BIOM 5202 / SYSC 5202 - Fall 2024: Assignment #3: Textures and Shapes
Name: Amirhossein Ebrahimi (300396640)

Source code Link: (I only put the main part of the code, you can access and run everything from scratch in Google Colab). In each section, I have hyperlinked each question title to its relevant part of the code in Colab

Instructions

- This assignment must be completed individually. It is okay to discuss things with your colleagues, but do not copy any of their answers, codes or notes. Plagiarism is a serious instructional offence that will not be tolerated. Please cite all sources when referring to external resources (e.g. course notes, books, papers, etc).
- For this assignment, it is recommended that you work with Matlab (or equivalent software). Carleton students can install Matlab on their personally owned computers for learning purposes (see http://carleton.ca/ccs/matlab/). Python is also acceptable provided the students submit codes with clear instructions on how to run their codes. All codes must be appropriately commented. Do not simply copy codes (or parts of it) from the internet! All sources must be cited.
- Questions require 1–2 sentences to answer. Long answers are not required.
- The deliverable for this assignment is a short report that must include: i) the answers to each question; ii) the images and plots generated in each step, carefully captioned; iii) discussion of all information requested in each section; iv) the Matlab codes, properly commented (.m file) and the image(s) you used to generate the results presented (yes! I will run your script and check the results).
- This assignment requires you to use ChatGPT (or a similar LLM) to generate a code for you. Think carefully about which prompt question you will use for this.
- Images used in this assignment

effusion.jpg	pneumonia.jpg	inflitration.jpg	Acanthocytes.jpeg
Echinocytes.jpeg			

- Due date and time: Nov 27th, 2024 at 23:59.

SECTION 1 - Textures

For this part, please use the images effusion.jpg, pneumonia.jpg, and inflitration.jpg

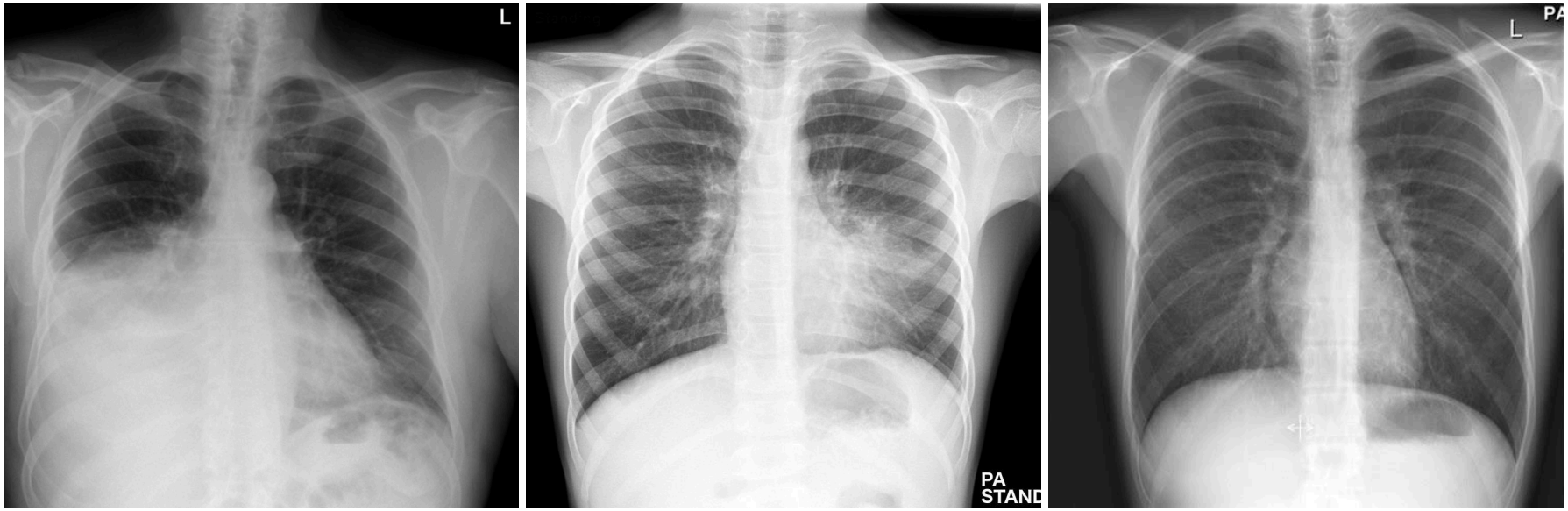
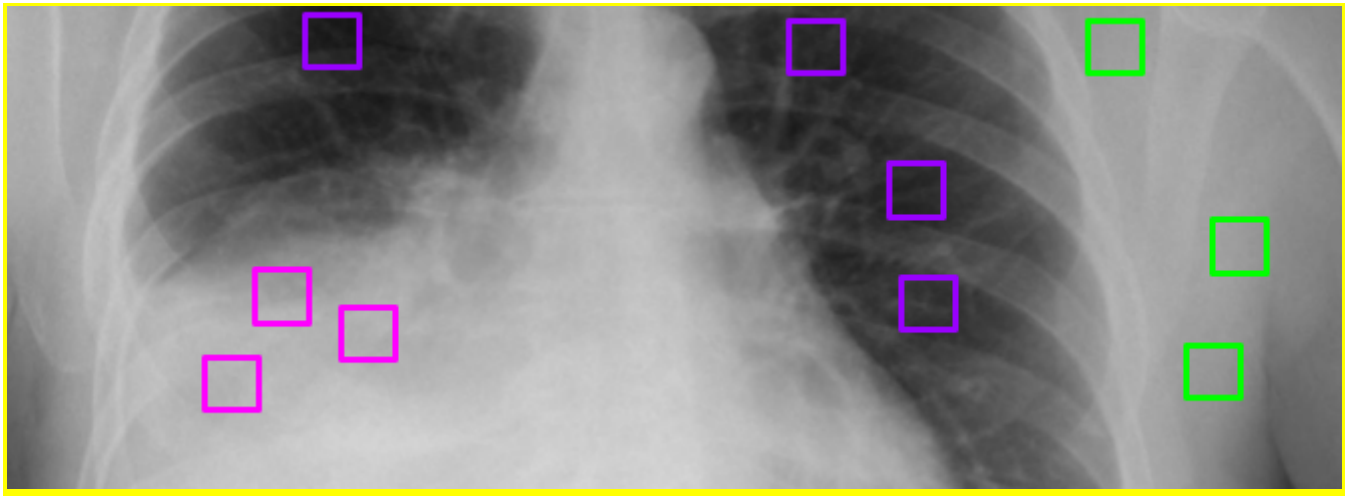
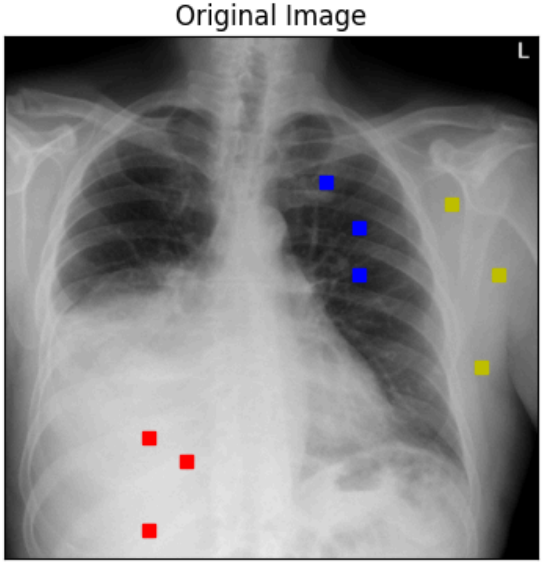
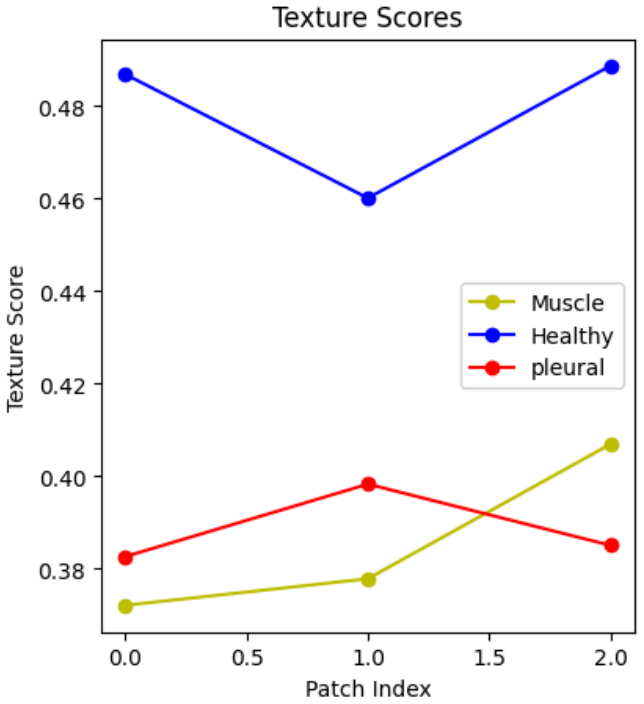


Figure 1: Each one of these images are chest X-rays, displaying an example of (left) a case of pleural effusion (an abnormal collection of fluid between the two layers of membrane covering the lungs); (Middle) a case of pneumonia (an infection that inflames the air sacs in one or both lungs); and (Right) a case of pulmonary infiltrate (when a substance denser than air, such as pus, blood, or protein, which lingers within the parenchyma of the lungs). The approximate regions where each medical condition can be identified is shown in Figure 1.

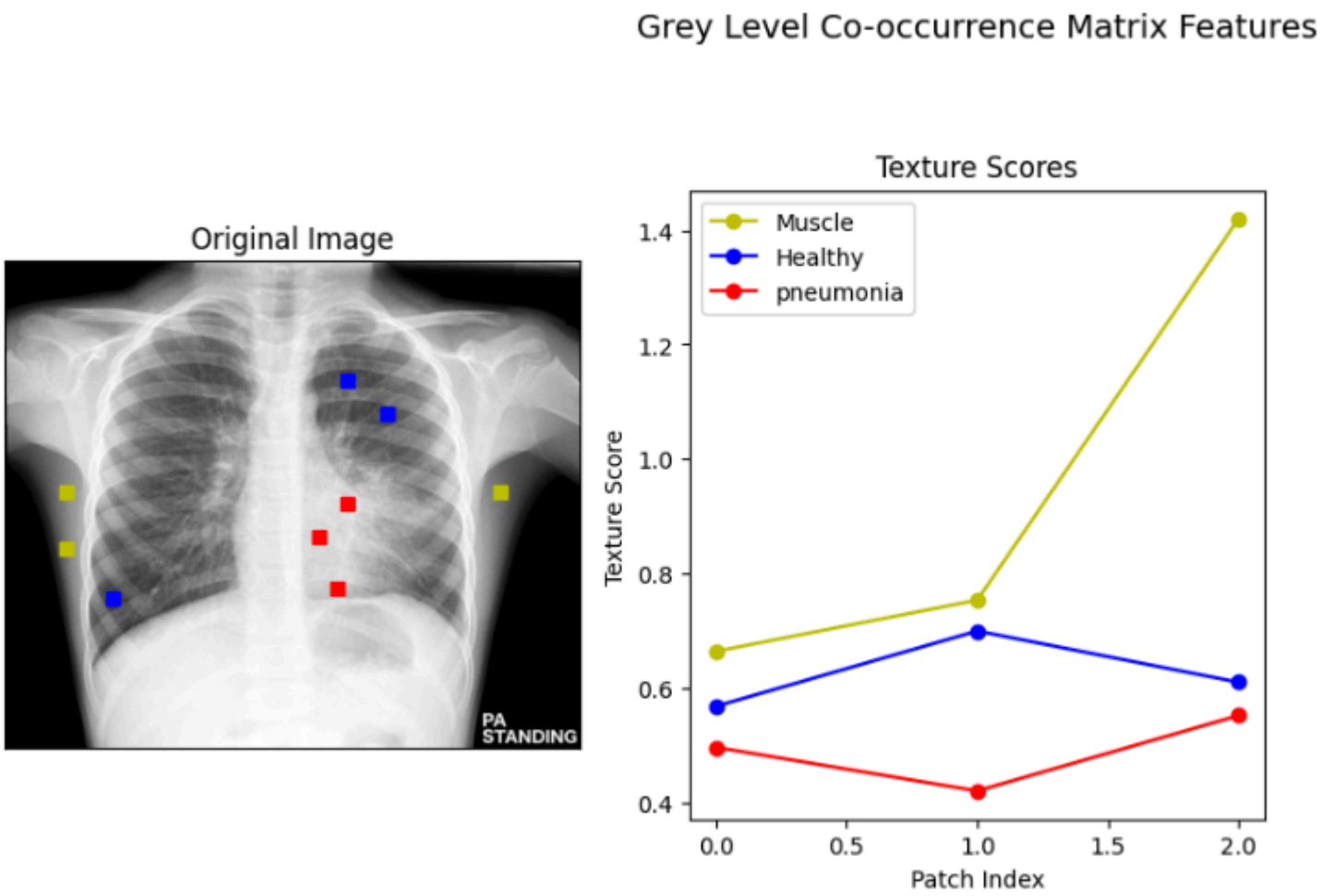
Update: for each image, select four 20×20 samples of three texture regions. For example, for the image below shows samples of (purple) healthy lung, (pink) fluid-filled (edematous) lung, and (green) muscle.



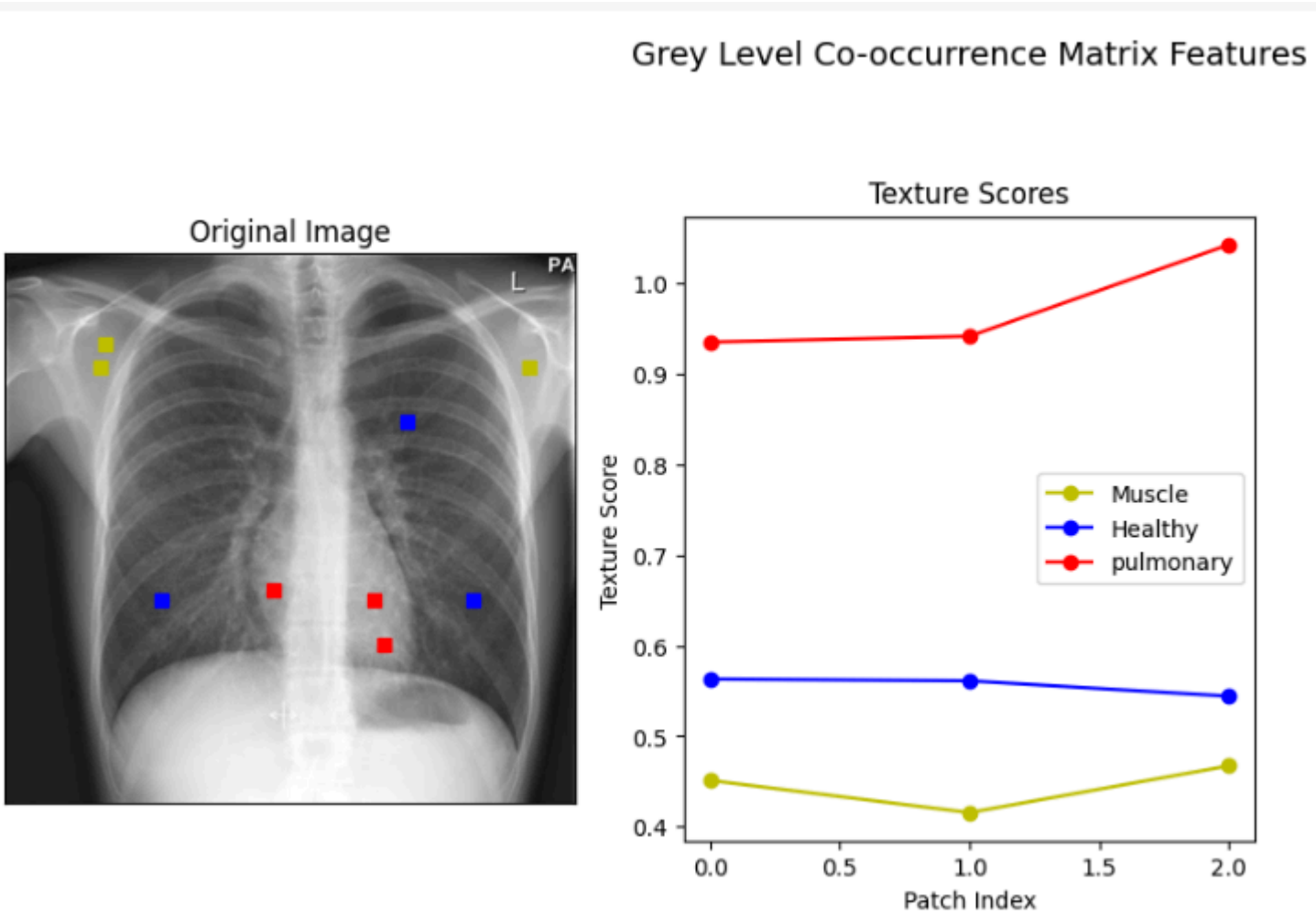
Question:	Answer / Code / Image
1A: Using a set of minimum 5 textural descriptors of your choice (e.g. GLCM-entropy counts as one, GLCM-IDM is another one, GLRLM-SRE is another one, etc), design a score able to objectively distinguish the cases. You can choose for example to average all five textures, or apply weights to them. Be creative	``` def compute_features(patch): glcm = graycomatrix(patch, distances=[5], angles=[0], levels=256, symmetric=True, normed=True) dissimilarity = graycoprops(glcm, 'dissimilarity')[0, 0] correlation = graycoprops(glcm, 'correlation')[0, 0] homogeneity = graycoprops(glcm, 'homogeneity')[0, 0] energy = graycoprops(glcm, 'energy')[0, 0] # Compute GLRLM Short Run Emphasis (SRE) sre = np.sum(glcm[:, :, 0, 0] ** 2) / np.sum(glcm[:, :, 0, 0]) return dissimilarity, correlation, homogeneity, energy, sre ``` Weights assigned for each feature: 'dissimilarity': 0.1, 'correlation': 0.1, 'homogeneity': 0.3, 'energy': 0.4, 'short_run_emphasis': 0.1

1B: Apply your texture descriptor to effusion.jpg. Identify its performance in different textural regions	Grey Level Co-occurrence Matrix Features  

1C: Apply your texture descriptor to pneumonia.jpg. Identify its performance in different textural regions.



1D: Apply your texture descriptor to inflitration.jpg. Identify its performance in different textural regions.



1E: Were you successful distinguishing the cases? Explain the reason why you used such textural descriptors, the advantages and their individual limitations.

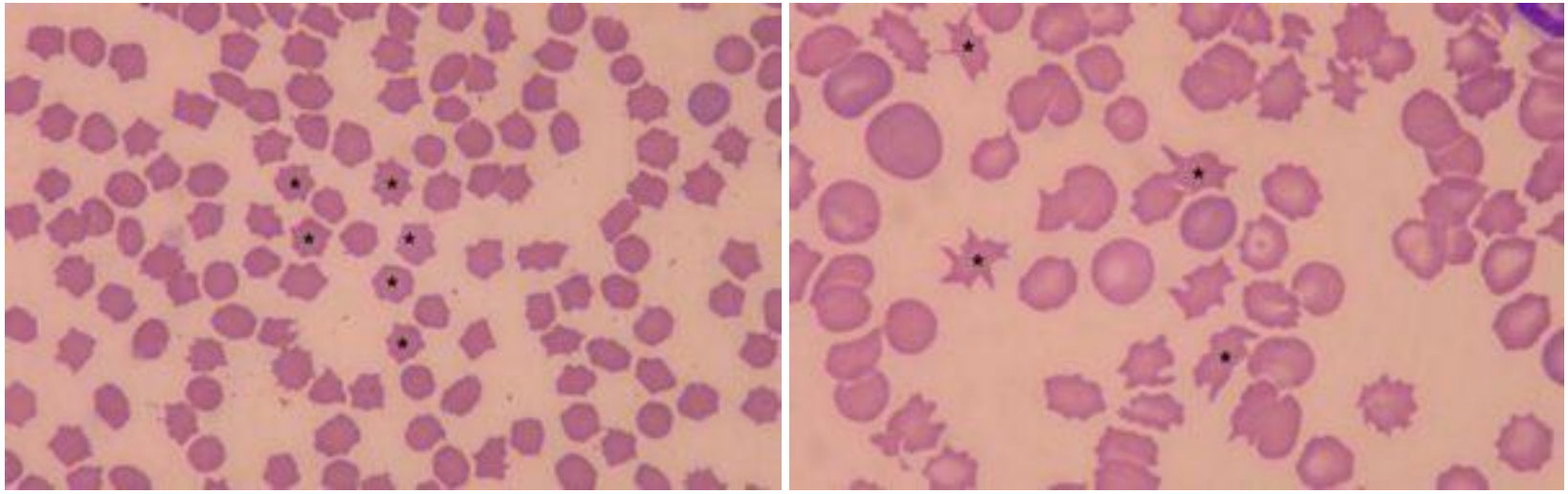
In most cases yes **1D and 1C.** In **1B,** distinguishing between muscle and damaged area in one case was not possible. But I have set the best possible set of weights for the features. Maybe If I chose another muscle area It could be distinguishable.

```
weights = {
    'dissimilarity': 0.1,
    'correlation': 0.1,
    'homogeneity': 0.3,
    'energy': 0.4,
    'short_run_emphasis': 0.1
```

	}

SECTION 2 - Textures

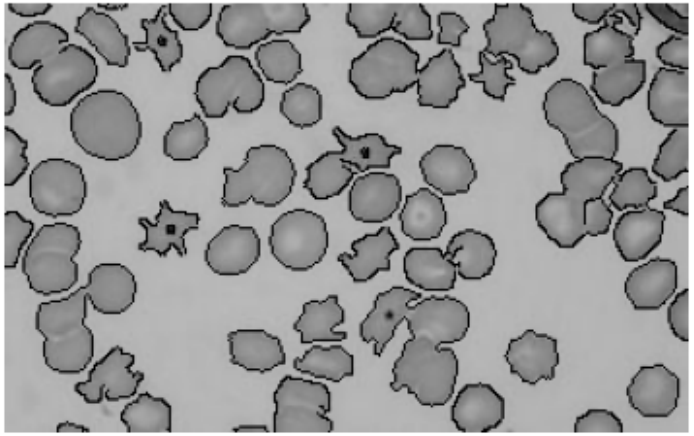
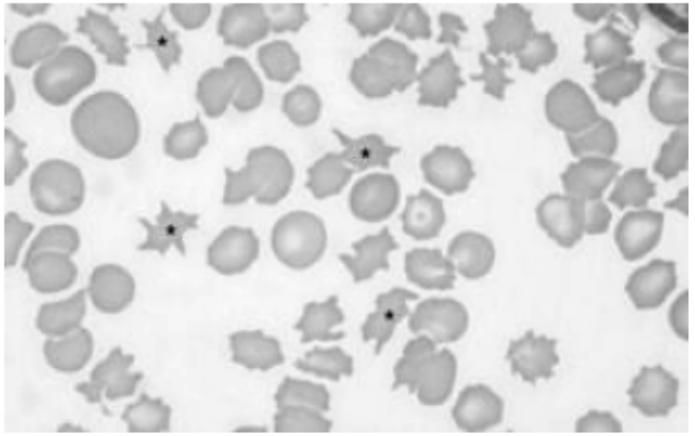
The following text is quoted from: https://www.heskavet.ca/article/top-5-red-blood-cell-pathologies/ for veterinary hematology.

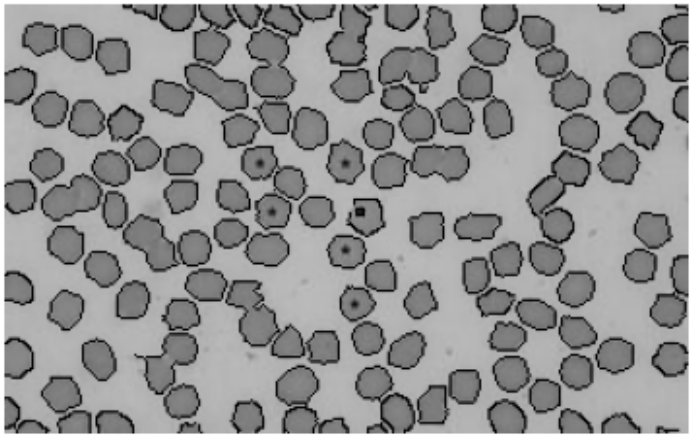
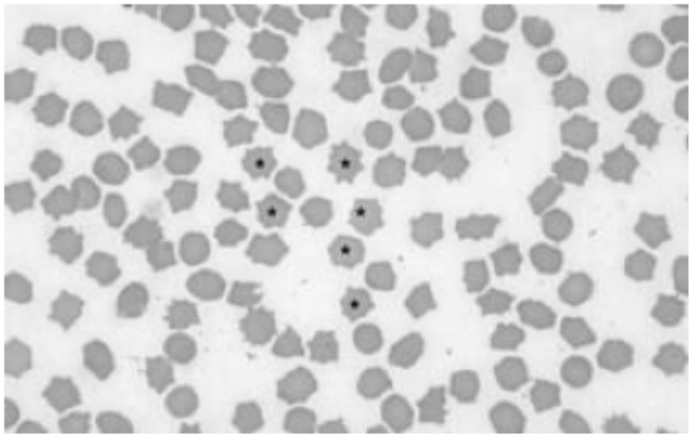
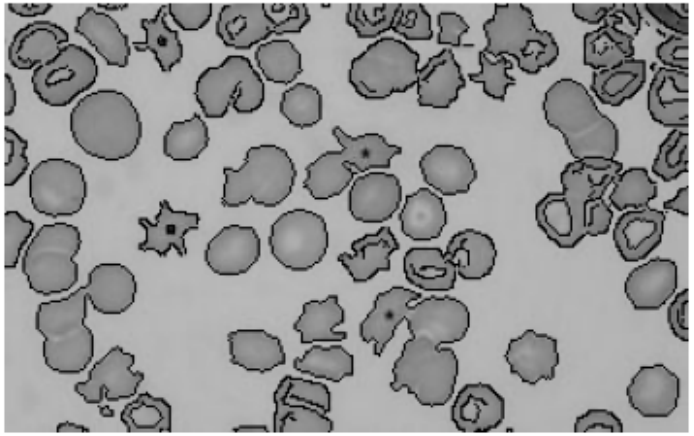
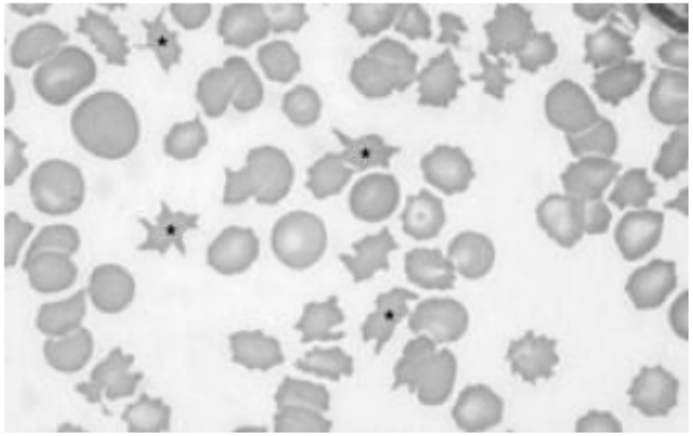


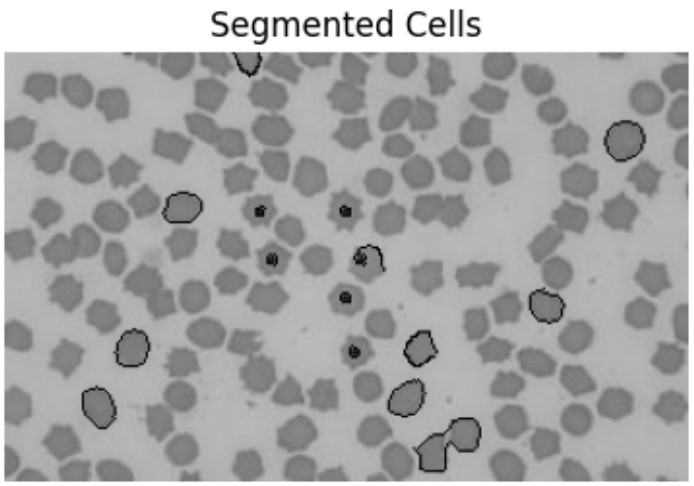
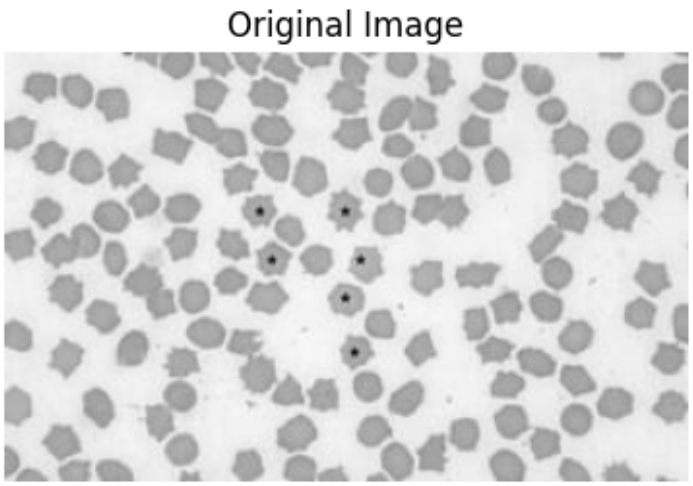
(Left) Echinocytes, or crenated erythrocytes, are spiculated erythrocytes. The surface projections are numerous, evenly distributed and of similar size. It is mostly considered an artifact secondary to slow drying of the blood smear or as a result of prolonged sample storage before slide preparation. This has also been related to snake envenomation, underlying neoplasia (such as lymphoma and mast cell tumor) and glomerulonephritis. This is by far the most common RBC pathology observed on microscopic examination. Most of the time, this is a non-significant finding. (Right) Acanthocytes, or spur cells, are erythrocytes covered by irregularly shaped unevenly distributed surface projections. This erythrocyte anomaly results from alteration in cholesterol or phospholipid concentration in the red blood cell membrane. This can be associated with liver disease in companion animals, and is a common finding in cats suffering from hepatic lipidosis. This has also been related to erythrocytes fragmentation secondary to disseminated intravascular coagulation, hemangiosarcoma, and glomerulonephritis.

You may (but are not required to) use an LLM to help for this question. If you do, list the prompt used, the software provided, and the modifications you made to improve the result.

Question:	**Answer / Code / Image**

2A: Design software to separate cells in each image. The output image should show the boundary for each cell. Use techniques we have seen in the course. Software with >90% accuracy is acceptable.	``` blurred = cv2.GaussianBlur(imag, (3,3), 0) edges = cv2.Canny(blurred, 50, 150) contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) output_image = imag.copy() cv2.drawContours(output_image, contours, -1, (0, 255, 0), 1) ```
2B: Show the images and the identified cell regions. What accuracy was achieved?	Showing all contour: Accuracy ~100% Segmented Cells  Original Image 

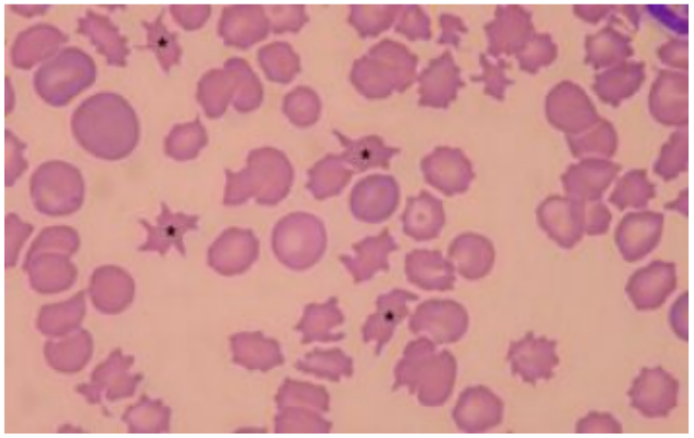
	Segmented Cells  Original Image 
2C: Did the software have any difficulties in the analysis and identification? What caused the issues?	Yes. If I just drew the contours without proper preprocessing, (edge detection parameters , noise reduction) it couldn't find the cells. See below: canny edge detector with lower upper threshold `20, 50` Segmented Cells  Original Image 

	Gaussian Kernel `(7,7)`   Without proper Noise reduction and edge detection, finding boundaries were difficult.
2D: Design software to classify the shapes and differentiate the normal and abnormal cells in each image. Aim for >90% accuracy	```# Circularity def calculate_circularity(contour): perimeter = cv2.arcLength(contour, True) area = cv2.contourArea(contour) if perimeter == 0: return 0 return 4 * np.pi * (area / (perimeter * perimeter)) # Compactness def calculate_compactness(contour): area = cv2.contourArea(contour) perimeter = cv2.arcLength(contour, True) if perimeter == 0: return 0 return (perimeter * perimeter) / area # Eccentricity```

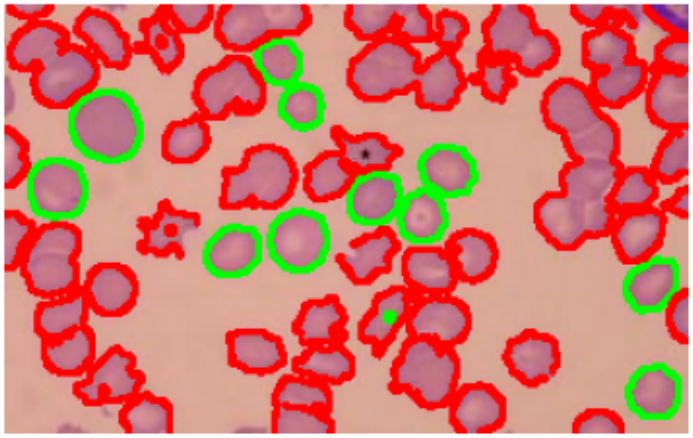
```
def calculate_eccentricity(contour):
    try:
        (x, y), (MA, ma), angle = cv2.fitEllipse(contour)
        eccentricity = np.sqrt(1 - (MA / ma) ** 2) if ma != 0 else 0
        return eccentricity
    except:
        return 0
```

2E: Show the images and the classified cell. What accuracy was achieved?

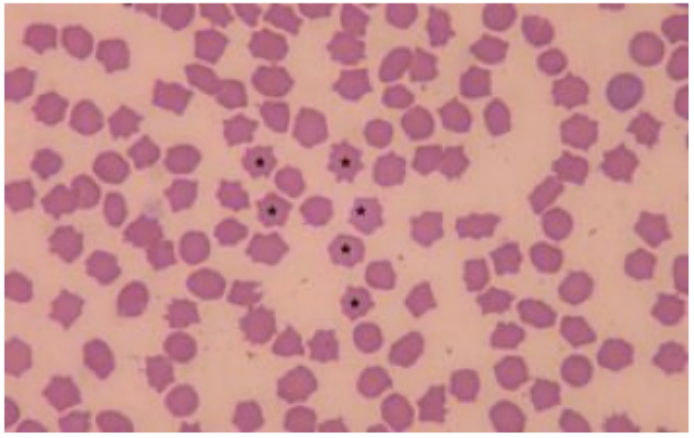
Original Image



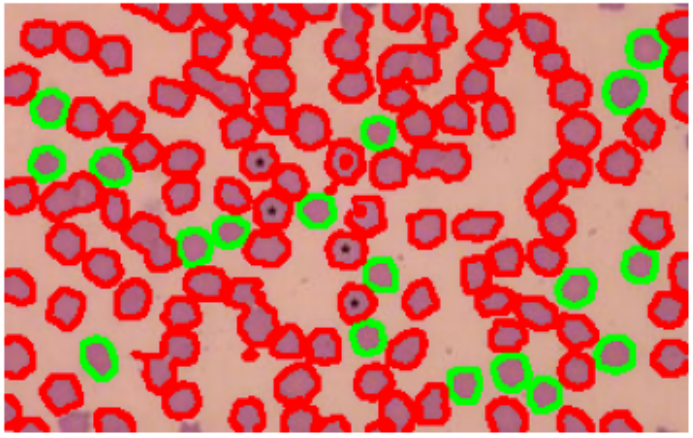
Classified Cells



Original Image



Classified Cells



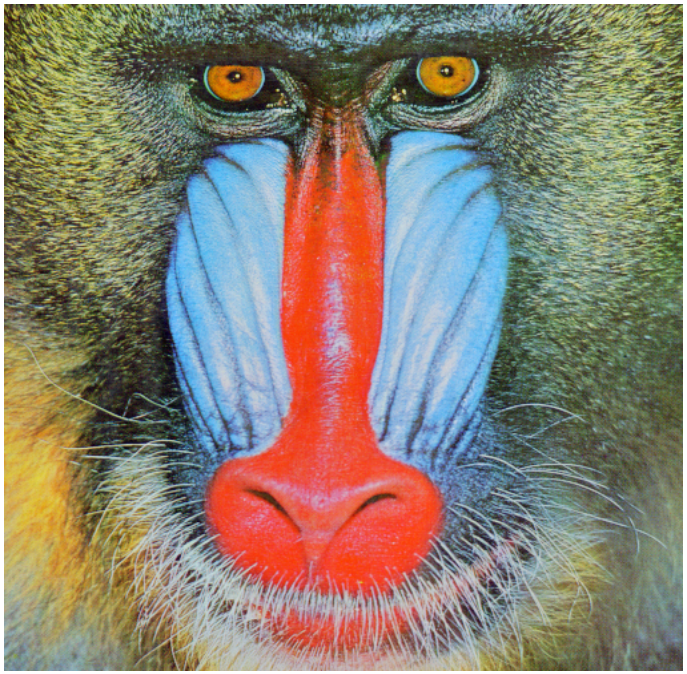
2F: Did the software have any difficulties in the shape analysis and classification? What caused the issues?	Yes.I still don't think that only those selected cells (with black marks on it) are damaged. In terms of shape, there are more other similar cells. If we're taking account of the circularity, compactness and eccentricity, all damaged cells should have similar features. So that's why my algorithm detects other cells as damaged as well. Other than that, cells that are clinging together are not detected correctly so the software thinks they are not circular,etc so it detects them as damaged.

SECTION 3 - Image registration

For this section, you need to pick two images to register to each other. In class we warped Andy Adler [Carleton.ca] to a Mandrill [Wikipedia]. You can choose any images which can reasonably be mapped to each other. Medical images are fine, but so are pictures of people or animals. In this question, use three different approaches.

- A manual landmark–based approach
- An image-feature approach (SIFT, Harris corners)
- A global image approach (matching histograms, entropy)

For this question, you may (but are not required to) use a LLM for suggestions. If you do, indicate what the LLM suggested, as well as any modifications you made to improve the result.



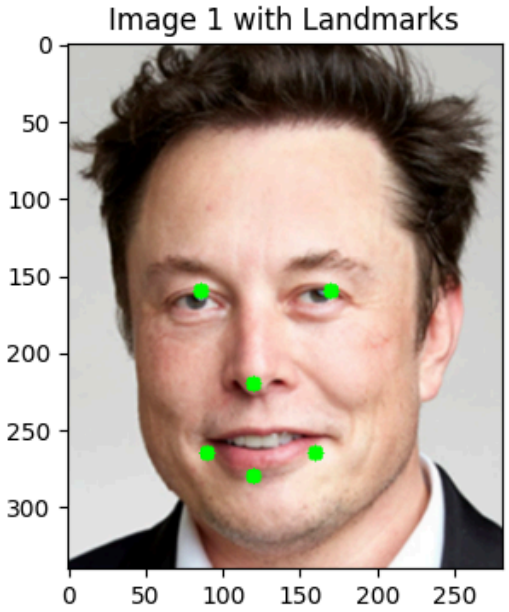
Question:	Answer / Code / Image
3A: What images did you choose? What kind of registration performance do you anticipate? Is this a hard or easy problem? Identify which is the fixed and moving image	I chose the above images, Elon and Mandrill. I didn't want to use the Mandrill. But It was the easiest one. This is a complex task as the software should be able to find perfect matching landmarks, then the registration is another challenge, the scaling, background effects, etc. Elon's face will be transformed to look like a Mandrill.
3B: Write software to perform manual, landmark-based, registration.	Full code: Here `elon = cv2.imread('elon.png')` `monk = cv2.imread('monk2.png')` `points_image1 = np.array([[86, 160], [170, 160], [120, 220], [90, 265], [120, 280], [160, 265]])` `points_image2 = np.array([[170, 50], [330, 50], [230, 360], [110, 420], [230, 480], [400,`

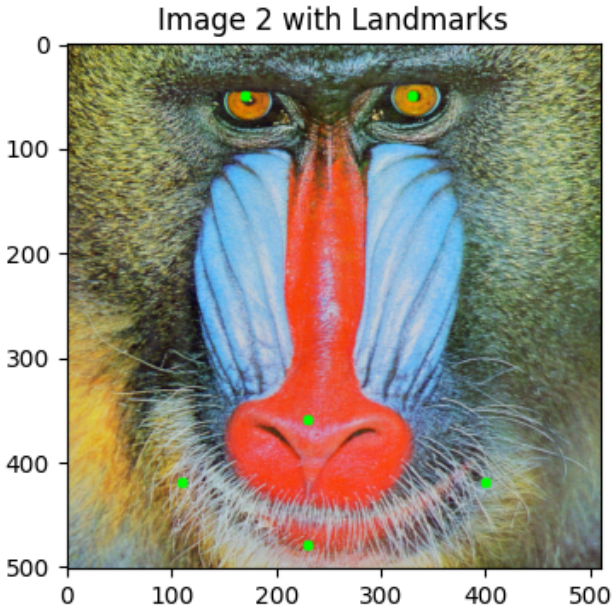
```
420]])
matrix_homography, _ = cv2.findHomography(points_image1, points_image2, cv2.RANSAC)
registered_image_homography = cv2.warpPerspective(elon, matrix_homography, (monk.shape[1],
monk.shape[0]))
plt.imshow(cv2.cvtColor(registered_image_homography, cv2.COLOR_BGR2RGB))
plt.show()
```

Reference: https://docs.opencv.org/3.4/d1/de0/tutorial_py_feature_homography.html and https://theailearner.com/tag/cv2-warpperspective/

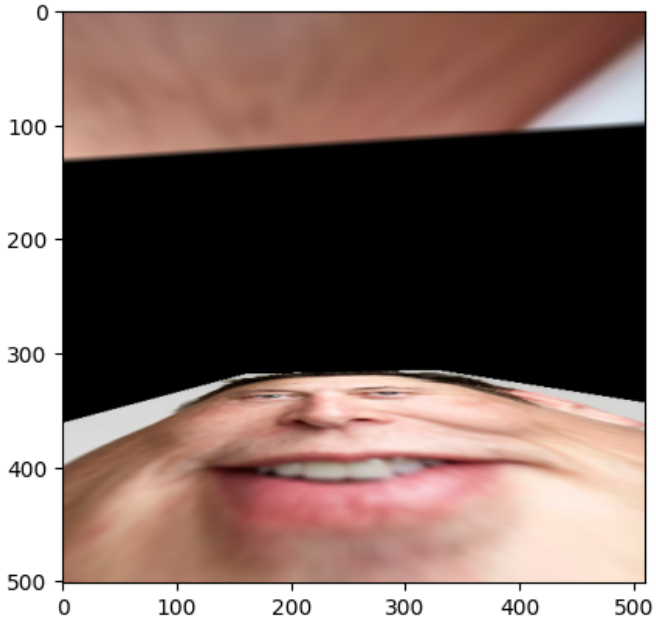
3C: Show results of 3B. What worked well, and what worked less well?

1- With less landmark points:





This is the result:



But If I add one more:

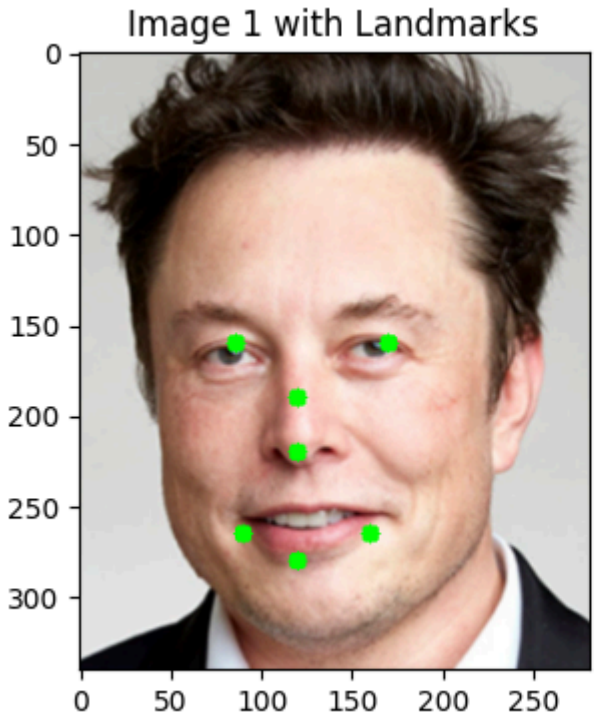


Image 1 with Landmarks
0
50
100
150
200
250
300
0
50
100
150
200
250

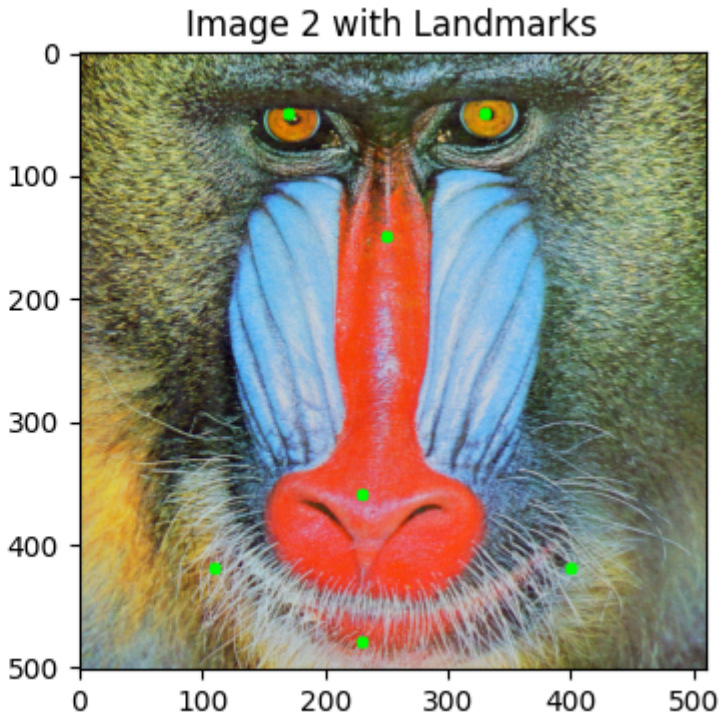
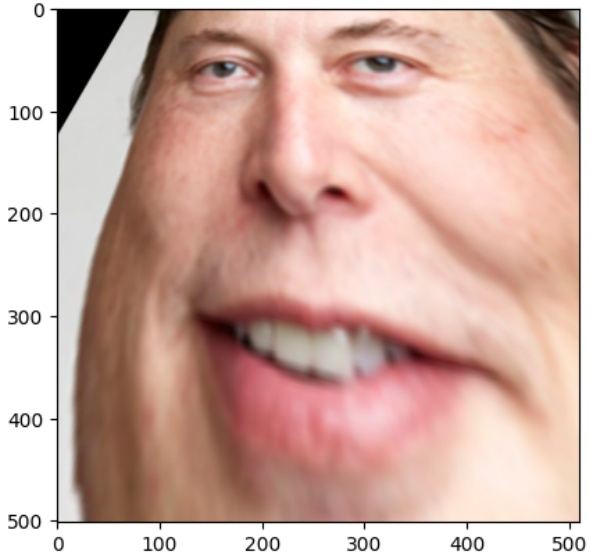
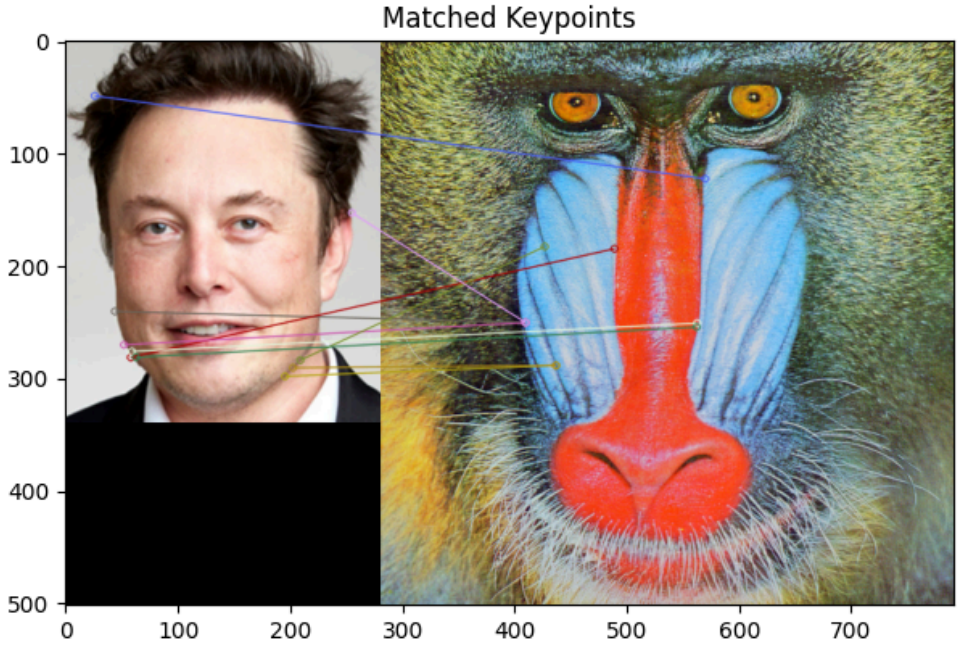


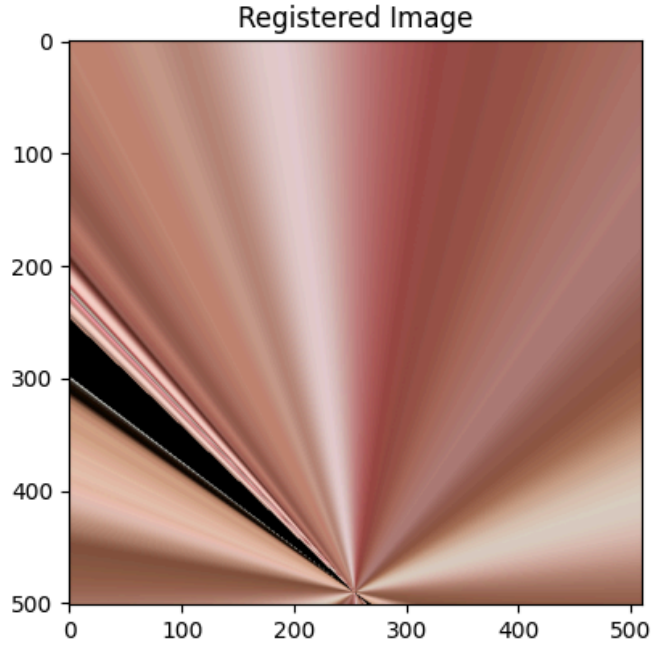
Image 2 with Landmarks
0
100
200
300
400
500
0
100
200
300
400
500

I get:

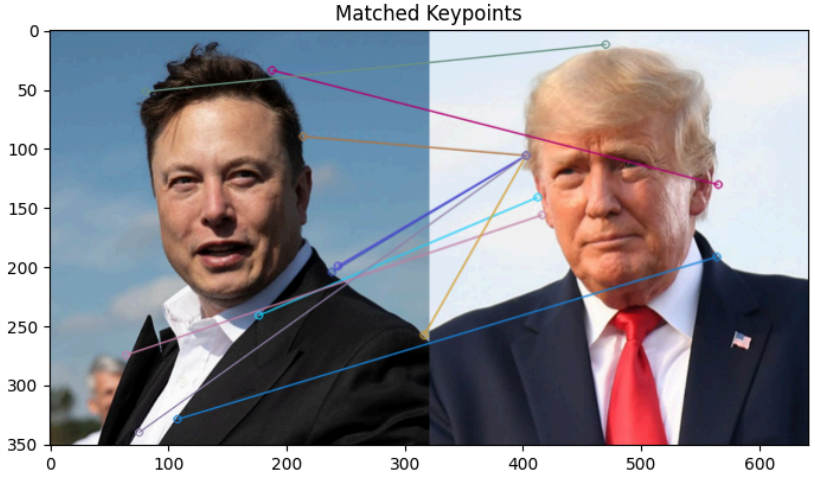

0
100
200
300
400
500
0
100
200
300
400
500

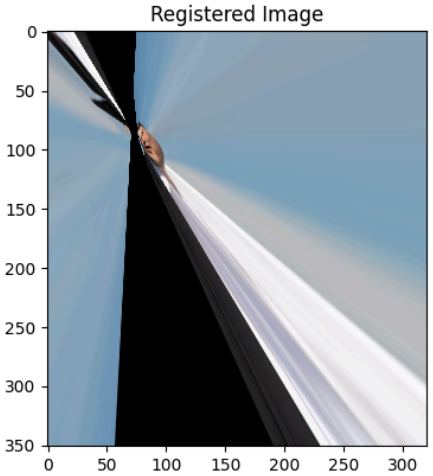
3D: Write software to perform automatic, keypoint-based registration.	Full code Here `sift = cv2.SIFT_create()` `#sift finds keypoints in two images` `keypoints1, descriptors1 = sift.detectAndCompute(gray1, None)` `keypoints2, descriptors2 = sift.detectAndCompute(gray2, None)` `#then using BFMatcher we find top matches between these two keypoints` `bf = cv2.BFMatcher()` `matches = bf.match(descriptors1, descriptors2)` `matches = sorted(matches, key=lambda x: x.distance)` `#then we use top 10 keypoints.` `matched_image = cv2.drawMatches(elon, keypoints1, monk, keypoints2, matches[:10], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)` Reference: https://www.geeksforgeeks.org/image-registration-using-opencv-python/
3E: Show results of 3D, including the keypoints identified and how the correspond. What worked well, and what worked less well?	Well, since the two image are not similar the SIFT failed to find perfect match keypoints so the registered image has no meaning:


Matched Keypoints
0
100
200
300
400
500
0
100
200
300
400
500
600
700

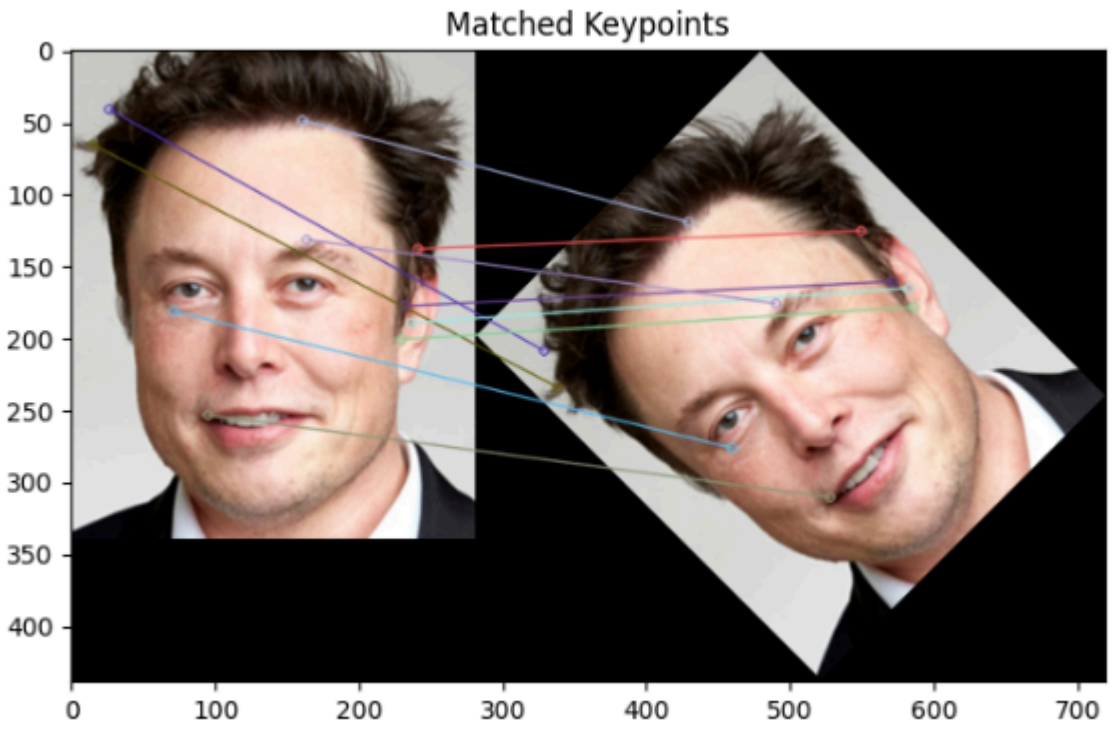

Registered Image
0
100
200
300
400
500
0
100
200
300
400
500

So I decided to use a trump face instead of mandrill. However, it was poor in detecting the match landmarks:





I don’t think algorithms such as SIFT would be that efficient and smart to detect left eyes from two different person as the same landmark. I think it would be beneficial for the task where the original image was rotated and skewed, and we want to find the matching landmarks from both pictures. I rotated elon’s face 45 degree and as expected it could find out the corresponding landmark:

	
3F: Write software to perform image-based registration (e.g. Histogram matching, Entropy)	`def histogram_matching(source_image, reference_image):` `  matched = exposure.match_histograms(source_image, reference_image)` `  return matched`

3G: Show results of 3F. What worked well, and what worked less well?	
3H: Compare the approaches. What worked well and less well for your images? Did the results match your expectations in 3A and why or why not?	Well, Registration for based on manual landmark selection was a bit challenging but the output matched my expectation. It was very sensitive to the choice of landmarks. For the SIFT, well I didn't expect to be able to detect the corresponding landmarks between two images that are not similar to each other (from the machine perspective, not our biased mind that can map the eyes of mandrill and elon) So I tried a human instead of mandrill, still couldn't perform well. But lastly I used the same image but tried to rotate it 45 degrees. It could find the corresponding landmarks as expected. And for the last part, If I understood the question correctly, based on histogram matching we only can register the pixel intensities between two images. As a result, in 3G I can see the mandrill's color pattern in Elon's image.