PROJECT REPORT ON

AI based microscopic image stitching algorithm for characterization of reactor core components

Submitted by

Rinkesh Hemant Paltiwale

Computer Science

Towards Partial Fulfillment of the award of certificate of

Orientation Course for Engineering and Science Graduates (OCES 2023)

Under The Guidance of

Dr. Ratnesh S Sengar

DRHR/RDDG

Human Resource Development Division

Bhabha Atomic Research Centre

Mumbai 400085. INDIA

June 2024

Certificate

This is to certify that **Rinkesh Hemant Paltiwale, Computer Science**, OCES-2023, BARC Training School, Mumbai has completed his Project Work on **AI based microscopic image stitching algorithm for characterization of reactor core components** under my guidance.

Signature _____

Name & Designation_____

Division/Unit_____

Acknowledgements

I would like to express my sincere gratitude to all those who provided me with the opportunity to take up and complete this project.

First and foremost, I am deeply grateful to Dr. Ratnesh S Sengar, my project guide, for their invaluable guidance, encouragement, and constructive feedback throughout the duration of this project. Their expertise, insights and permission to tackle the problem as per my own freedom were instrumental in shaping this work.

I would also like to extend my thanks to Mr. Rohit Sharma, for providing access and knowhow to all the systems in the automation systems lab, which was crucial in understanding how other systems have similarities and differences with my problem statement, allowing a unique viewpoint to my project.

My heartfelt thanks to HRDD and DRHR for providing this unique opportunity to experience the work life and culture at BARC, giving a head start to my scientific career.

Table of Contents

Abstract

Image stitching is the process of combining multiple images with overlapping fields of view to produce a segmented panorama or high-resolution image. Its applications include, but are not limited to, document mosaicing, panorama photography, super resolution imaging and medical imaging.

This project focuses on studying existing image stitching methodologies and building a robust and fast stitching algorithm for the **“In-cell Video Microscopic Imaging System''** using principles of homography, random sampling and linear regression. The developed algorithm is real time, more accurate than existing stitching solutions, and is able to handle huge datasets efficiently. Further, incorporating regression models allows stitching of images without computing homography.

Introduction

Division of Remote Handling and Robotics has indigenously developed, a standalone, compact and real time **“In-cell Video Microscopic Imaging System''** for automated analysis, visual inspection and dimensional measurements of various reactor core components (pressure tubes and low burn up fuels of PHWRs and IPHWRs) at different zoom levels, inside the hot cell. It is a four axis robotic system (X, Y, Zoom, Focus) with imaging capability to resolve details upto two microns. To pair with it, a software has been developed which allows controlling the X-Y plate, zoom and focus axes and the ability to scan samples given a position, range and step size in X and Y . Scan produces a set of images and their combined panoramic image allowing a complete view of the sample. Currently, for creating the panorama, a heuristic approach has been adopted which uses apriori information about the step size, zoom level and pixel count to compute overlapping portions of images and then joins the images. The approach works well for small size scans in one direction and samples which do not have highly varying depths or curvature. In other cases, the panorama consists of various artifacts and inaccurate stitching results.

Image stitching is a widely studied topic in image processing wherein two or more images having overlapping regions are combined, to produce a seamless image with a larger viewing angle and resolution.

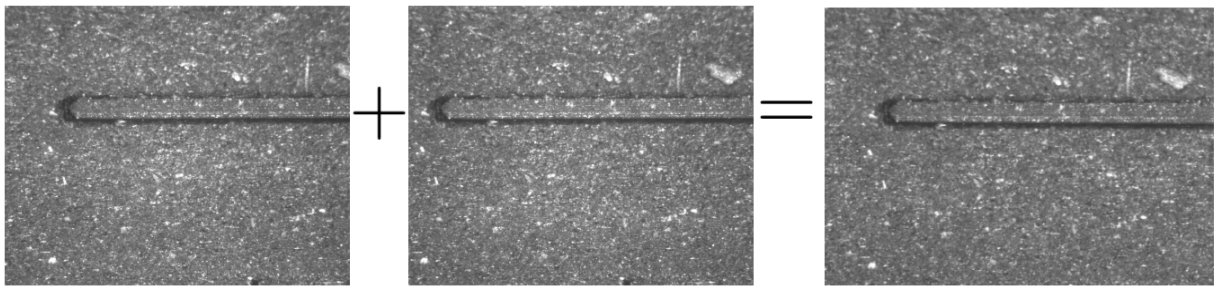

Figure 1: A simple visualization of Image Stitching

Briefly, it consists of three steps i.e.

1. Image registration
2. Image stitching
3. Image fusion/blending

These steps further may contain more steps which are discussed in detail hereafter.

1. Image Registration

Many computer vision tasks require that two or more images of the same object/scene, taken at different times, perspectives or viewing conditions, are aligned w.r.t each other. **Image registration is the task of aligning multiple scenes into a single integrated image**. It helps overcome issues such as image rotation, scale, and skew that are common when overlaying images.

It is the process of transforming different sets of data into one coordinate system. Registration is necessary in order to be able to compare or integrate the data obtained from different measurements/sources (different images of the same scene in our case). Mathematically, given a reference image and a transformed image, registration is the process of finding the transformation matrix that will transform the transformed image coordinate system to the reference image coordinate system.

Feature Based Image Registration

The idea here is to find a set of corresponding points(tie/control points) in two adjacent images to be registered, which can be provided manually or automatically.

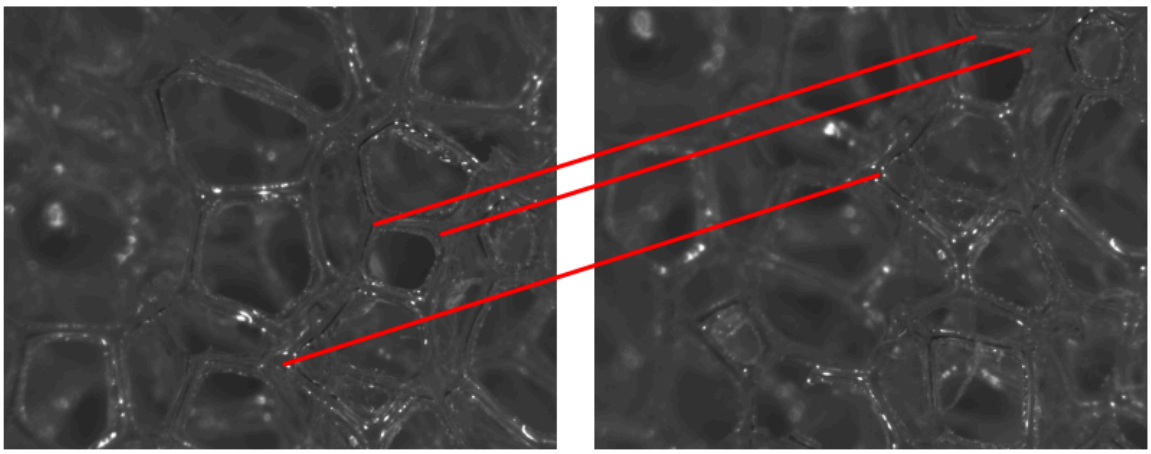

Figure 2: Matching features on images of a sponge

Manually: The user can provide corresponding points or imaging systems can have physical artifacts embedded in imaging sensors which will be present in all images, guiding the location of control points.

Automatic: A feature extractor is used to find correspondence between images which is used as the set of control points.

Knowing the set of control points, the next step is to calculate the projective matrix i.e. homography. Any transformation of the form

$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix} \tag{1}$$

$$\tilde{P}_2 = H \cdot \tilde{P}_1$$

is a projective transformation, also called homography.

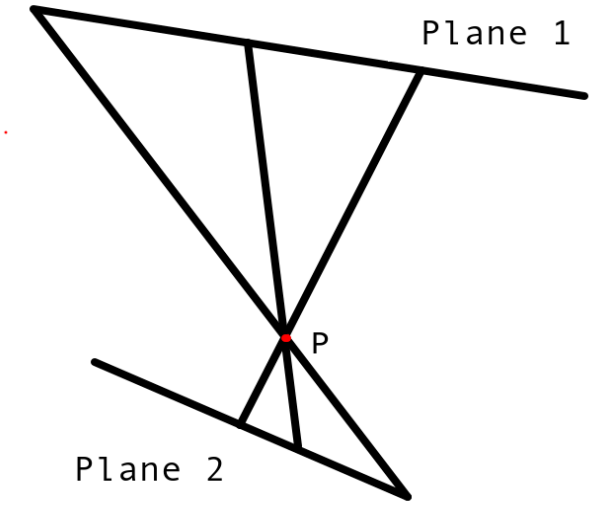

Figure 3: Visualization of projective transform(homography)

The projective matrix maps points on one plane to onto another plane through a point of projection. This is essentially what a camera does when taking images.

A homography is an invertible matrix. Although the matrix has nine entries, it has only eight degrees of freedom, which means that, in practice, there are only eight unknowns. As a convention, we set $h_{33} = 1$

Properties of Projective transformation

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Closed under composition

We can now intuitively prove how homography is useful in image stitching. **By computing the homographies between images and projection plane, we can map all images to a single plane which is useful in planar stitching.**

Two images are said to be related by a homography if any one of the conditions are true

- The scene points lie on a plane
- The image plane should be at practically infinite distance
- Images are taken from the same point(rotation of camera)

Given a set of matching features between two images, we find a homography that best agrees with the matches, as follows

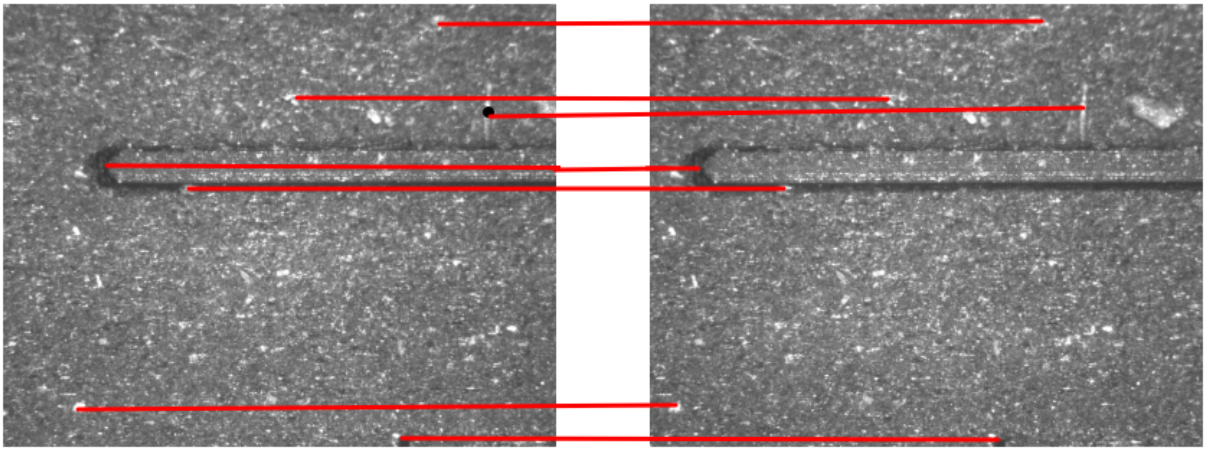

Figure 4: Matching features on images of an incision on a rubber surface

We use the Direct Linear Transform method

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

d and s refer to destination and source image respectively. Rewriting the system of equations in algebraic notation, for a given pair i of corresponding points we have

$$\frac{x_d^{(i)}}{1} = \frac{h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}} \quad (2) \qquad \frac{y_d^{(i)}}{1} = \frac{h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}} \quad (3)$$

Rearranging the terms we get

$$x_d^{(i)}(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}) = h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13} \quad (4)$$

$$y_d^{(i)}(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}) = h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23} \quad (5)$$

Rewriting above equations in matrix notation

$$\begin{bmatrix} x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -x_d^{(i)}y_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (6)$$

Combining equations for all the corresponding points

$$\begin{bmatrix} x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)}x_s^{(1)} & -x_d^{(1)}y_s^{(1)} & -x_d^{(1)} \\ 0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -x_d^{(1)}y_s^{(1)} & -y_d^{(1)}y_s^{(1)} & -y_d^{(1)} \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_s^{(n)} & y_s^{(n)} & 1 & 0 & 0 & 0 & -x_d^{(n)}x_s^{(n)} & -x_d^{(n)}y_s^{(n)} & -x_d^{(n)} \\ 0 & 0 & 0 & x_s^{(n)} & y_s^{(n)} & 1 & -x_d^{(n)}y_s^{(n)} & -y_d^{(n)}y_s^{(n)} & -y_d^{(n)} \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ . \\ 0 \\ 0 \end{bmatrix} \quad (7)$$

Solve for

$$A \cdot h = 0$$

Such that (8)

$$\left\| h^2 \right\| = 1$$

This is an **overdetermined system of equations** and **we are interested in a non trivial solution**.

We define a Constrained Least Squares problem as follows

$$\min_h \left\| Ah^2 \right\|$$

Such that (9)

$$\left\| h^2 \right\| = 1$$

We know that

$$\left\|Ah^2\right\| = (Ah)^T(Ah) = (h^T A^T)(Ah)$$
$$\left\|h^2\right\| = (h)^T(h) = 1 \tag{10}$$

Define a loss function using method of Lagrangian multipliers

$$L(h, \lambda) = h^T A^T Ah - \lambda(h^T h - 1) \tag{11}$$

Differentiating w.r.t h

$$\frac{\partial L(h, \lambda)}{\partial h} = 2A^T Ah - 2\lambda h = 0 \tag{12}$$

We now have an eigenvalue problem

$$A^T Ah = \lambda h \tag{13}$$

The eigenvector with the smallest eigenvalue minimizes the loss function.

This method works well when we have a set of valid correspondences. **A drawback of this is that it is not automatic**. There can be a scenario where 2 features match perfectly in the 2 images but they are not projections of each other, and there is no way of distinguishing a valid match(inlier) from an invalid one(outlier). To tackle this, we rely on a statistical approach **RANSAC(RAndom SAmpling Consensus).**

We need to robustly compute the transformation in the presence of wrong matches. If the number of outliers < 50%, RANSAC can be used.

Steps in RANSAC

1. Randomly choose s samples. s is typically the minimum samples to fit a model. In our case s=4.
2. Fit the model to randomly chosen samples.
3. Count the number M of data points(inliers) that fit the model within a measure of error ε.
4. Repeat steps 1-3 N times.
5. Choose the model that has the largest number M of inliers.

RANSAC will iteratively improve the model and give a close enough approximation to the analytical solution depending on the number of samples and error threshold.

2. Image stitching

Once the homography has been computed, we can apply the perspective transform on the transformed image to align it with the reference image. This amounts to multiplying the image pixels with the homography.

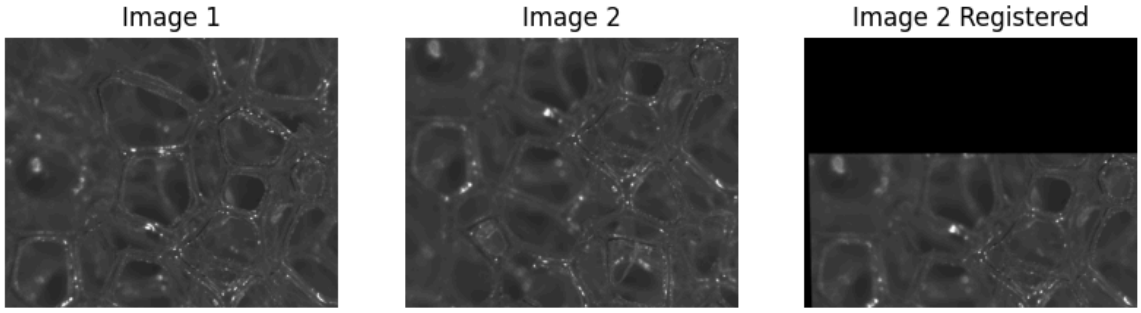

Figure 5: Working of Image Registration

It can be seen that the image after transformation may go out of viewport, resulting in loss of data. We fix this as follows

1. First apply the perspective transform on corners of the image. This will take care of the misalignments due to camera orientation (which is more at corners) and will compensate for perspective transformation.
2. Transformed corners may be negative, we then find the equivalent valid transformation by translating the corners.
3. The transformed image is overlaid on the reference image, creating a panorama.

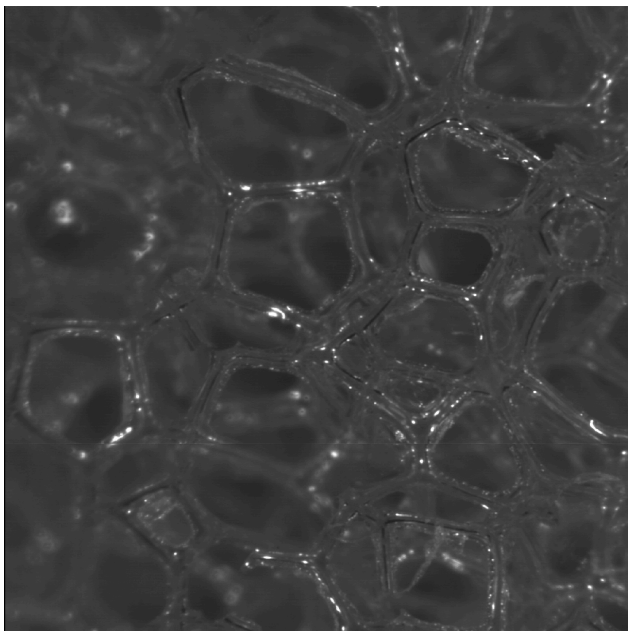

Figure 6: Output of Stitching process

3. Image Blending

No 2 images captured are the same. This is due to the fact that there is a variation in exposure, lighting conditions and motion blur, parallax errors, vignetting etc. The result is that stitching procedure results in visible artifacts known as **seamlines**. Seamlines are rectified by image blending, a technique that allows us to combine 2 or more images to produce a seamless and visually appealing image. Mathematically, blending is adding a fraction of one image with a fraction of another image. The simplest method is Alpha Blending

$$I_{blend} = \alpha I_1 + (1 - \alpha) I_2 + \gamma \tag{14}$$

Adjusting the value of $\alpha(0 \sim 1)$ results in varying blended images. Below figure shows seamlines and the blended result. Another option is to apply a smoothing/low pass filter at the location of seamlines.

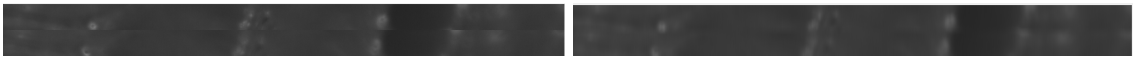

Figure 7(a): Seamlines visible; (b): Blended result

Experimental Procedure

The general stitching pipeline as discussed in the previous section, does solve the problem of stitching for our case but comes with its caveats as follows

- Not near real time performance.
- Computation and memory intensive.
- Feature/Keypoint detection is not very reliable due to the nature of microscopic material surface images.
- Unable to compute homography/wrong homography computation in case of no sufficient overlap or feature detection not working as expected.
- Extra parameters are estimated which actually are fixed in our case.
- The sample being scanned is not flat, resulting in images which are out of focus and not related by only translations.

Existing methods do not leverage the information provided by the imaging system like

- Movement only in X and Y direction in predefined steps.
- Anti vibration plates and mounted camera resulting in no rotation.
- Diffuse lighting source for adequate and uniform illumination while raster scanning a sample
- Fixed zoom level during a scan, resulting in a uniform scaling parameter for all images
- Knowledge of sample location to minimize out of focus sample images resulting in a fair amount of images where homography can be estimated

The perspective transformation for our case is modified as follows

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} \tag{15}$$

This reduces the number of parameters to be estimated from 8 to 2. From discussion in the earlier section, this can be solved by merely a single pair of correspondence.

"findHomography" function from OpenCV2's python implementation was used to find homography between 2 images. Some of the homographies are as follows

$$\begin{bmatrix} 1.00370026e+00 & 2.08444247e-02 & 1.84224656e+00 \\ -5.73224568e-03 & 1.02115683e+00 & 4.36990254e+02 \\ -1.09672245e-05 & 2.18106793e-05 & 1.00000000e+02 \end{bmatrix}$$

$$\begin{bmatrix} 1.00890026e+00 & 3.034562397e-03 & 4.44554652e+00 \\ -6.43223568e-03 & 1.051157583e+00 & 7.26990254e+02 \\ -1.12622265e-05 & -1.18136593e-05 & 1.00000000e+02 \end{bmatrix}$$

The results are in agreement with the assumption.

"findHomography" uses RANSAC and still solves for 8 unknowns and is therefore computation heavy. Solution to this is **not computing the homography for every consecutive pair of images.** Here, **we have randomly chosen a fixed number of pairs depending on the number of images and computed the homography apriori. Multiple pairs are used to make the homography estimation robust. This works because the imaging system ensures precise movement of the sample, so any 2 consecutive images chosen from the scan will be related by approximately the same homography.**

Knowing the movement of the imaging system, the parameters t_x and t_y can't be negative. Furthermore, a value of t_x> width, t_y> height of the image, or any of the fixed parameters deviating from their expected values suggest a wrong homography. In case the feature detector is unable to find a sufficient number of matching keypoints (minimum 4), it throws an error.

Following images are cases where "findHomography fails"

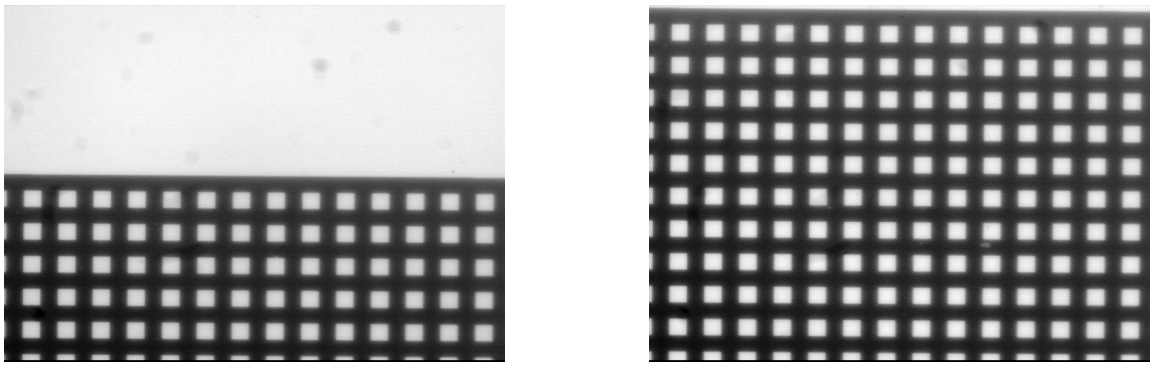

Figure 8(a), (b) : Wrong homography due to mismatch(outliers > inliers)

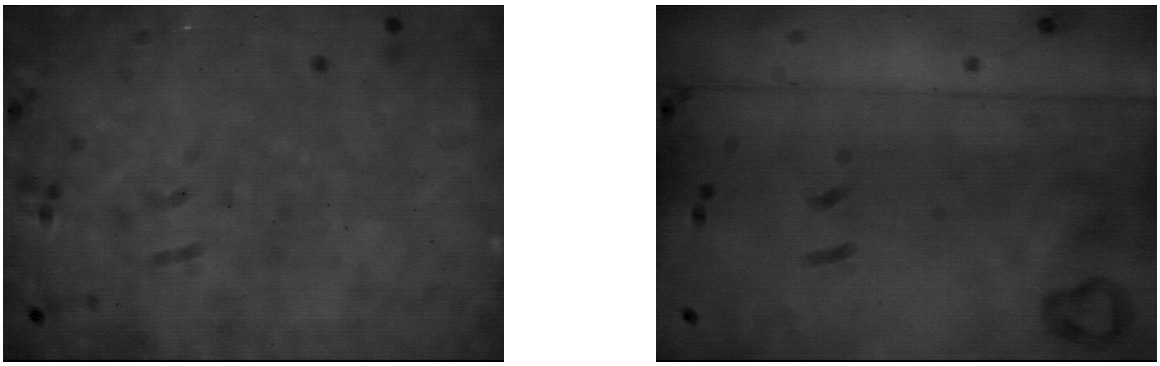

Figure 9(a), (b): Insufficient number of keypoints detected(out of focus area/area without the sample)

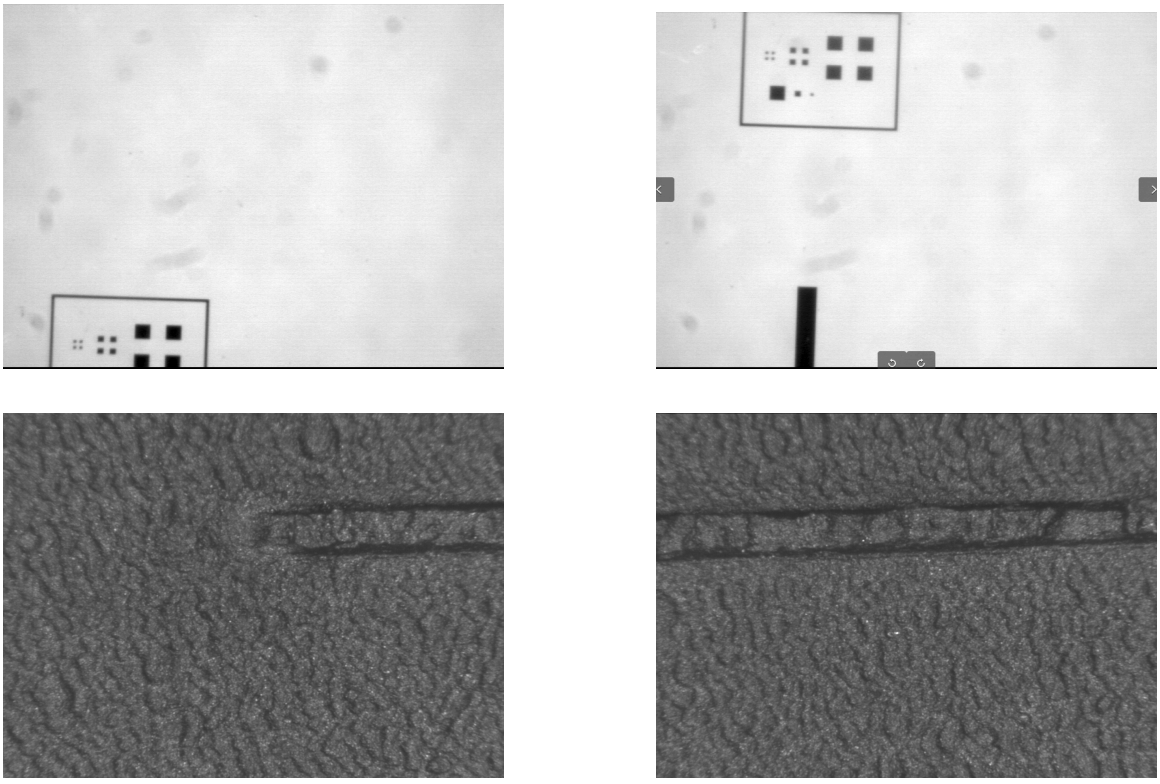

Figure 10(a), (b), (c), (d): Step size too large/Insufficient overlap between images resulting in wrong homography

These cases are frequent in microscopic imaging. **To deal with such cases, a huber regression model has been adopted with the independent variable as Step size (mm) and dependent variable as pixel overlap.**

Huber regression(robust linear regression with huber loss) was chosen as we need the model to be less sensitive to outliers than with mean squared error (MSE) loss but also not completely reject the outliers as with mean absolute error (MAE) loss. Huber loss combines the differentiability of MSE and the ignorance of MAE.

Huber loss is defined as

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases} \tag{16}$$

$$a = y - f(x) \tag{17}$$

δ is a hyperparameter which controls the behavior of loss by giving importance to outliers. A higher value of δ makes it behave like MSE whereas a lower value makes it behave as MAE.

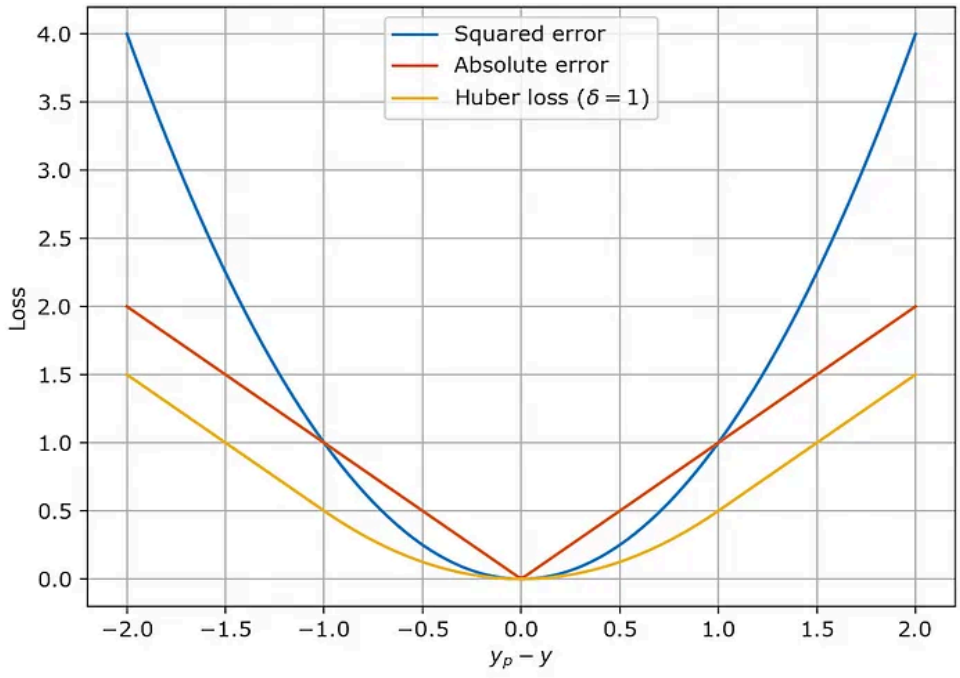

Figure 11: Huber loss

<u>Boundary condition:</u> An overlap of 0 pixels means a step size of 0, i.e the camera is taking images of the same view. This hypothesizes that the y-intercept of all models will be 0.

A set of linear regression models corresponding to different zoom levels(2.5x, 5x, 7x) with pixel translation as dependent variable and step size as independent variable has been trained. **The models are updated whenever a valid homography is computed to make it more robust.**

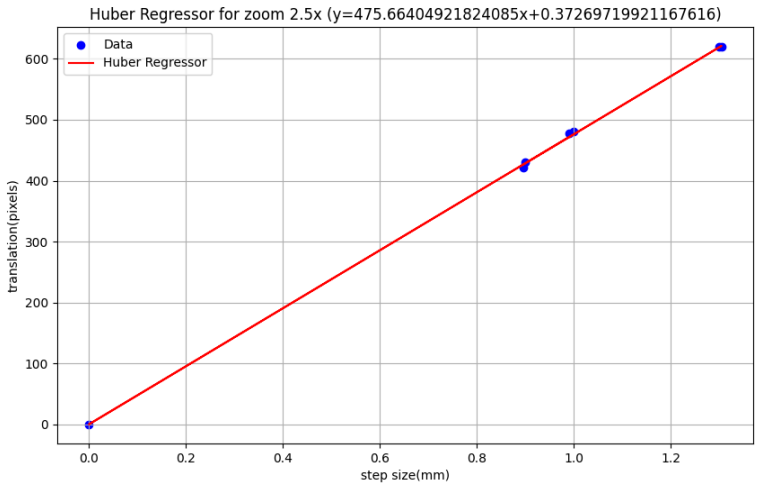

Figure 12: Huber Regressor for zoom 2.5x

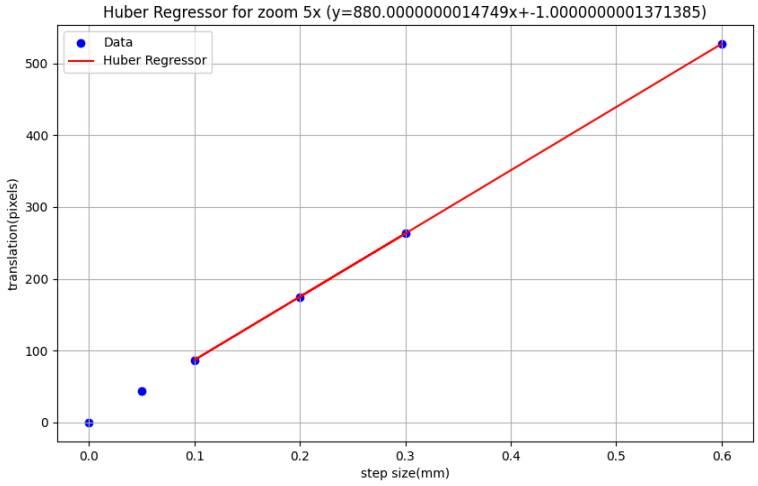

Figure 13: Huber Regressor for zoom 5x

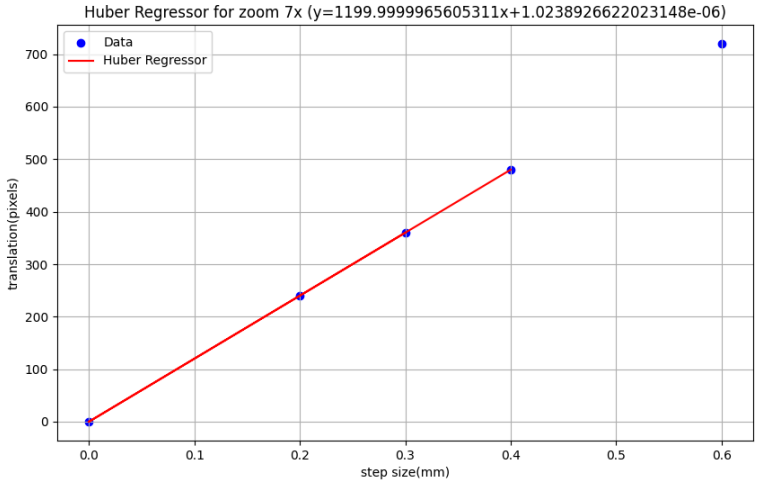

Figure 14: Huber Regressor for zoom 7x

Table 1: Model evaluation parameters for training data			
zoom	**MSE**	**MAE**	**R^2**
2.5	13.600856593413624	2.8779052108033234	0.9996839205086765
5	1.6937243679713024 e-19	3.0533797712450905 e-10	1.0
7	3.9074978445973184 e-13	4.6124518278580255 e-07	1.0

Table 2: Model evaluation parameters for test data			
zoom	**MSE**	**MAE**	**R^2**
2.5	8.652586545324153	2.6136461998113703	0.9982341660111583
5	1.0000000002005331	1.0000000001002665	0.9979338842971064
7	5.970278824710847e -13	6.87893788153815e-07	1.0

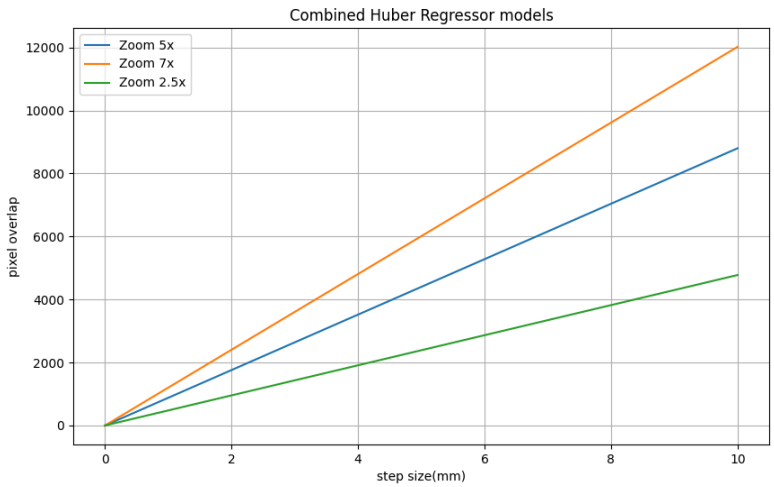

Figure 15: Combined Huber Regressor models

The observations are in agreement with the boundary condition hypothesis. We can neglect the y-intercept values (since a variation of ~1 pixel isn't distinguishable) and observe the trend of slope of model w.r.t zoom levels. From the combined plot, **we can visualize changing the zoom level as simply changing the slope of the line(rotating the line about the point passing through the origin).**

Table 3: Zoom vs Slope of Huber Regressor	
zoom	**slope of huber regressor**
2.5	475.66404922
5	880
7	1199.99999656

There is a strong correlation, and a linear relationship between zoom and slope of huber regressor. Going a step further, this can be modeled as a linear regression problem with MSE loss. MSE loss is chosen since in the previous stage Huber loss has taken care of dealing with outliers and intuitively, there are no outliers present in this data.

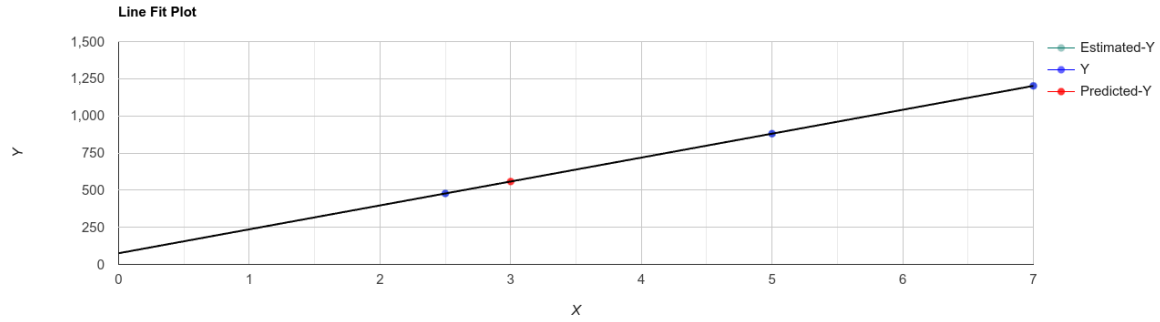

Figure 16: Zoom vs Slope of Huber Regressor

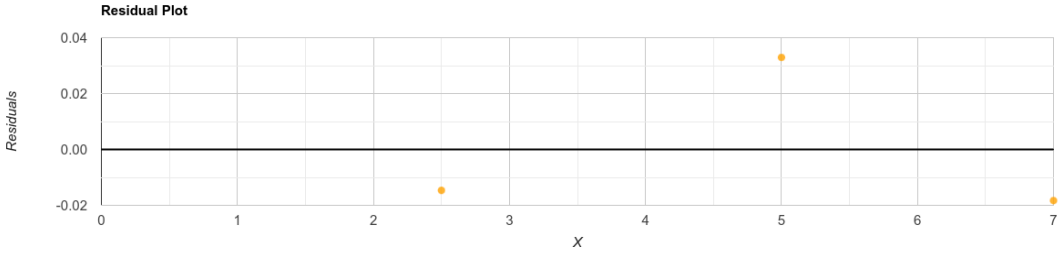

Figure 17: Residual plot of Zoom vs Slope of Huber Regressor

$$y = 73.7499 + 160.9951x$$ (18) (Current equation for Zoom vs Slope of Huber Regressor)

As done previously, this model too is updated when a model corresponding to a new zoom level is trained. We can now estimate the translation between 2 images when homography estimation fails/results in errors as follows

Algorithm 1 Estimate Translation

Require: $zoom \geq 2.5$
Require: $step\_size \geq 0$
Require: $image\_size \geq 0$
$huber\_slope \leftarrow zoom\_huber\_predict(zoom)$ ▷ given zoom level, predict its huber slope using the linear regression model
$translation \leftarrow huber\_slope \times step\_size$
if $translation > image\_size$ **then return** $image\_size$
else return $translation$
end if

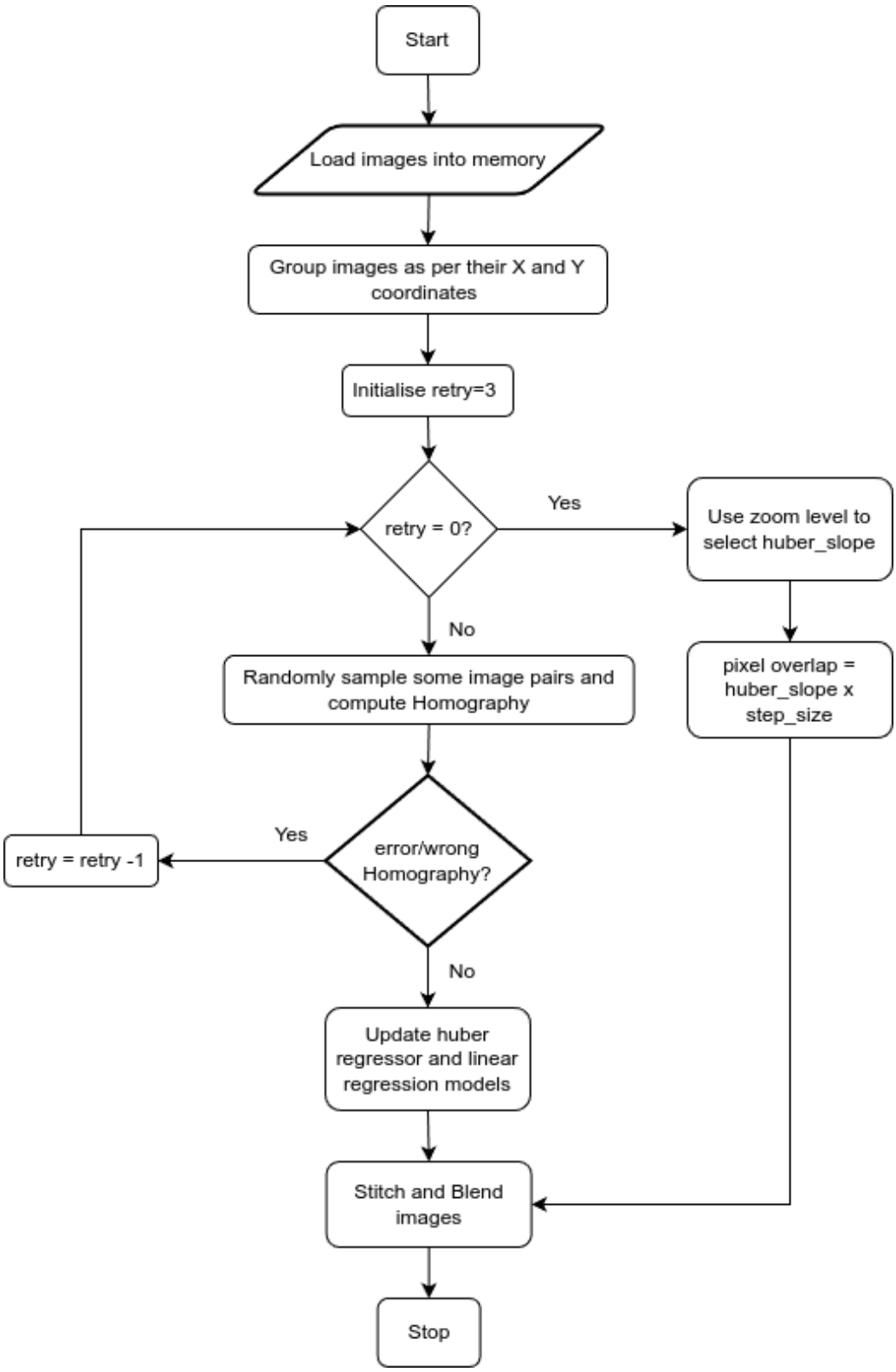

Figure 18: Flowchart for complete Image Stitching

Results and Conclusions

Table 4: Pixel translation computed from homography vs regression			
zoom	**step size**	**pixel translation**	
3	0.75mm	homography	415
		regression	417
2.5	0.5mm	homography	240
		regression	239
3.5	1.25mm	homography	790
		regression	796
7	0.6mm	homography	720
		regression	721
2.5	1.3mm	homography	620
		regression	620

Regression model results are found to be in agreement with the homography computations.

Stitching results with homography computed

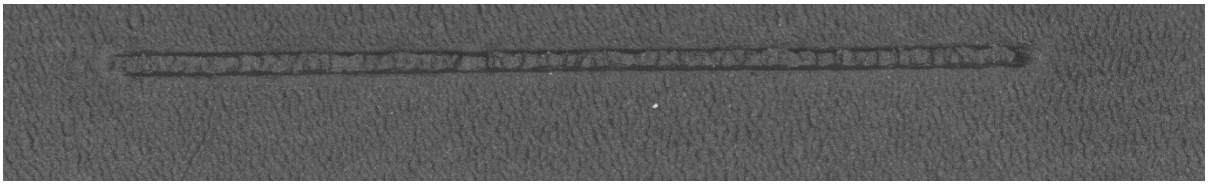

Figure 19: Step size = 0.05mm Zoom = 7x

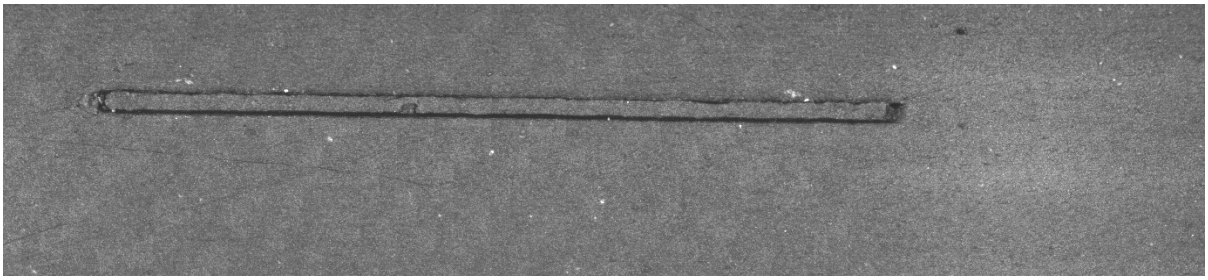

Figure 20: Step size = 0.5mm Zoom = 5x

Figure 21: Step size = 0.75mm Zoom = 2.5x

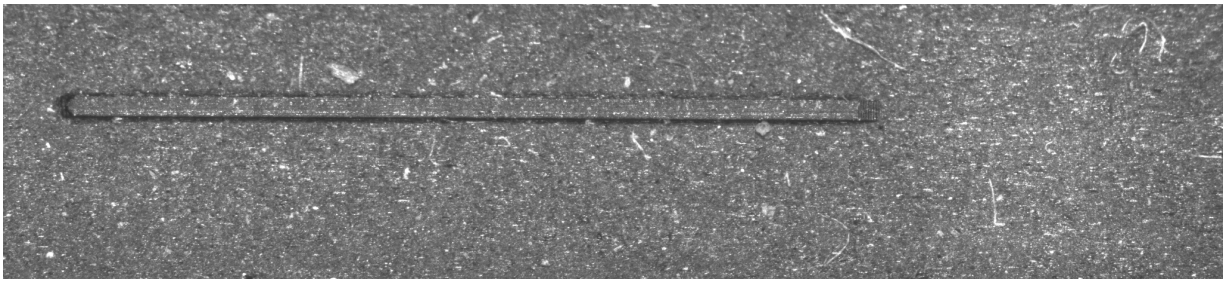

Figure 22: Step size = 0.5mm Zoom = 5x

Figure 23: Step size = 1mm Zoom = 2.5x

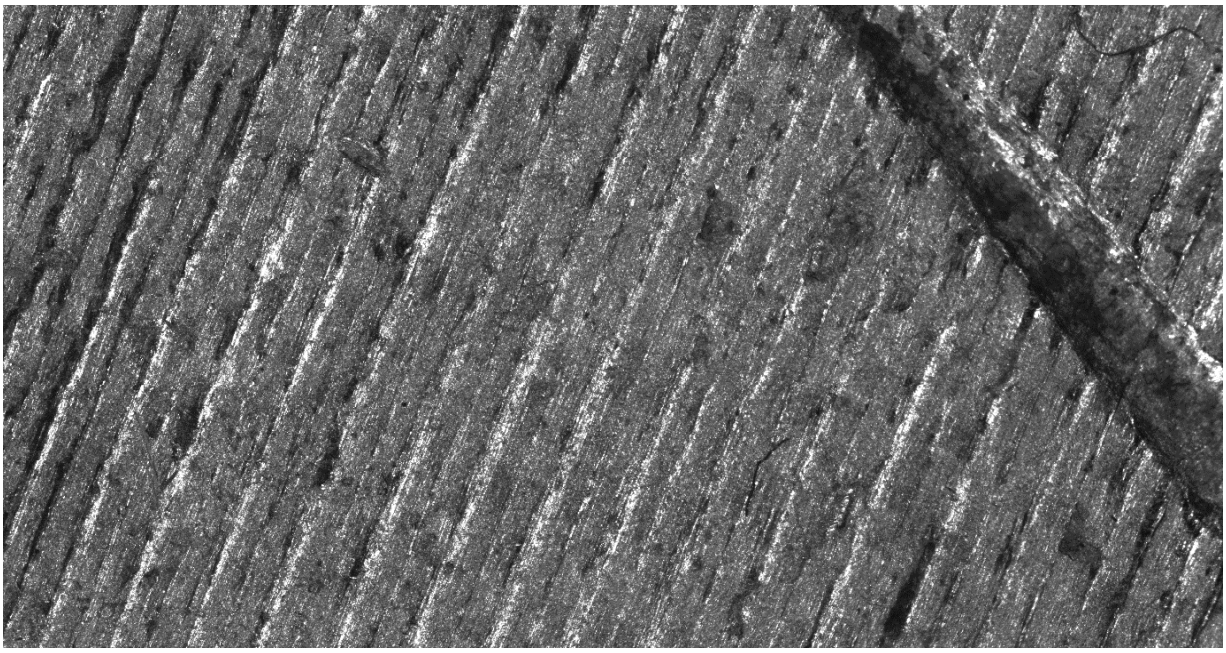

Figure 24: Step size = 0.75mm Zoom = 3x

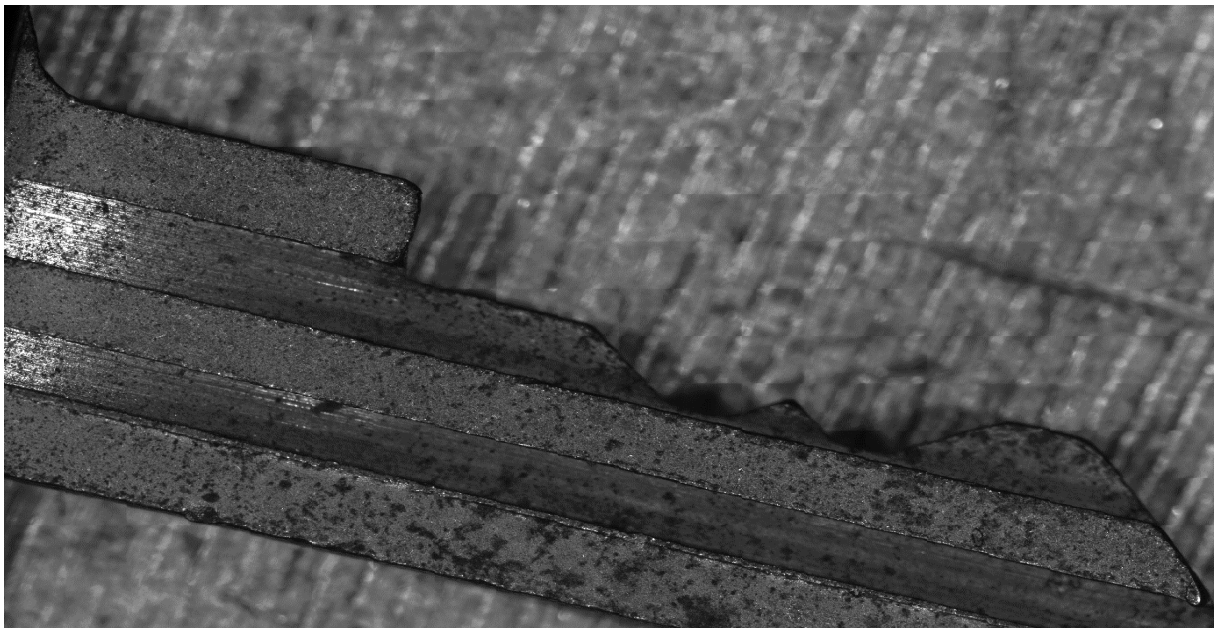

Figure 25: Step size = 1mm Zoom = 3x

Stitching results with homography estimated using regression

Figure 26: Step size = 1.5mm Zoom = 2.5x

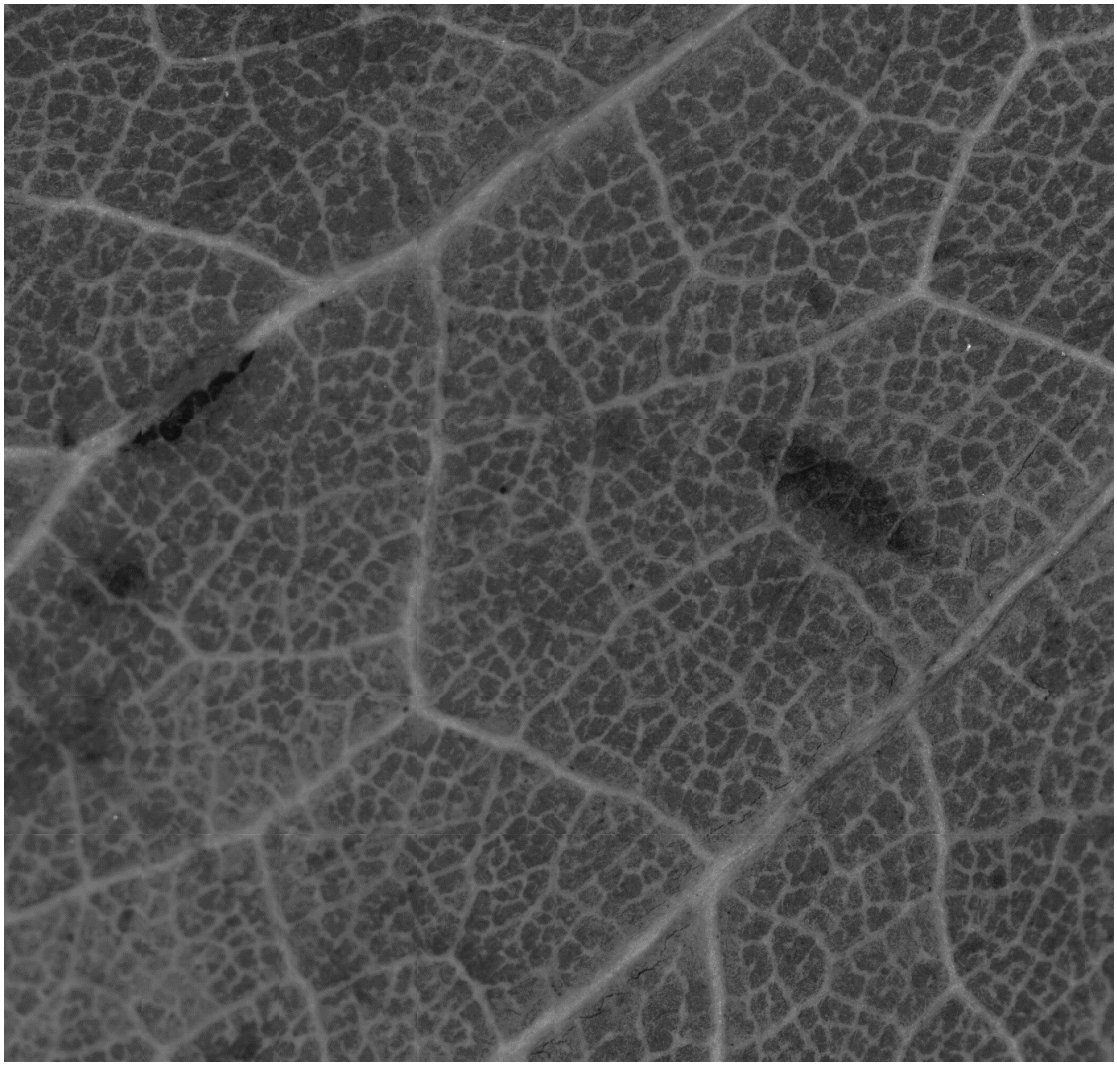

Figure 27: Step size = 1.3mm Zoom = 3x

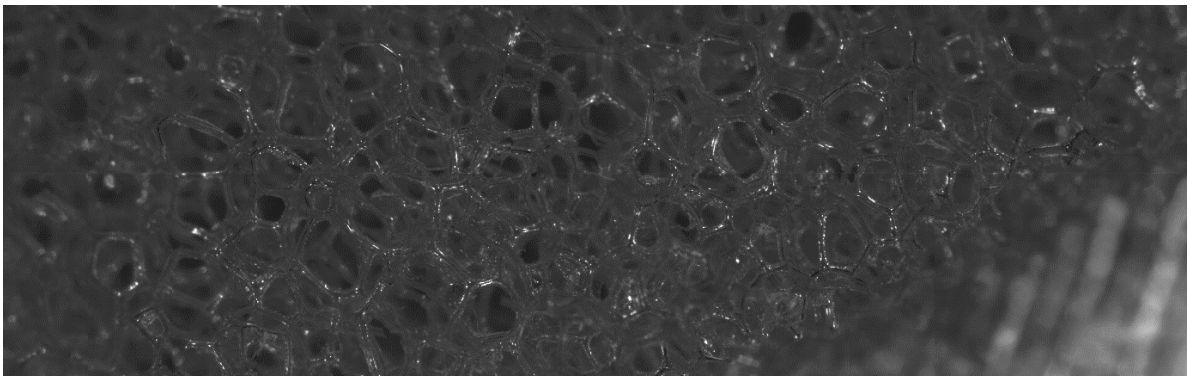

Figure 28: Step size = 1.5mm Zoom = 2.5x

The regression models can result in a pixel translation > image size, which suggests that the step size was large enough to skip some part of the sample. In such cases simple concatenation of images is done. Gaussian blur with kernel size of 1x5 and 5x1 with σ = 10 was used for removing seamlines.

Stitching results with concatenation

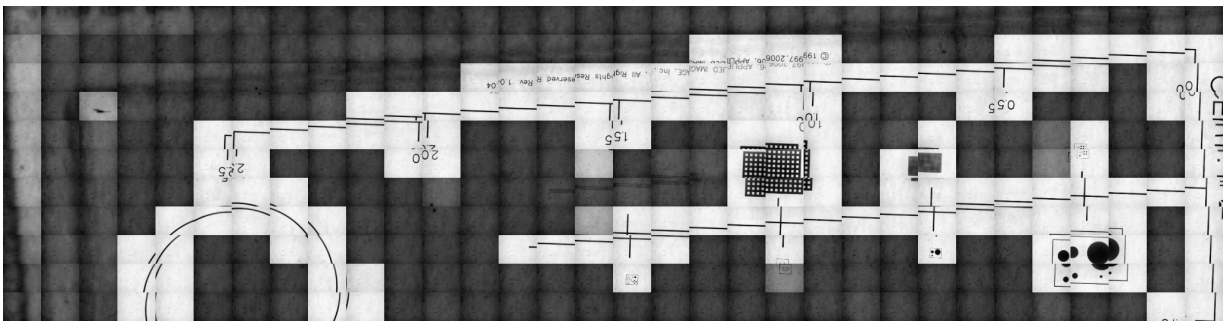

Figure 29: Step size = unknown Zoom = unknown

Measuring dimensions of sample

Knowing the pixel translation, we have the relation

$$\begin{aligned} stitched\ image\ size\ &=\ image\ size\ +\ (n-1)\ *\ pixel\ translation \\ movement\ of\ the\ sample\ &=\ (n-1)\ *\ step\ size\ (mm) \end{aligned} \tag{19}$$

Pixel to Distance(mm) ratio can be found which will determine the dimensions of area scanned.

Estimation of optimum step size for minimum images and homography computation

Homography was correctly estimated for $step\ size\ <\ 1.3mm$

Performance benchmark

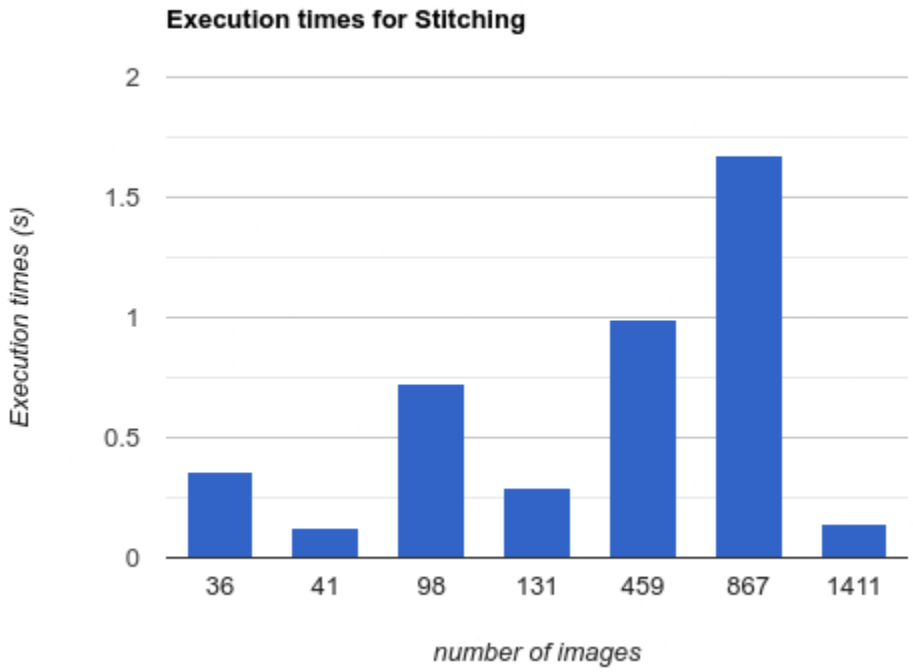

Figure 30: Execution times vs number of images

The erratic behavior is attributed to different types of images and when homography is not computed. The algorithm was able to handle a wide range of images efficiently. **The algorithm was executed on a system running Windows 10 with 12th Gen Intel(R) Core(™) i9 12900k@3.20Ghz with 128Gb of ram.**

Conclusion

A self learning, accurate and real time stitching algorithm was developed for the **"In-cell Video Microscopic Imaging System''. The algorithm was implemented in Python3 and used OpenCV, scikit-learn and matplotlib packages for homography computation, regression and visualization respectively.** The models were trained on synthetically generated image patches and images of various surfaces acquired from the imaging system. Its performance was found to be much better than that of existing solutions, in terms of scaling and execution times. **The simplicity of the proposed algorithm makes it suitable for implementation in a parallel processing paradigm, further increasing its scalability.**

References

1. Deep Learning & Robotics for Biomedical, Nuclear Applications, November-December 2022 BARC newsletter, Ratnesh S. Sengar*, R. Sharma, S. Mishra and A. K. Upadhyay
2. In-cell Video Microscopic Imaging System operation manual
3. https://medium.com/@navekshasood/image-stitching-to-create-a-panorama-5e030ecc8f7
4. https://cs205-stitching.github.io/ (Real time image stitching)
5. https://cmsc426.github.io/pano/(Panorama stitching)
6. https://viso.ai/computer-vision/image-registration/ (Image registration)
7. https://ai.stackexchange.com/questions/21042/how-do-you-find-the-homography-matrix-given-4-points-in-both-images (How do you find the homography matrix given 4 points in both images?)
8. https://dsp.stackexchange.com/questions/40289/why-do-we-need-4-points-for-homography-but-7-8-points-for-fundamental-matrix-cal(Why do we need 4 points for homography but 7/8 points for fundamental matrix calculation?)
9. https://stackoverflow.com/questions/17114880/up-to-a-scale-factor(Up to a scale factor)
10. https://www.youtube.com/watch?v=I_qjO4cM74o&ab_channel=FirstPrinciplesofComputerVision (Computing Homography | Image Stitching)
11. Image Blending, 15-463: Computational Photography, Alexei Efros, CMU, Spring 2010
12. https://www.astroml.org/book_figures/chapter8/fig_huber_loss.html (Huber loss function)
13. https://medium.com/@iqra.bismi/different-loss-functions-used-in-regression-cfb875bf5561 (Different Loss Functions used in Regression)