Fahim Tanvir

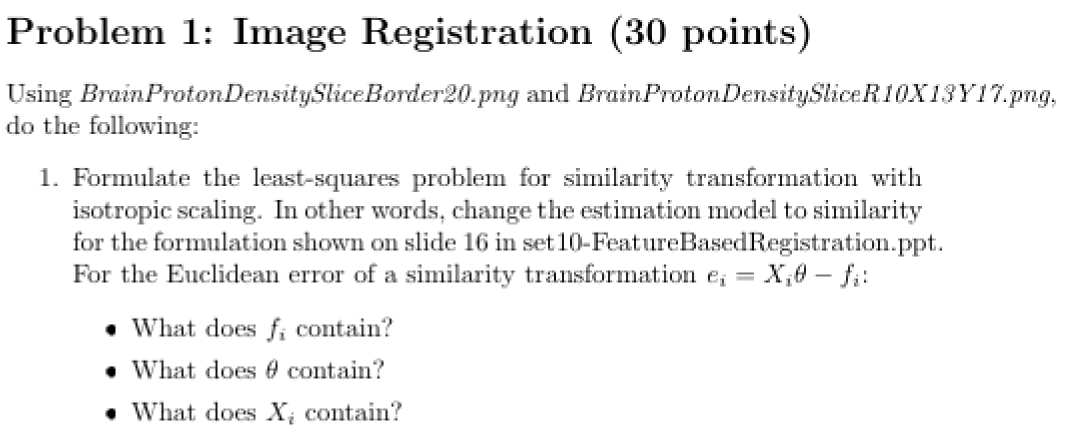
Ahmed Ali

CSCI 367

Homework 3

Introduction:

At this point of the course, we learned about many ways to modify and change details and features about images using filters and feature registration. In this assignment, we are given images regarding internal analysis of a human brain and yeast cells, and modify those images using image registration, similarity transformation, and segmentation. This allows us to give us perspective how image processing is used in the medical as medical examiners need to be able to properly access detail by separating different features from an image.



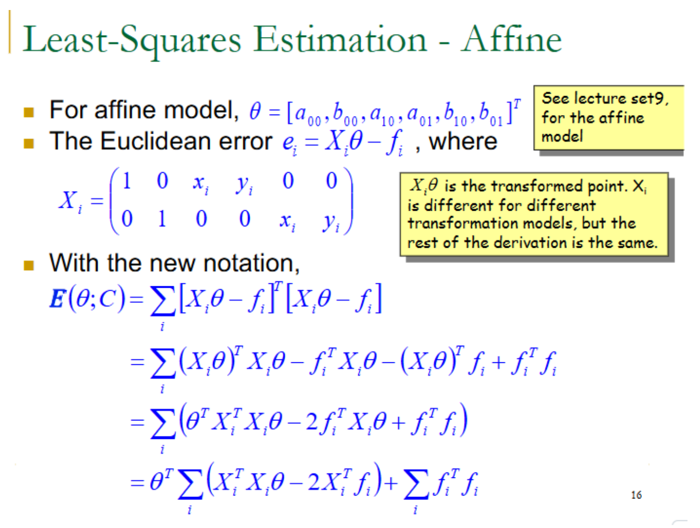
Problem 1: Image Registration (30 points)

Using *BrainProtonDensitySliceBorder20.png* and *BrainProtonDensitySliceR10X13Y17.png*, do the following:

1. Formulate the least-squares problem for similarity transformation with isotropic scaling. In other words, change the estimation model to similarity for the formulation shown on slide 16 in set10-FeatureBasedRegistration.ppt. For the Euclidean error of a similarity transformation $e_i = X_i\theta - f_i$:
   - What does f_i contain?
   - What does θ contain?
   - What does X_i contain?

Algorithm:

The slide referenced by the homework shows this algorithm:



Least-Squares Estimation - Affine

- For affine model, $\theta = [a_{00}, b_{00}, a_{10}, a_{01}, b_{10}, b_{01}]^T$
- The Euclidean error $e_i = X_i\theta - f_i$, where

$$X_i = \begin{pmatrix} 1 & 0 & x_i & y_i & 0 & 0 \\ 0 & 1 & 0 & 0 & x_i & y_i \end{pmatrix}$$

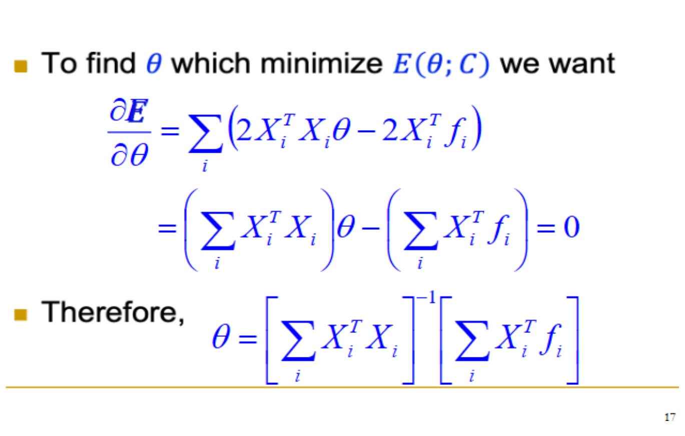
- With the new notation,

$$\begin{aligned} E(\theta;C) &= \sum_i [X_i\theta - f_i]^T[X_i\theta - f_i] \\ &= \sum_i (X_i\theta)^T X_i\theta - f_i^T X_i\theta - (X_i\theta)^T f_i + f_i^T f_i \\ &= \sum_i (\theta^T X_i^T X_i\theta - 2 f_i^T X_i\theta + f_i^T f_i) \\ &= \theta^T \sum_i (X_i^T X_i\theta - 2X_i^T f_i) + \sum_i f_i^T f_i \end{aligned}$$

See lecture set9, for the affine model

$X_i\theta$ is the transformed point. X_i is different for different transformation models, but the rest of the derivation is the same.

16



- To find θ which minimize $E(\theta;C)$ we want

$$\begin{aligned} \frac{\partial E}{\partial \theta} &= \sum_i (2X_i^T X_i\theta - 2X_i^T f_i) \\ &= \left(\sum_i X_i^T X_i\right)\theta - \left(\sum_i X_i^T f_i\right) = 0 \end{aligned}$$

- Therefore, $\theta = \left[\sum_i X_i^T X_i\right]^{-1}\left[\sum_i X_i^T f_i\right]$

17

This is the formulation for similarity transformation:

Similarity: Euclidean + scaling

$$T_x(x, y) = a_{00} + s_x(x\cos\phi + y\sin\phi)$$
$$T_y(x, y) = b_{00} + s_y(-x\sin\phi + y\cos\phi)$$

Which in linear form is :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} s_x \\ s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

In this formulation, a_{00} and b_{00} are our translation components which are points used to shift. To affect sizing, s_y and s_x as scaling factors along both axes (horizontal and vertical or x and y). For rotation, $\emptyset$ is used for the angle or orientation for rotation. Finally, we are using trigonometric values (cosine and sine) in conjunction with everything else to create new transformed points, also known as T_X and T_y. Those 2 are also polynomial expressions which are the result when a pixel is mapped and moved by a vector function.

They can be expanded as:

$$T_x(x, y) = \sum_{r=0}^{m}\sum_{k=0}^{m-r} a_{rk} x^r y^k \qquad T_y(x, y) = \sum_{r=0}^{m}\sum_{k=0}^{m-r} b_{rk} x^r y^k$$

Essential this summation form of T_x and T_y where m is the degree of the polynomial while r and k are exponent values for x and y respectively which affects how many times they are used in a term. These are specific constraints on coefficients to enforce rotation and isotropic scaling.

Overall, the formula can be simplified further into this to accommodate Euclidean error $e_i = X_i\theta - f_i$, we must simplify our equation:

$$x'_i = t_x + ax_i + by_i$$
$$y'_i = t_y - bx_i + ay_i$$

In this case, a and b are not the same as a_{00} or b_{00} but are instead parameters for rotation as they use the trigonometric values (which in this case a= s * cos ∅ and b = s cos ∅ with s being the scaling factor used for isotropic scaling which will affect size).

Using our Euclidean error formula, f_i contains the fixed coordinates from a fixed point within i or the i-th point from a referenced image within a 1x2 matrix. θ contains transformation parameters contained in a 1x4 vector with two translations(t_x, t_y) and the two terms derived from the scaling and rotation. X_i is a matrix which contains constructed from moving image coordinates for the i-th point:

$$e_i = X_i\theta - f_i$$

$$\mathbf{f}_i = \begin{pmatrix} f_{ix} \\ f_{iy} \end{pmatrix} \qquad \theta = \begin{pmatrix} t_x \\ t_y \\ a \\ b \end{pmatrix} \qquad \mathbf{X}_i\theta = \begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{pmatrix} t_x + ax_i + by_i \\ t_y - bx_i + ay_i \end{pmatrix}$$

$$\mathbf{X}_i = \begin{pmatrix} 1 & 0 & x_i & y_i \\ 0 & 1 & y_i & -x_i \end{pmatrix}$$

The final result is e_i, which is the error vector for the i-th correspondence is the difference between the transformed source point ($X_i\,\theta$) and the measured target point (f_i).)

$$\mathbf{e}_i = \mathbf{X}_i\theta - \mathbf{f}_i = \begin{pmatrix} 1 & 0 & x_i & y_i \\ 0 & 1 & y_i & -x_i \end{pmatrix} \begin{pmatrix} t_x \\ t_y \\ a \\ b \end{pmatrix} - \begin{pmatrix} f_{ix} \\ f_{iy} \end{pmatrix}$$

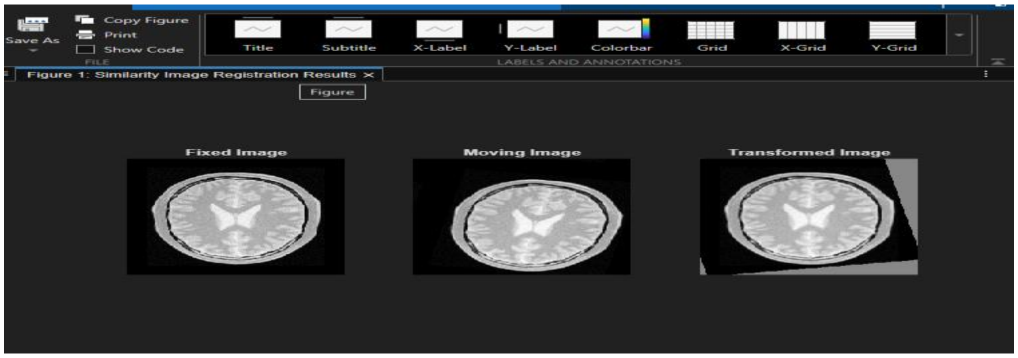
Our new changed estimation model would be:

$$E(\theta) = \sum_i \mathbf{e}_i^T \mathbf{e}_i = \sum_i [\mathbf{X}_i\theta - \mathbf{f}_i]^T [\mathbf{X}_i\theta - \mathbf{f}_i]$$

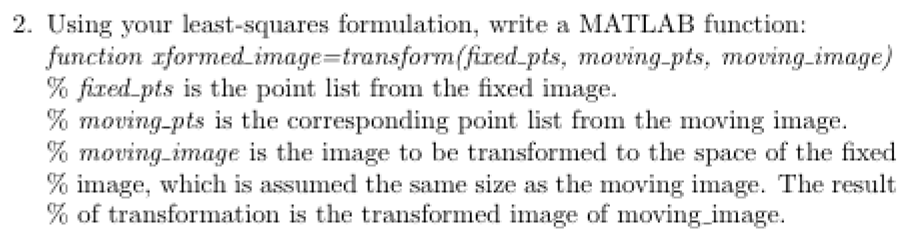
$$\theta = \left[\sum_i \mathbf{X}_i^T \mathbf{X}_i\right]^{-1} \left[\sum_i \mathbf{X}_i^T \mathbf{f}_i\right]$$

Which gives us the optimal translation, rotational and scaling parameters for aligning the moving image to the fixed point.

Result:



As seen above, the moving image was manipulated, so it matched the orientation of the fixed image. Our similarity transformation is a success in that regard as using least squares estimation and the error equations allowed us to transform each coordinate point to create a matching result.


2. Using your least-squares formulation, write a MATLAB function:
function xformed_image=transform(fixed_pts, moving_pts, moving_image)
% *fixed_pts* is the point list from the fixed image.
% *moving_pts* is the corresponding point list from the moving image.
% *moving_image* is the image to be transformed to the space of the fixed
% image, which is assumed the same size as the moving image. The result
% of transformation is the transformed image of moving_image.

Hypothesis:

We believe that the the transformed image will appear as the source image uniformly scaled and rotated to correct its size and angle, and translated to match the position of the target image. Since our testing image is that , because of to isotropic scaling, the shape and aspect ratio of the anatomical features will be perfectly preserved while achieving alignment.

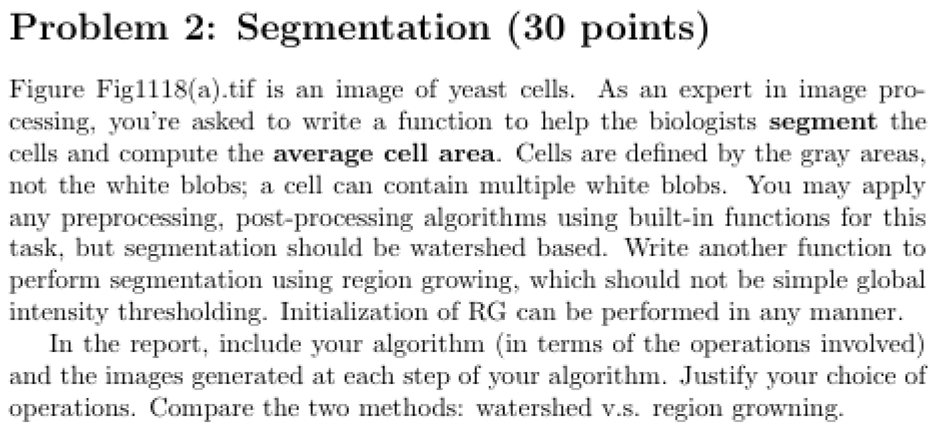

Problem 2: Segmentation (30 points)

Figure Fig1118(a).tif is an image of yeast cells. As an expert in image processing, you're asked to write a function to help the biologists **segment** the cells and compute the **average cell area**. Cells are defined by the gray areas, not the white blobs; a cell can contain multiple white blobs. You may apply any preprocessing, post-processing algorithms using built-in functions for this task, but segmentation should be watershed based. Write another function to perform segmentation using region growing, which should not be simple global intensity thresholding. Initialization of RG can be performed in any manner.

In the report, include your algorithm (in terms of the operations involved) and the images generated at each step of your algorithm. Justify your choice of operations. Compare the two methods: watershed v.s. region growning.

In this problem, we are working on segmenting yeast cells from the image Fig1118(a).tif. The goal is to clearly separate the gray cells from the background and remove any small noise or white spots inside the cells. Using:

Algorithm 1: Marker-Controlled Watershed (boundary-driven; great at splitting touching cells)

Algorithm 2: Region Growing (region-driven; great at smooth, filled interiors)

For each method, we evaluate the same image and report num of regions (N) and average area (px). We also show what each preprocessing step did for that algorithm since the best prep isn't identical for both.

A) Watershed Segmentation – Algorithm 1:

1. Preprocessing:

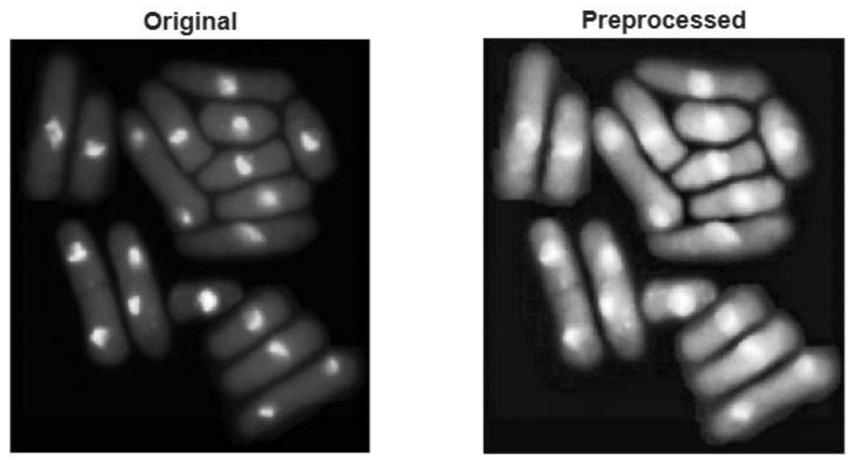
We preprocessed the image for segmentation, applying three filters from the lecture materials to improve the clarity of yeast cells and reduce noise before running the Watershed algorithm.

First, we used adapthisteq to enhance local contrast. This brightened the dark areas and made the thin borders between cells and the background much more visible. In the context of Watershed, this step is important because the algorithm relies on the difference between object edges and the background when calculating the distance map. Clearer edges result in more accurate “hills” and “valleys” in the distance transform.

As a follow-up step to thresholding, we used imgaussfilt to blur small intensity fluctuations due to inhomogeneous lighting or random noise. This slightly softened the appearance of the image and did not destroy the shape of the cells. For Watershed, this smoothing step prevents the distance transform from producing many small false peaks that could cause over-segmentation.

Then we applied the median filter, medfilt2, to remove small bright or dark dots - the so-called salt-and-pepper noise. It filters out outliers but maintains sharp edges. In this way, cell interiors became homogeneous, which allows the Watershed algorithm to produce smooth distance surfaces without superfluous boundaries.

Result of the step:



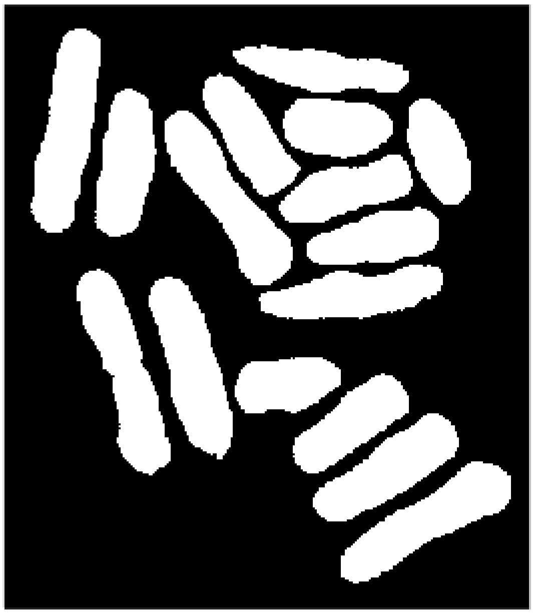
The yeast cells look smoother, the interiors are less noisy, and the background is more even. The bright spots are still visible but less harsh, and the edges of the cells are more distinct. This makes it much easier for the next segmentation steps to correctly detect and separate each cell. In a medical application, this allows us to view the cell in a translucent view.

<u>2.) Binary Mask Creation:</u>

First, we applied graythresholding using Otsu's method to convert the image into a binary mask, then morphologically cleaned it. We filled in holes to make each cell solid (imfill), removed small debris (bwareaopen), and took care of small gaps between cell edges using a small closing operation (imclose). These morphological steps ensured continuity of each yeast cell and good separation from the background.

This preprocessing step resulted in a clean binary image consisting of smooth, connected yeast cells, ready for the next step: Watershed segmentation.

Cleaned Mask



$$g(x,y)=\begin{cases}1 & \text{if } f(x,y)>T\\ 0 & \text{if } f(x,y)\le T\end{cases}$$

$$T=T\left[x,y,\overbrace{p(x,y)}^{\text{Some local property}},\underbrace{f(x,y)}_{\text{Grayscale value}}\right]$$

This method converts our image into black and white, where the cells appear white (foreground), and the background appears black. Which makes the weird shapes around the shapes disappear (as part of the background not the objects themselves) here in the figure (under this), making each separated.

Figure methods

3) Segmentation:

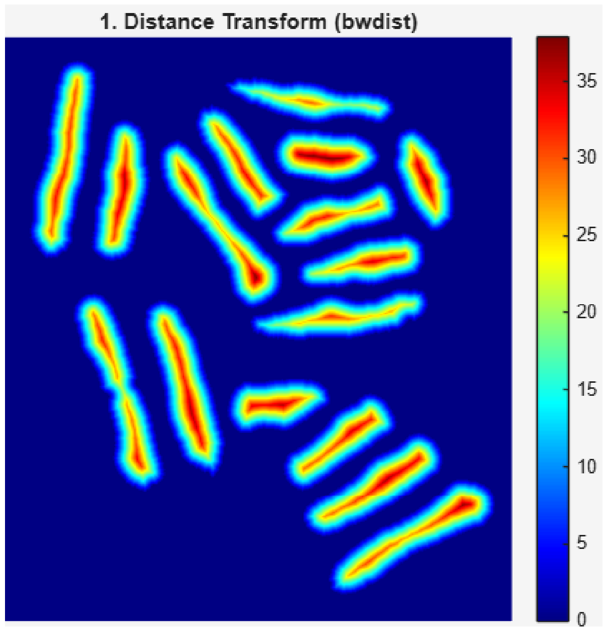
After preprocessing, the Watershed segmentation was performed step by step to separate each yeast cell in detail.

Step 1) Distance Transform:

To find the distance of each pixel within the yeast-cell regions from the nearest background pixel, the distance transform was computed using the function bwdist(~BW). This process turns the binary mask into a topographic surface in which cell centers become high peaks (warm colors), and the boundaries become valleys (cool colors). This map provides the groundwork for the watershed algorithm, representing the geometric depth of each region.

In order to reduce noise and avoid false local maxima, imhmin(D, 0.5) was optionally applied to suppress shallow fluctuations and retain only true peaks corresponding to real cell centers

Result:

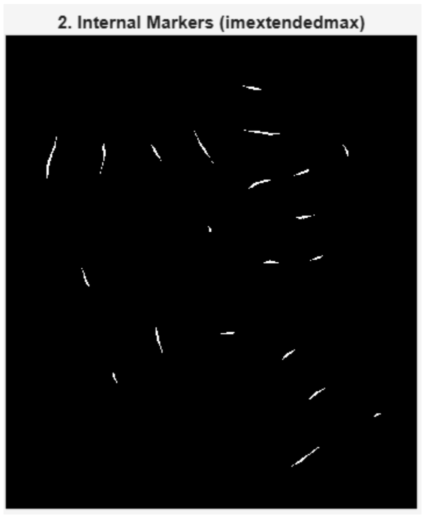


The figure shows the distance transform map. Each yeast cell exhibits a bright red or yellow center flanked by cooler blue boundaries. This indicates that the transform correctly emphasizes the central "hills" used for marker detection in later steps.

<u>Step 2) Inverting the Distance Map</u>

Internal markers were generated from the distance map using imextendedmax(D, 2).This function identifies the most prominent regional maxima, ensuring that each cell interior receives at least one marker while avoiding spurious peaks.These markers act as "seeds" that signal the watershed algorithm where each individual object should start forming.

Result:



As shown, each white dot represents a detected internal marker positioned roughly at the center of a yeast cell. This confirms that the algorithm successfully located valid starting points for the segmentation process without excessive noise.

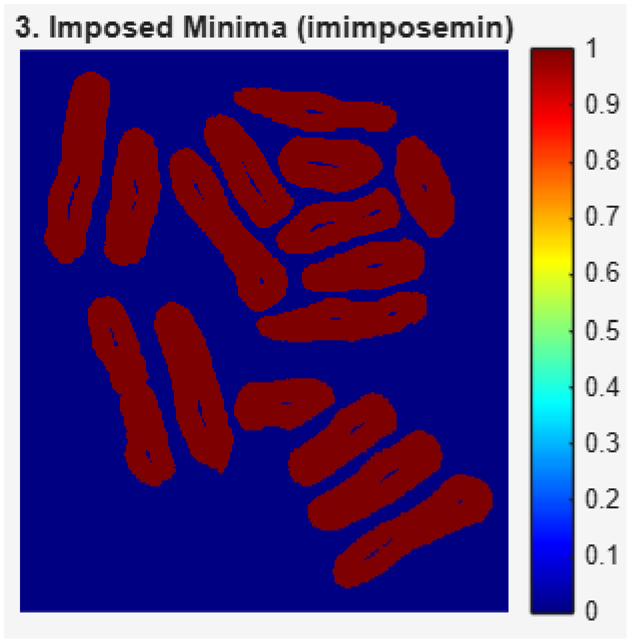
Step 3) Imposing Minima:

To avoid flooding the watershed across incorrect boundaries, minima were imposed on the inverted distance map, D_neg = −D.

The operation imimposemin(D_neg, markers | ~BW) forces the algorithm to treat both the internal markers and the background as fixed minima.

This ensures that the watershed flooding starts in each cell from valid seeds only and will stop naturally at the boundaries between them.

Result:

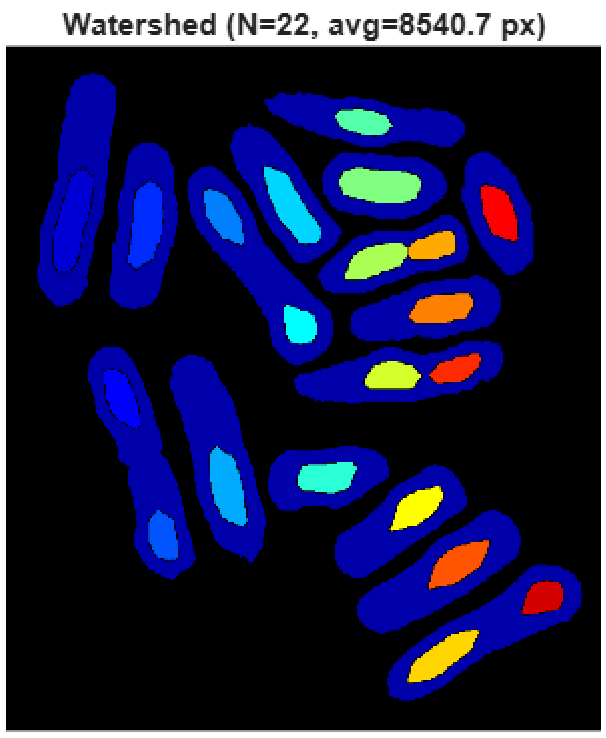


The imposed-minima map shows red signifies catchment basins controlled by the interior of each yeast cell, while blue forms the background, suppressing areas where segmentation will not spread. This effectively restricts the next watershed lines to actual cell boundaries.

Step 4) Watershed Segmentation and Final Result:

After setting the controlled minima, we applied the watershed transform using L = watershed(Wimpose). Each of the resulting basins corresponds to a single yeast cell. The background value was set to 0, L(~BW)=0. To display the regions, label2rgb was used to assign a unique color to each label.

Result:



The final segmented image below illustrates 22 distinct yeast-cell regions with an average cell area of ≈8540.7 pixels. Most touching cells were separated successfully; internal white vacuoles were correctly treated as part of each.

Conclusion for Watershed:

The marker-controlled watershed method achieved accurate separation of most yeast cells, particularly those that were touching but clearly outlined. By enhancing contrast, removing noise, and defining strong internal markers, the algorithm produced precise boundary-based segmentation. Its ability to follow intensity gradients made it ideal for distinguishing closely packed cells, although regions with weak edges occasionally remained merged. Overall, this approach generated clear, reliable, and measurable yeast-cell regions for further analysis.

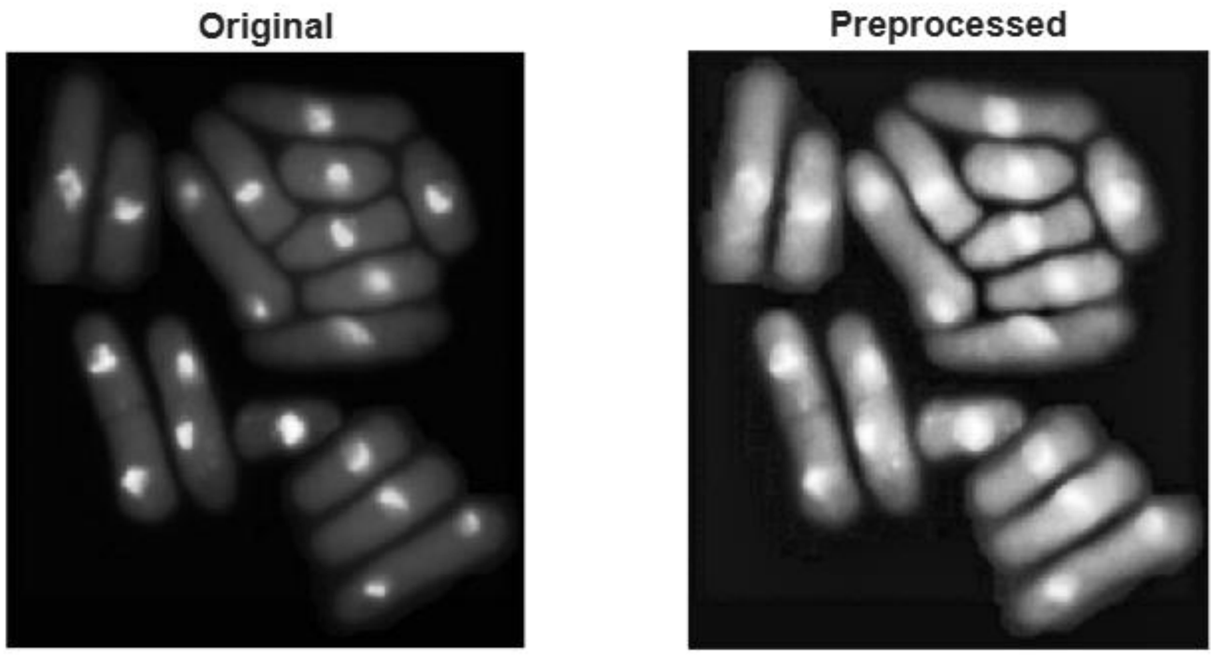
B) Seeded Region Growing - Algorithm 2:

1) **Preprocessing**:

For the Region Growing method, we used the same preprocessing steps described in the Watershed section.

We first enhanced local contrast using adapthisteq to make the edges of each yeast cell more distinct and better separated from the background.
Then, imgaussfilt was used to smooth small intensity variations, reducing background noise without distorting the general shape of the cells.
Finally, medfilt2 removed small bright or dark spots, ensuring each cell to appear uniform and solid.

After that, thresholding and morphological operations-imfill, bwareaopen, and imclose-resulted in a clean binary mask with bright cells and a completely dark background. This preprocessing created a clear, noise-free image, perfect for placing seeds and allowing the Region Growing algorithm to expand accurately within each yeast cell.

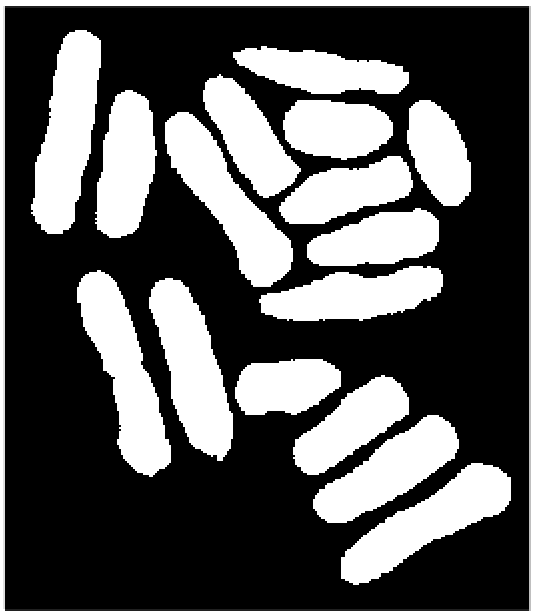


2) **<u>Binary Mask Creation:</u>**

Same , we applied graythresholding using Otsu's method to convert the image into a binary mask, then morphologically cleaned it. We filled in holes to make each cell solid (imfill), removed small debris (bwareaopen), and took care of small gaps between cell edges using a small closing operation (imclose). These morphological steps ensured continuity of each yeast cell and good separation from the background.

This preprocessing step resulted in a clean binary image consisting of smooth, connected yeast cells, ready for the next step: Watershed segmentation.

Cleaned Mask

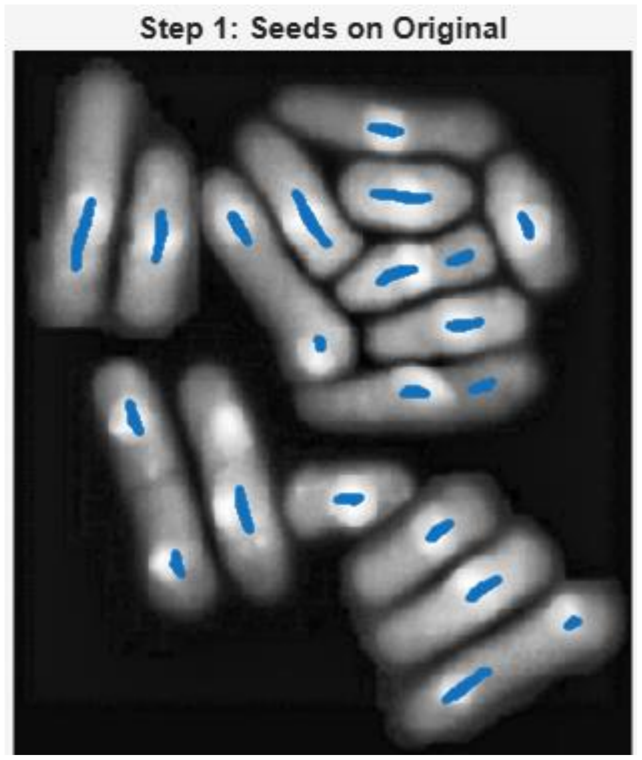


3) <u>Segmentation</u>:
<u>Step 1): Selection of Seed Points:</u>

Region growing was initiated with the identification of seed points, representing the starting center for every yeast cell.
We used the distance transform peaks detected by imextendedmax(D, 2) to locate these initial positions.
Each seed roughly corresponds to a single cell centre, providing a reference intensity for controlled growth.

First, the seeds were labeled by bwlabel(peaks, 8) such that each of the regions could grow independently.

Result:



The figure of the selected seeds, shown as blue dots superimposed on the original grayscale image. Each seed lies approximately at the center of a yeast cell, confirming that the initialization stage correctly detected one representative point per cell while avoiding false detections in the background.

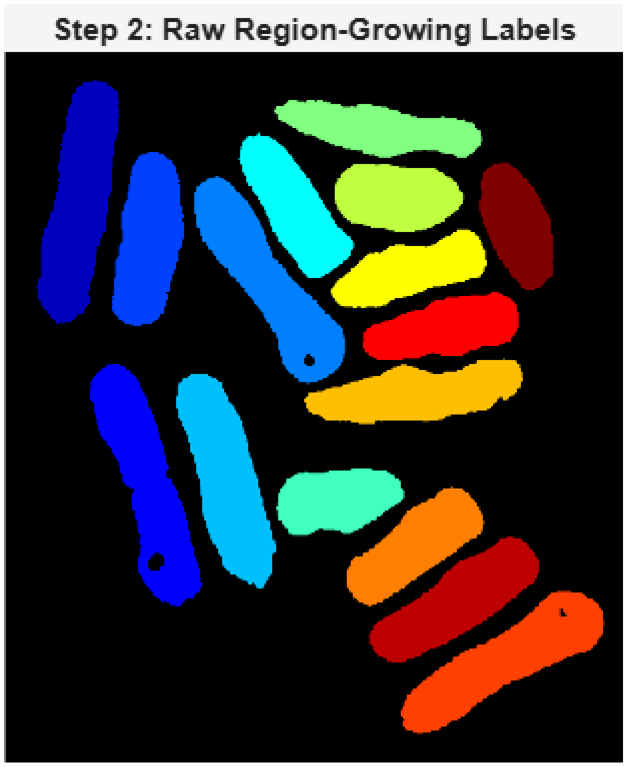
Step 2) Region Growing Process:

Starting from each seed, the algorithm expanded the area by adding adjacent pixels whose intensity values were within a specified tolerance (tol = 30 in this case) of the mean intensity value of the area.

When a new pixel was incorporated, the mean was updated to accommodate the gradual intensity changes within a cell.

This was done until there were no pixels left that satisfied the conditional similarity or the maximum iteration limit (maxIter) was reached.

Result:

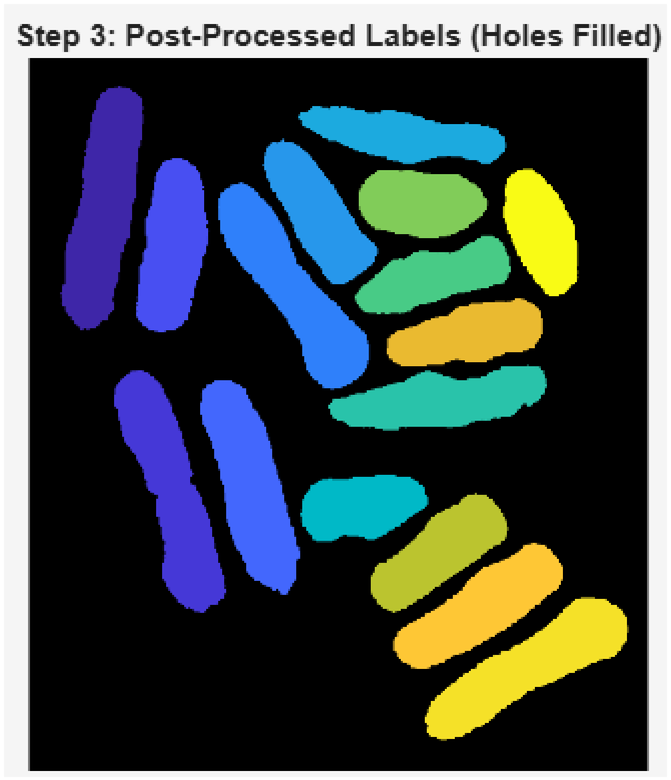


The figure shows the raw region-growing labels where each cell interior is displayed in a different color. Most of the yeast cells were correctly separated into individual regions; some touching cells remained connected where their gray levels were very similar. On the whole, this step yielded smooth, filled interiors that correspond well with the visible cells. But we see some holes in each we will fill it .

Step 3: Post-Processing (Hole Filling):

To make sure every region was solid on the inside, we applied imfill binary hole filling after the first segmentation. This took care of tiny gaps or absent pixels that might have been created during the growth phase. To help with accurate quantification, the image was bwlabel relabeled so that every continuous area received an exclusive label.

Result:



The post-processed label map below has well-defined, smooth yeast cell regions without internal holes or background noise.

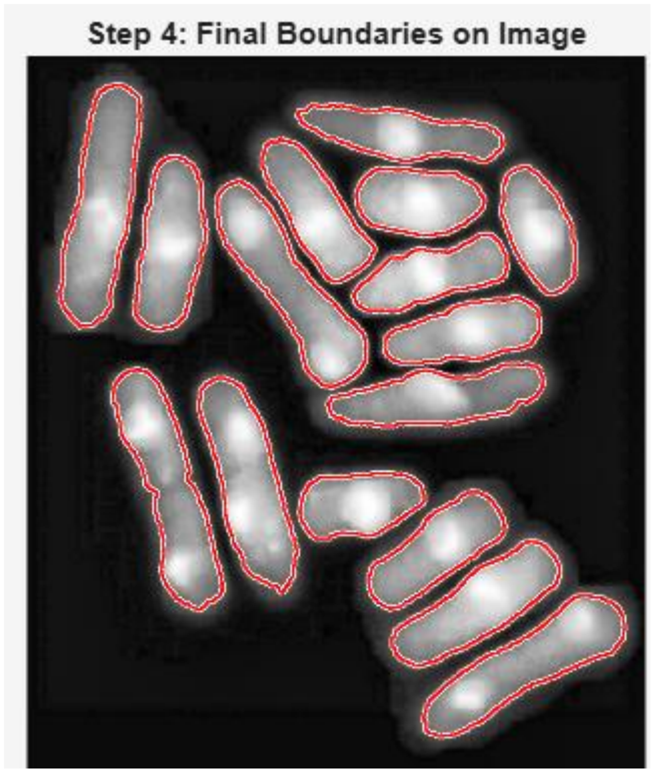
All internal gaps were filled appropriately, which affirmed that the post-processing step improved the uniformity of the segmentation in general.

Step 4) final Boundary Overlay and Measurements:

To check for its accuracy, we overlaid the final region boundaries, bwperim, over the original grayscale image.

This helps compare the detected boundaries with the actual visible edges of the cells. We also employed region props("table", L, "Area") to calculate the area of the segmented cells to count the number of regions (N), and compute the average area segmented cells.

Result:

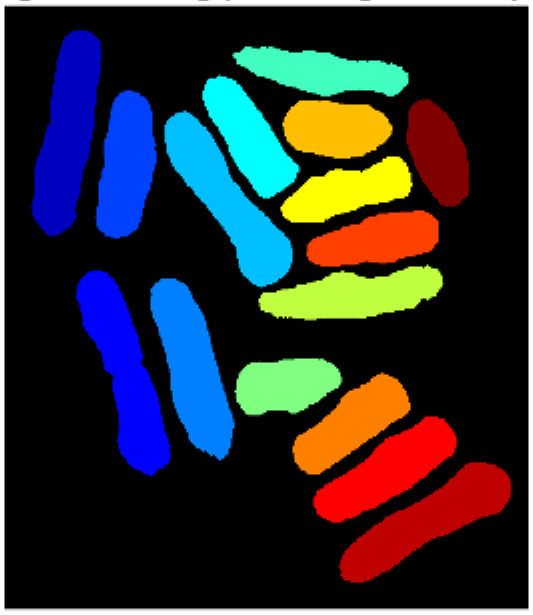


The figure represents the final boundary overlay, where red outlines closely match the edges of the visible yeast cells. This confirms that the segmentation was able to capture the shape and size of most cells well, with very minor merging in areas of similar intensity.

<u>Step 5) Final Segmentation Result:</u>

The final segmented result is visualized below, showing individual yeast cells in unique colors. In total, 16 regions were detected with an average cell area of about 11,921 pixels. Compared to the watershed method, region growing produced fewer segments, since cells of similar intensity were merged, but each region was smoother and better filled.

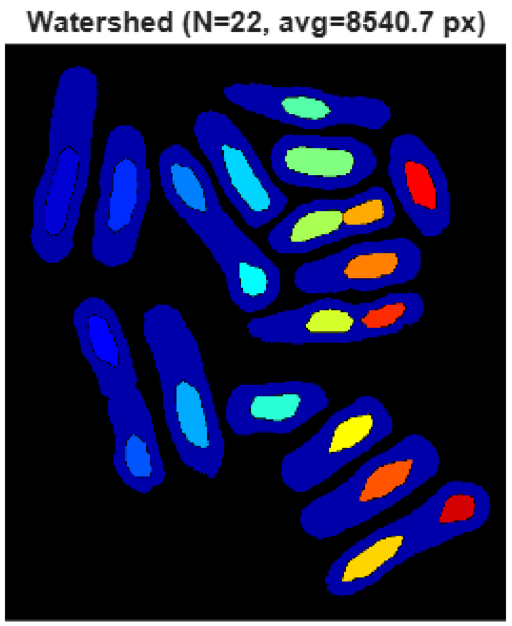
Region Growing (N=16, avg=11921.0 px)

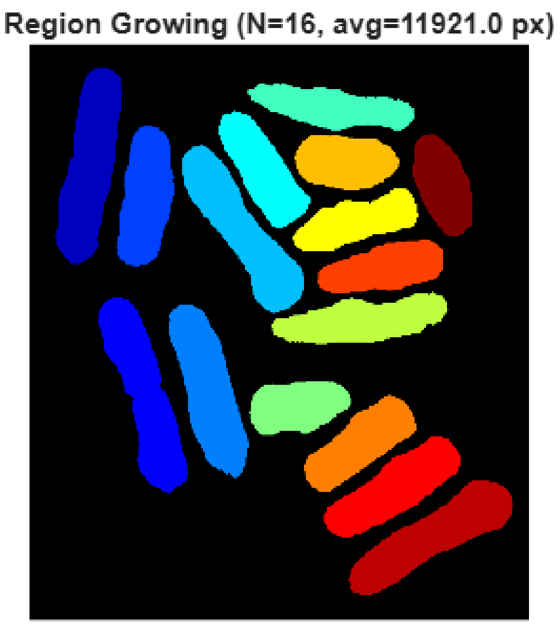


Conclusion for Region Growing:

This resulted in the successful segmentation of the yeast cells through the region-growing algorithm that grows from seed points based on intensity similarities. This region-driven approach produced smoother and more homogeneous regions compared to the boundary-driven watershed method. However, since it is mainly dependent on pixel intensity rather than edge gradients, some of the closely touching cells remained fused when their brightness values were nearly identical.

Final Result & comparison:





The watershed method generated 22 regions, each averaging around 8540 pixels. While it was effective at delineating touching cells, it also tended to fracture one cell into multiple small pieces. We believe this is because the bright regions within each cell were misidentified as separate. This method is particularly sensitive to small variations in brightness and often over-segment images, especially when there are bright regions within the cells.

Using the region growing method, 16 regions were generated, each averaging 11,921 pixels, and it performed best in keeping the cells intact. Although there were some near the merged cells, the rest were separated in a way that closely resembled the actual cell shape. The newly formed cell outlines were smoother and more precise when compared to the watershed method.

The region growing method aligned more closely with the actual definition of the yeast cell and yielded more practical results, especially when one considers the measurement of average cell size. The watershed method is useful at finding boundaries, but less applicable since it overly segments single cells into too many parts.

Conclusion:

In conclusion, this assignment showcases how digital image processing techniques, specifically similarity transformations and thresholding can be applicable to real world applications. We can totally see how someone in the medical field can image pictures of human organs to check for tumors or other diseases using modifcations/transformations of an image's features to help extract and analyze certain details.

Resources that helped me :

Camera geometry fundamentals - ScienceDirect

Similarity Transformation - an overview | ScienceDirect Topics

Geometric Transformations — nVision User Guide 2016.3 documentation

What Is Image Segmentation?

Image segmentation - Explained!

adapthisteq - Contrast-limited adaptive histogram + Lecture Notes.

graythresh - Global image threshold using Otsu's method + computer vision 381 thresholding techniques

Watershed transform - MATLAB

Example of Segmentation using Watershed Algorithm in Matlab