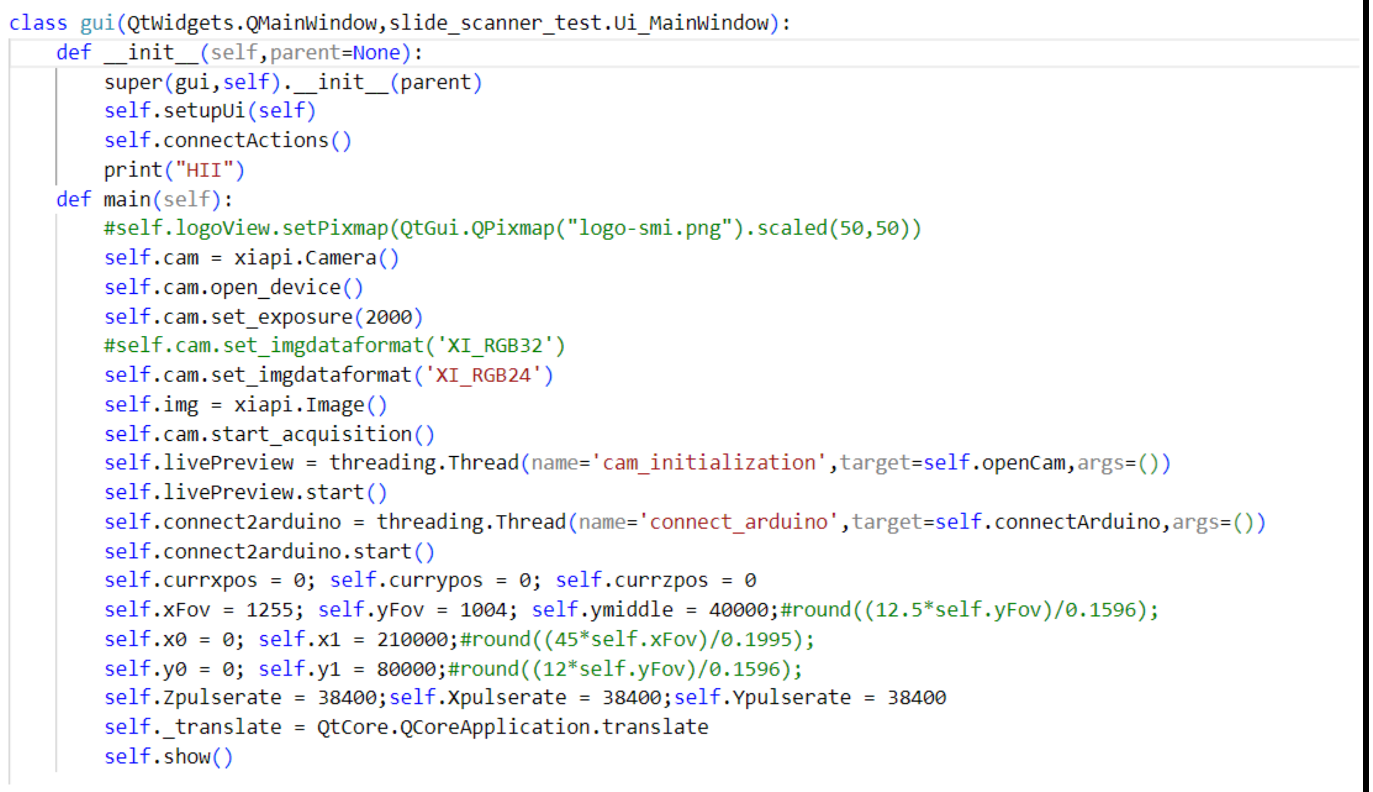
New GUI uses PyQt5, A python GUI was created using PyQt5 module .It consists of 16 buttons (15 functional buttons) and a screen to view the loaded specimen. Six buttons correspond to single step actuation in all the three axes(X,Y,Z). Three buttons correspond to direct stepping using cartesian coordinates in the three axes. One button corresponds to the autofocus functionality. One button allows user to take snapshots of smear under observation.

The snapshots are saved automatically with a timestamp to ensure ease in case of multiple clicks. One button slows for resetting of coordinates of all axes to home (0,0,0). It is noted to be noted here the coordinate setting here is absolute in nature- i.e. if current axis-location is 100 and user wishes to go to axis-location 2000 then the user must simply type 2000 and relevant ‘GoToAxis’ command. To ensure visibility of current coordinates the current position of the slide with respect to origin is always being displayed on the GUI panel. One button has been provided to allow users to change the exposure (in ms). The image can be viewed on the screen in 2 formats: coloured (RGB24) and Black & White (Mono8). The default setting on this GUI is the RGB 24 format. This python GUI commands the microcontroller (Arduino Mega 2560 Rev3) board to actuate the stepper motors and objective lens to achieve user specified objectives. For this purposed a code for controlling the microcontroller was readily available and implemented on the Arduino IDE.



```
def connectActions(self):
    self.pushButton.clicked.connect(self.snapshot)
    self.pushButton_2.clicked.connect(self.autoFocus)
    self.pushButton_3.clicked.connect(self.xPlus)
    self.pushButton_4.clicked.connect(self.yMinus)
    self.pushButton_5.clicked.connect(self.yPlus)
    self.pushButton_6.clicked.connect(self.xMinus)
    self.pushButton_7.clicked.connect(self.zUp)
    self.pushButton_8.clicked.connect(self.zDown)
    self.pushButton_9.clicked.connect(lambda: self.gotoX(self.lineEdit.text()))
    self.pushButton_10.clicked.connect(lambda: self.gotoY(self.lineEdit_3.text()))
    self.pushButton_11.clicked.connect(lambda: self.gotoZ(self.lineEdit_2.text()))
    self.pushButton_12.clicked.connect(lambda: self.setExposure(self.lineEdit_4.text()))
    self.pushButton_13.clicked.connect(self.test)
    self.pushButton_17.clicked.connect(self.bloodsmear)
    self.pushButton_18.clicked.connect(self.resetOrigin)
    self.pushButton_13.clicked.connect(self.areascan)
```

Figure 8: Code for different Push button link for functionalities in GUI



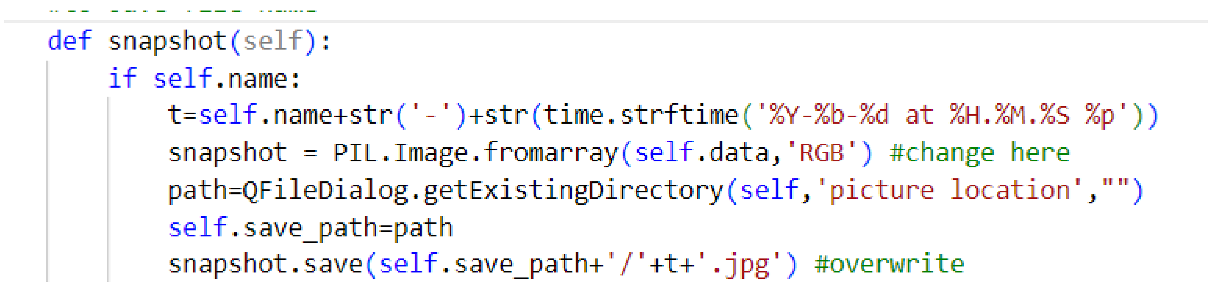
```
class gui(QtWidgets.QMainWindow,slide_scanner_test.Ui_MainWindow):
    def __init__(self,parent=None):
        super(gui,self).__init__(parent)
        self.setupUi(self)
        self.connectActions()
        print("HII")
    def main(self):
        #self.logoView.setPixmap(QtGui.QPixmap("logo-smi.png").scaled(50,50))
        self.cam = xiapi.Camera()
        self.cam.open_device()
        self.cam.set_exposure(2000)
        #self.cam.set_imgdataformat('XI_RGB32')
        self.cam.set_imgdataformat('XI_RGB24')
        self.img = xiapi.Image()
        self.cam.start_acquisition()
        self.livePreview = threading.Thread(name='cam_initialization',target=self.openCam,args=())
        self.livePreview.start()
        self.connect2arduino = threading.Thread(name='connect_arduino',target=self.connectArduino,args=())
        self.connect2arduino.start()
        self.currxpos = 0; self.currypos = 0; self.currzpos = 0
        self.xFov = 1255; self.yFov = 1004; self.ymiddle = 40000;#round((12.5*self.yFov)/0.1596);
        self.x0 = 0; self.x1 = 210000;#round((45*self.xFov)/0.1995);
        self.y0 = 0; self.y1 = 80000;#round((12*self.yFov)/0.1596);
        self.Zpulserate = 38400;self.Xpulserate = 38400;self.Ypulserate = 38400
        self._translate = QtCore.QCoreApplication.translate
        self.show()
```

Figure 9: Code for declaring and linking data member to the devices.

My Work on GUI

i) Saving Name Dialog box

The previous GUI code was lacking option for saving file names as per user convenience and also if it is taking multiple snapshots it was not numbering it, To resolve this I've used PyQt5 famous library QFileDialog to browse the menu to save files according to the user choice of location and name, I've also provided time stamp to be saved along with the user provided name and by default, if the user doesn't provide any name then the system saves the image according to date and time.



```
def snapshot(self):
    if self.name:
        t=self.name+str('-')+str(time.strftime('%Y-%b-%d at %H.%M.%S %p'))
        snapshot = PIL.Image.fromarray(self.data,'RGB') #change here
        path=QFileDialog.getExistingDirectory(self,'picture location',"")
        self.save_path=path
        snapshot.save(self.save_path+'/'+t+'.jpg') #overwrite
```

Figure 10: Code for saving snapshot with a user provided name

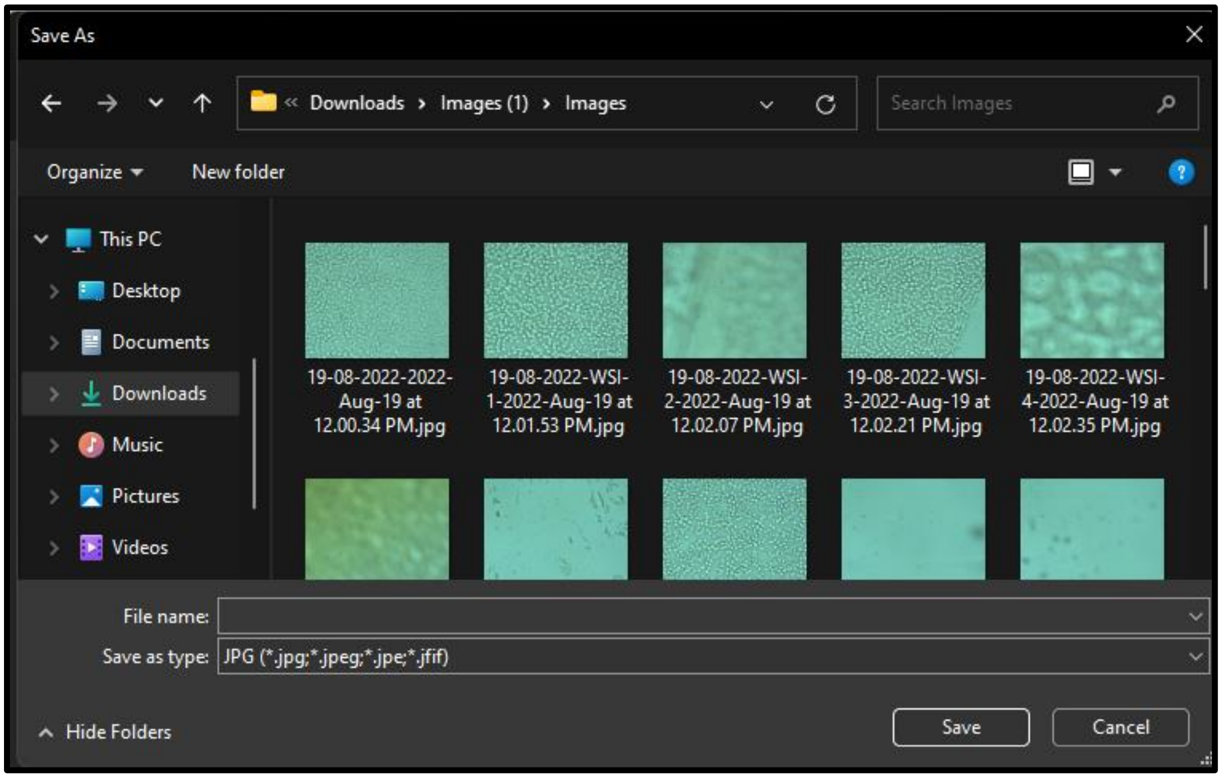

Save As
Downloads › Images (1) › Images
Search Images
Organize
New folder
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
19-08-2022-2022-Aug-19 at 12.00.34 PM.jpg
19-08-2022-WSI-1-2022-Aug-19 at 12.01.53 PM.jpg
19-08-2022-WSI-2-2022-Aug-19 at 12.02.07 PM.jpg
19-08-2022-WSI-3-2022-Aug-19 at 12.02.21 PM.jpg
19-08-2022-WSI-4-2022-Aug-19 at 12.02.35 PM.jpg
File name:
Save as type: JPG (*.jpg;*.jpeg;*.jpe;*.jfif)
Hide Folders
Save
Cancel

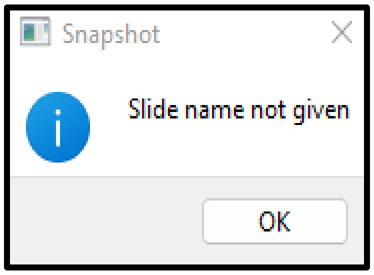
Figure 11: Save file with location and name dialog-box

ii) Pop-up Box

To avoid any issue while saving a name pop-up box is implemented which informs the user if the slide name is not being provided for this, we've to use QMessageBox Which pop-ups when called and informs the user about the issue.

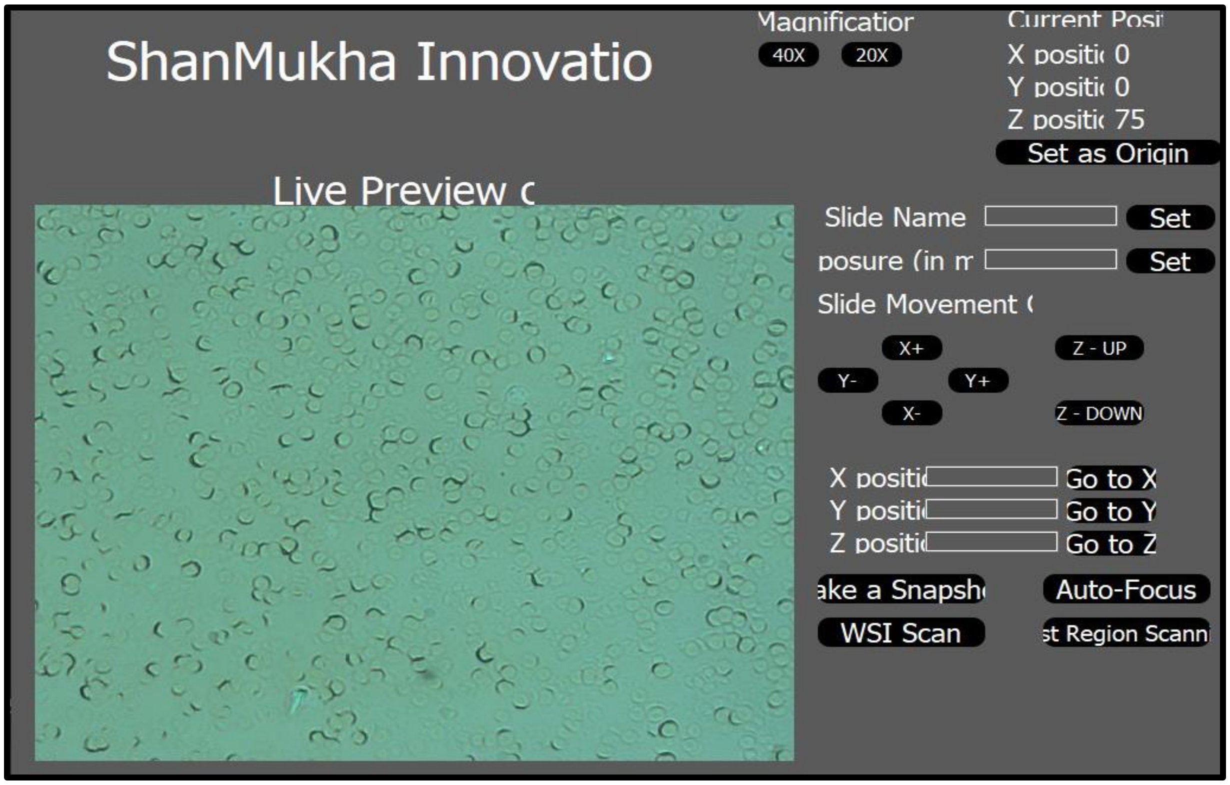
This feature is also used for the WSI image saving technique which wars user to provide a slide name.

```
else:
    mess=QMessageBox()
    mess.setWindowTitle('Snapshot')
    mess.setText('Slide name not given')
    mess.setIcon(QMessageBox.Information)
    x=mess.exec_()
```


Snapshot

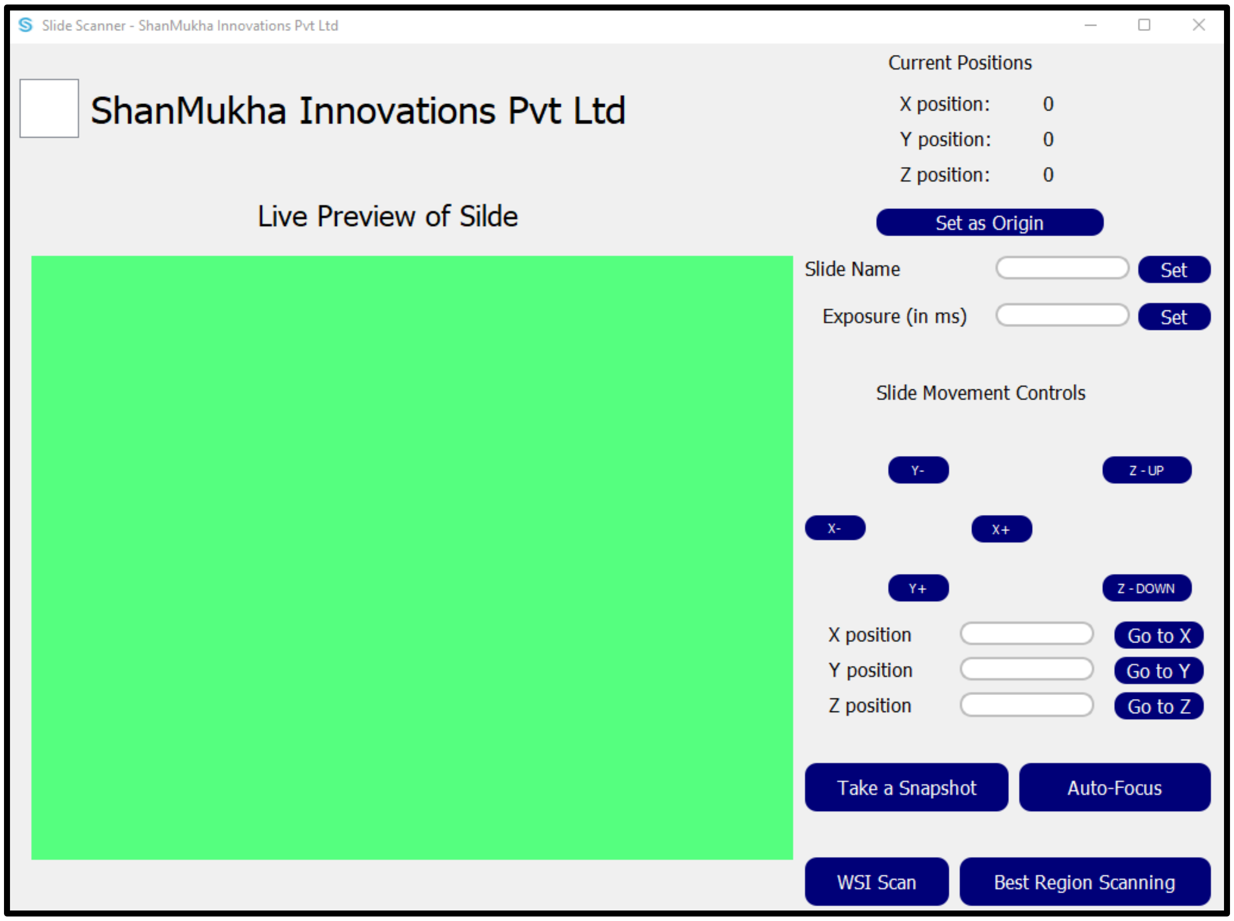
Slide name not given

OK



X positi 0
Y positi 0
Z positi 75

Figure 13: Interface of previous GUI



X position: 0
Y position: 0
Z position: 0

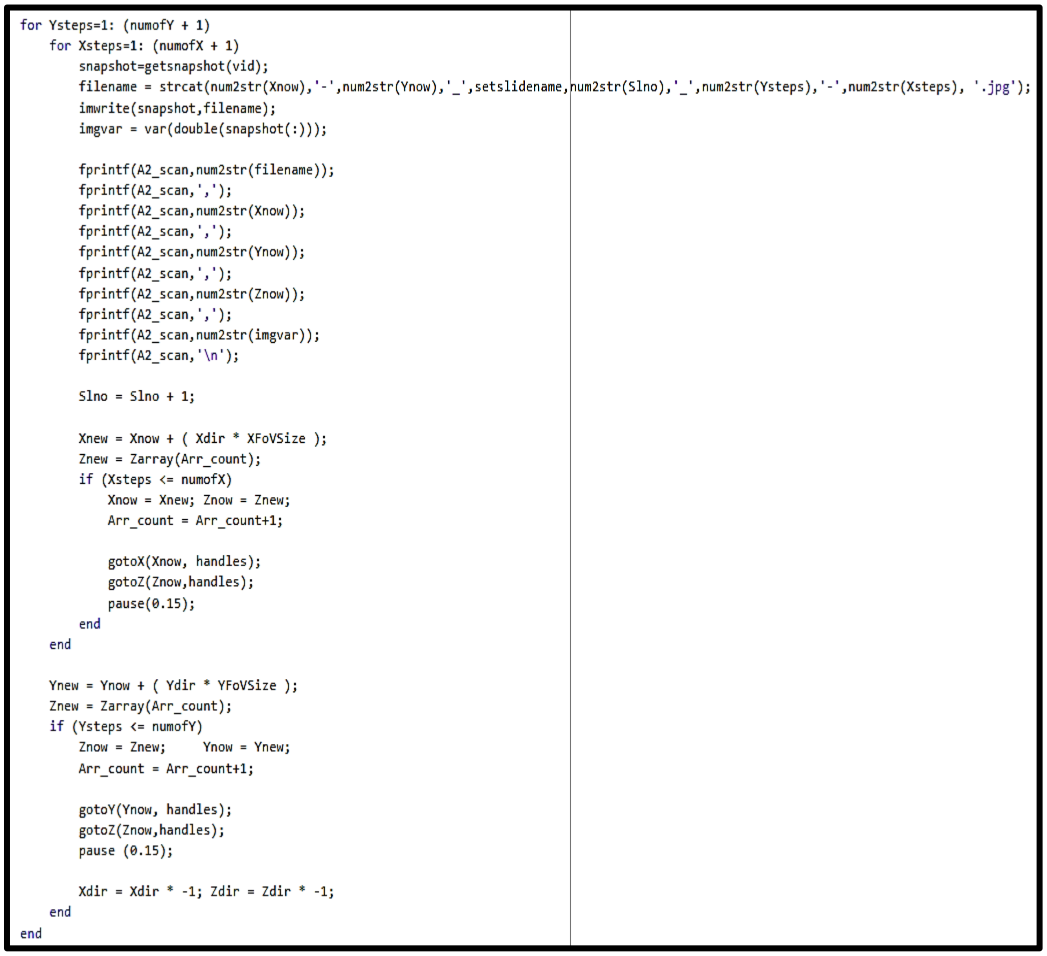
Figure 14: Interface of my GUI

Python .py file to .exe file

To create a standalone executable file from PyQt5 UI and .py file I've used PyInstaller Which is a library that creates a .exe file, the code to convert the .py file to completely standalone .exe I've executed "pyinstaller.exe --OneFile --windowed slide_scanner_gui.py".

WSI

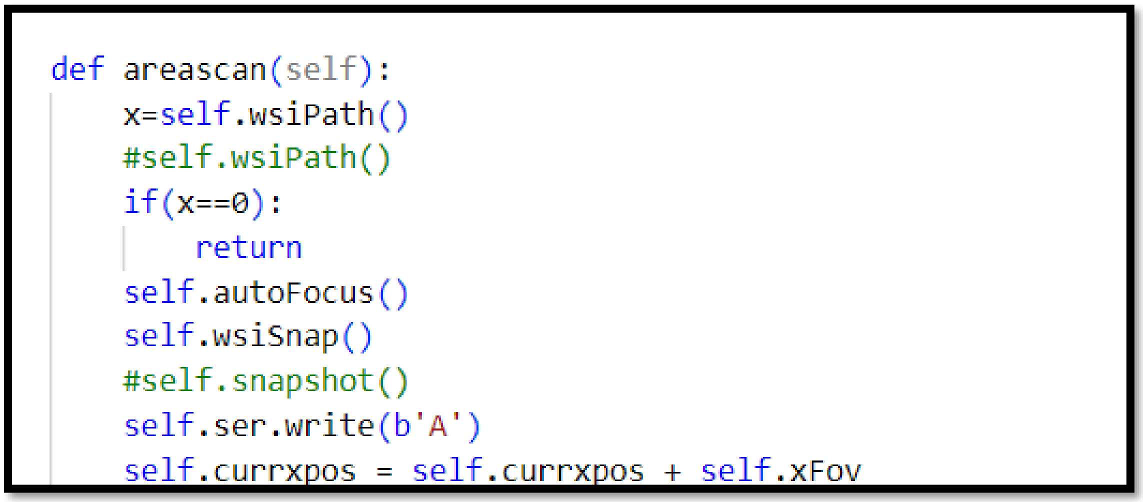
My task for WSI is to convert the existing MATLAB WSI code to python code



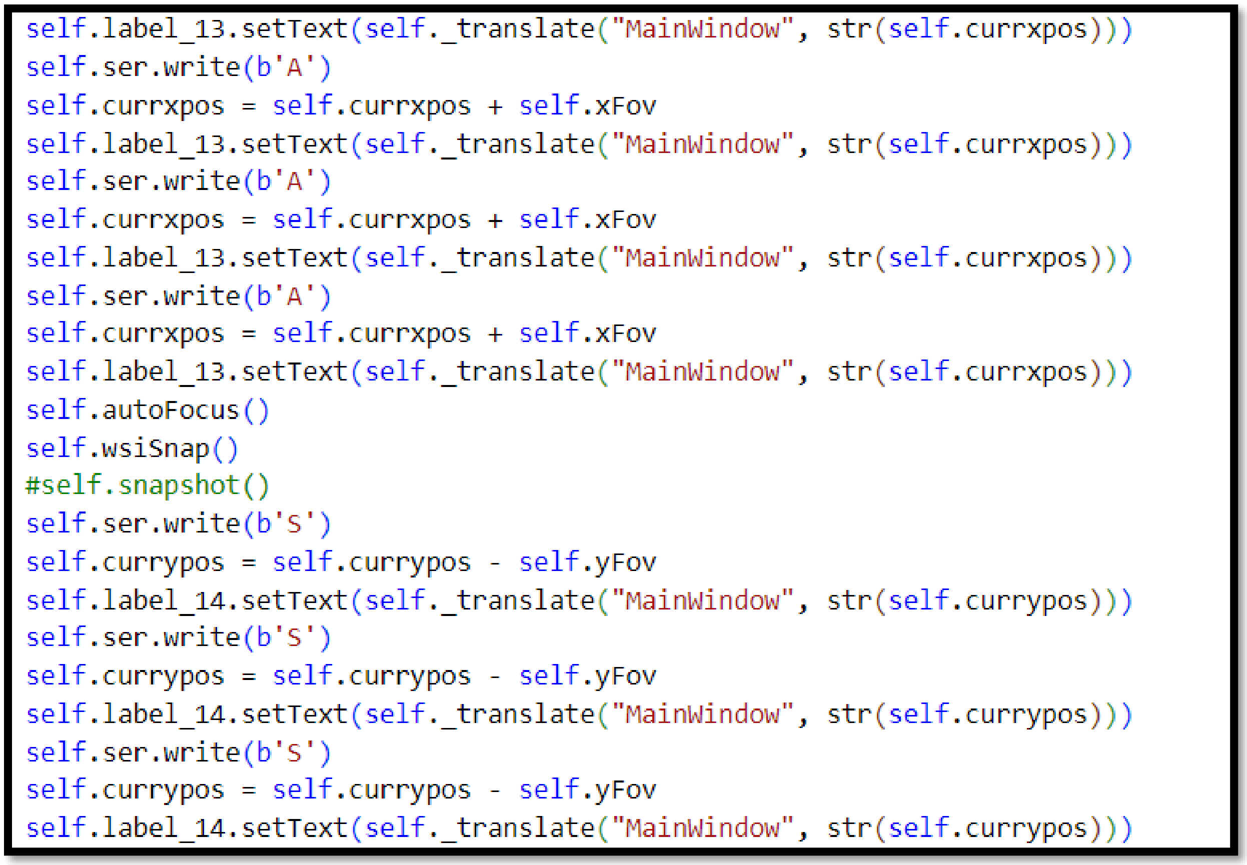
```
for Ysteps=1: (numofY + 1)
    for Xsteps=1: (numofX + 1)
        snapshot=getsnapshot(vid);
        filename = strcat(num2str(Xnow),'-',num2str(Ynow),'_',setslidename,num2str(Slno),'_',num2str(Ysteps),'-',num2str(Xsteps), '.jpg');
        imwrite(snapshot,filename);
        imgvar = var(double(snapshot(:)));

        fprintf(A2_scan,num2str(filename));
        fprintf(A2_scan,',');
        fprintf(A2_scan,num2str(Xnow));
        fprintf(A2_scan,',');
        fprintf(A2_scan,num2str(Ynow));
        fprintf(A2_scan,',');
        fprintf(A2_scan,num2str(Znow));
        fprintf(A2_scan,',');
        fprintf(A2_scan,num2str(imgvar));
        fprintf(A2_scan,'\n');

        Slno = Slno + 1;

        Xnew = Xnow + ( Xdir * XFoVSize );
        Znew = Zarray(Arr_count);
        if (Xsteps <= numofX)
            Xnow = Xnew; Znow = Znew;
            Arr_count = Arr_count+1;

            gotoX(Xnow, handles);
            gotoZ(Znow,handles);
            pause(0.15);
        end
    end

    Ynew = Ynow + ( Ydir * YFoVSize );
    Znew = Zarray(Arr_count);
    if (Ysteps <= numofY)
        Znow = Znew;     Ynow = Ynew;
        Arr_count = Arr_count+1;

        gotoY(Ynow, handles);
        gotoZ(Znow,handles);
        pause (0.15);

        Xdir = Xdir * -1; Zdir = Zdir * -1;
    end
end
```

Figure 15: WSI code for MATLAB

Converted WSI code to Python



```python
def areascan(self):
    x=self.wsiPath()
    #self.wsiPath()
    if(x==0):
        return
    self.autoFocus()
    self.wsiSnap()
    #self.snapshot()
    self.ser.write(b'A')
    self.currxpos = self.currxpos + self.xFov
```



```python
self.label_13.setText(self._translate("MainWindow", str(self.currxpos)))
self.ser.write(b'A')
self.currxpos = self.currxpos + self.xFov
self.label_13.setText(self._translate("MainWindow", str(self.currxpos)))
self.ser.write(b'A')
self.currxpos = self.currxpos + self.xFov
self.label_13.setText(self._translate("MainWindow", str(self.currxpos)))
self.ser.write(b'A')
self.currxpos = self.currxpos + self.xFov
self.label_13.setText(self._translate("MainWindow", str(self.currxpos)))
self.autoFocus()
self.wsiSnap()
#self.snapshot()
self.ser.write(b'S')
self.currypos = self.currypos - self.yFov
self.label_14.setText(self._translate("MainWindow", str(self.currypos)))
self.ser.write(b'S')
self.currypos = self.currypos - self.yFov
self.label_14.setText(self._translate("MainWindow", str(self.currypos)))
self.ser.write(b'S')
self.currypos = self.currypos - self.yFov
self.label_14.setText(self._translate("MainWindow", str(self.currypos)))
self.autoFocus()
self.wsiSnap()
#self.snapshot()
self.ser.write(b'D')
self.currxpos = self.currxpos - self.xFov
self.label_13.setText(self._translate("MainWindow", str(self.currxpos)))
self.ser.write(b'D')
self.currxpos = self.currxpos - self.xFov
self.label_13.setText(self._translate("MainWindow", str(self.currxpos)))
self.ser.write(b'D')
self.currxpos = self.currxpos - self.xFov
self.label_13.setText(self._translate("MainWindow", str(self.currxpos)))
self.ser.write(b'D')
self.currxpos = self.currxpos - self.xFov
self.label_13.setText(self._translate("MainWindow", str(self.currxpos)))
self.autoFocus()
self.wsiSnap()
#self.snapshot()
```

WSI Images

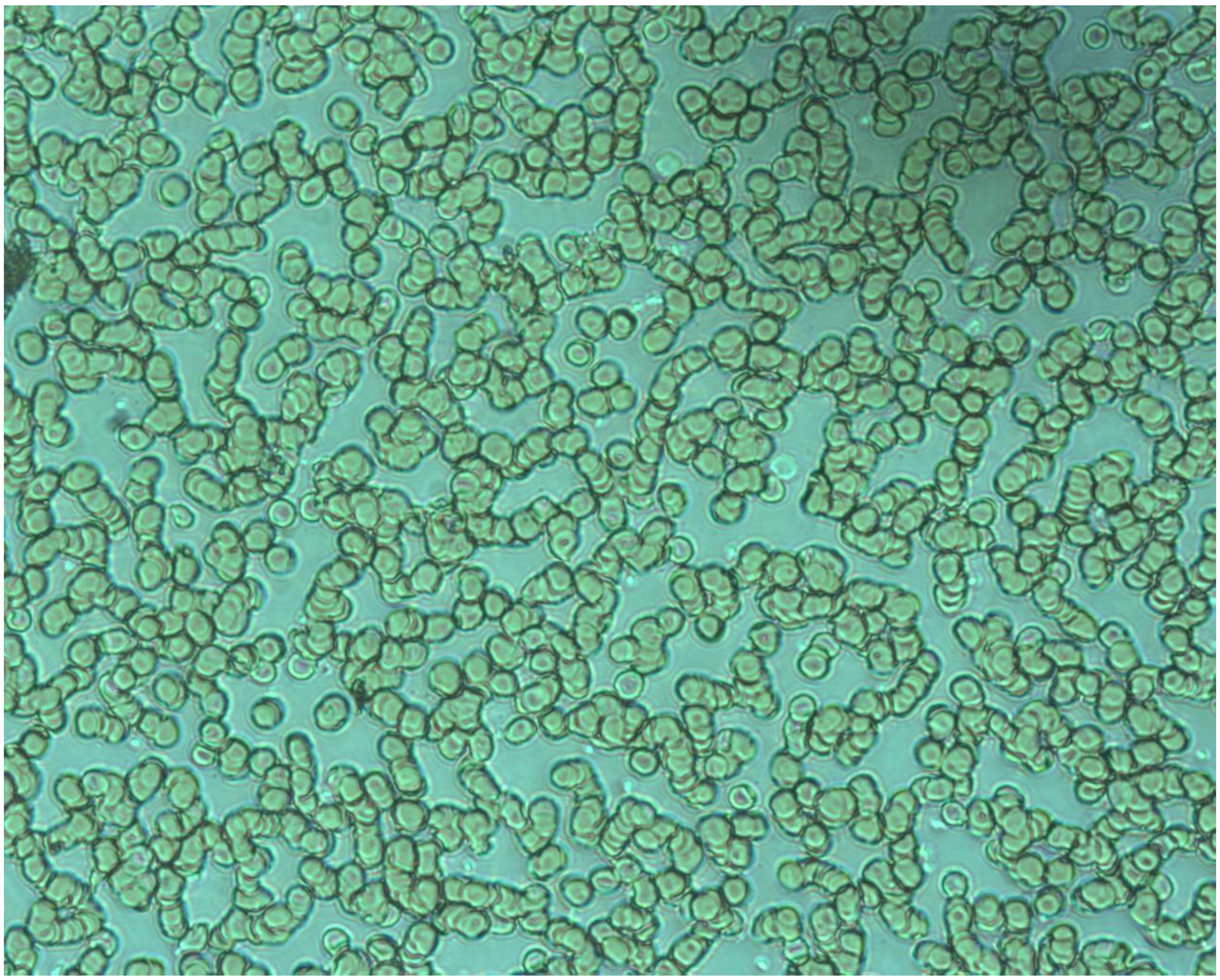


Figure 16: WSI image 1

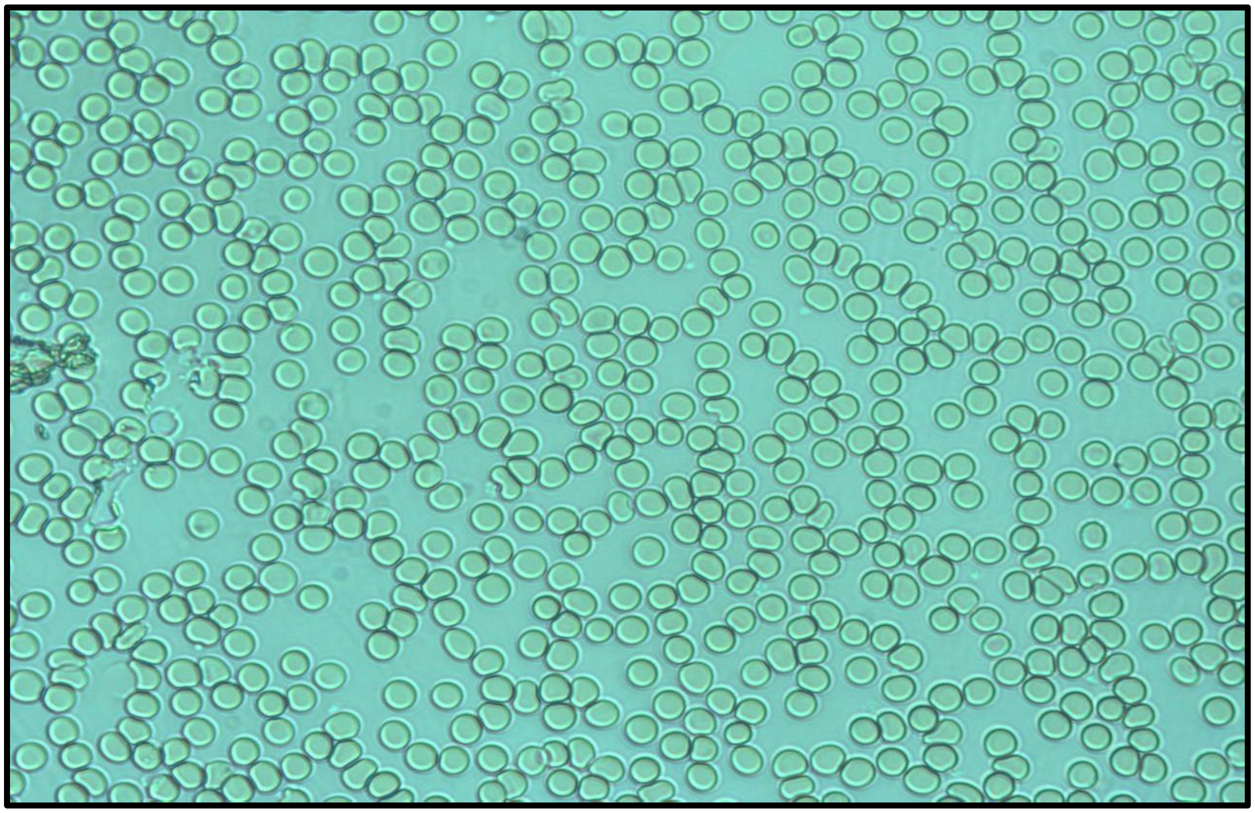


Figure 17: WSI image 2

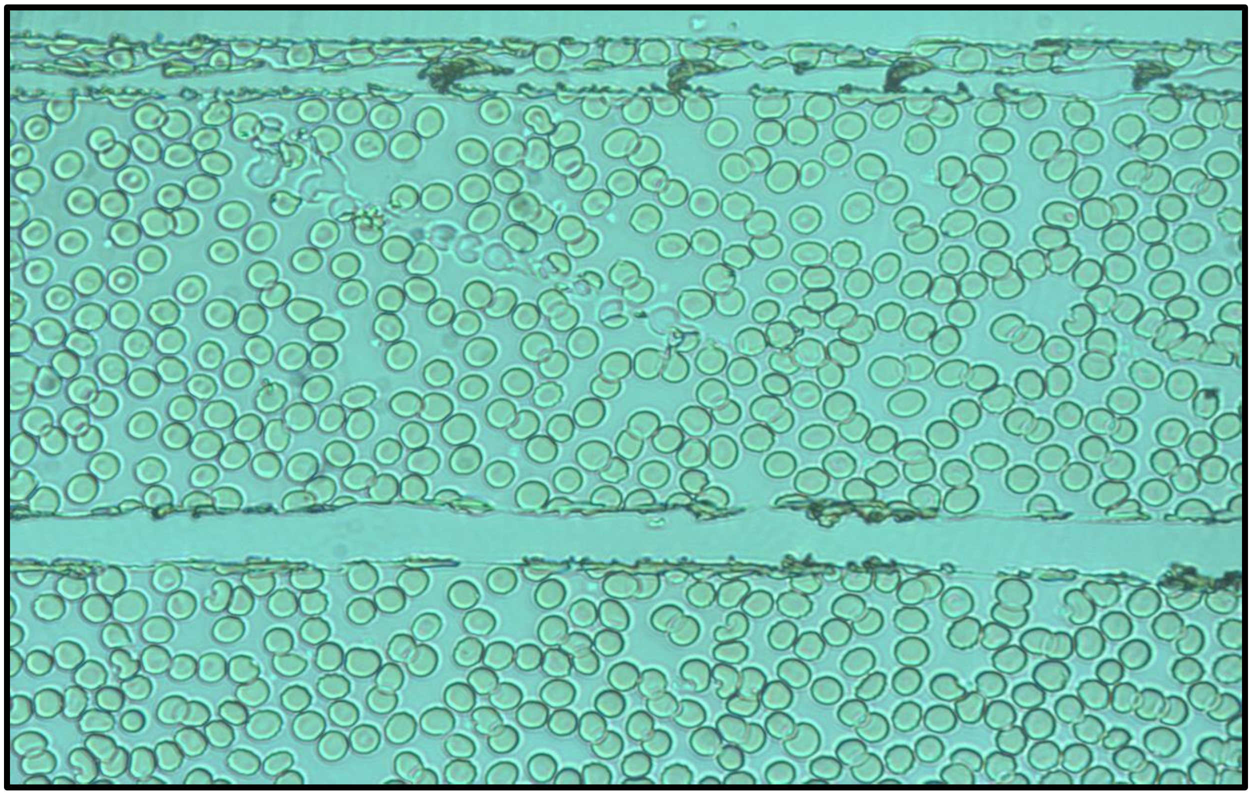


Figure 18: WSI image 3

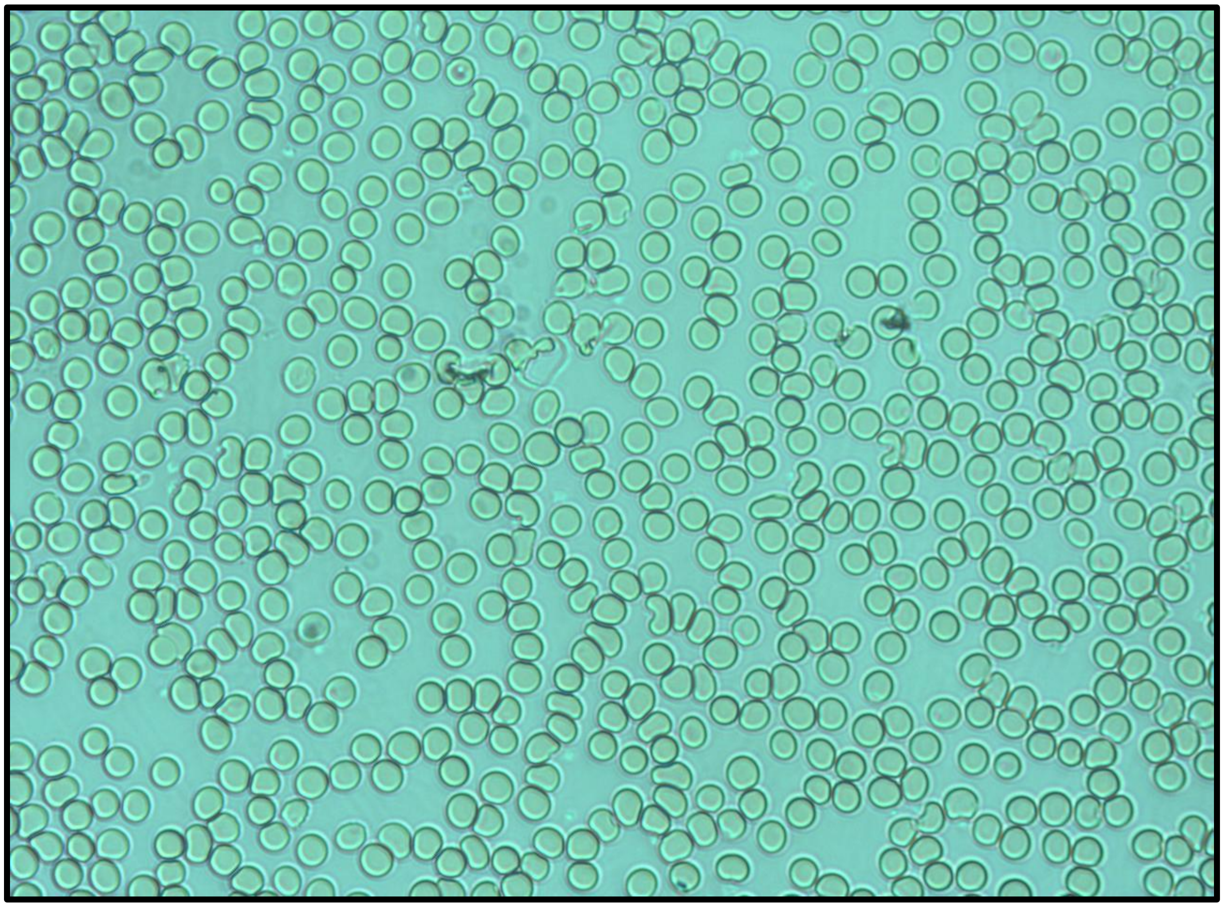


Figure 19: WSI image 4 o

References

[1] Y. Yagi and J. Gilbertson, “Speed, resolution, focus, and depth of field in clinical whole slide imaging applications,” Virtual microscopy and virtual slides in teaching, diagnosis, and research, p. 217, 2005.

[2] M. C. Montalto, R. R. McKay, and R. J. Filkins, “Autofocus methods of whole slide imaging systems and the introduction of a second-generation independent dual sensor scanning method,” Journal of pathology informatics, vol. 2, 2011.

[3] N. Peyman, C. Gregory, S. Aaron, S. Dirk, A. L. Christopher, and P. Cynthia, “Whole slide fluorescence scanner.” Patent No. US9523844B2, 2008.

[4] Katare P, Awasthi N, Venukumar A, Gorthi SS. Low-Cost, Continuous Motion Imaging, Computationally Augmented Whole Slide Imager for Digital Pathology. IEEE Journal of Selected Topics in Quantum Electronics. 2021 Mar 23;27(4):1-7