דו"ח מטלה 2 עיבוד תמונה.

מגישים:
יואב סימני: 208774315
עדי רז:206875874

שאלה 1:
בשאלה זו התבקשנו ליצור את שלושת הפונקציות המתאימות לפעולות הבאות:
Brightness and contrast stretching
Gamma correction
Histogram equalization

תמונה 1 (התמונה האפורה הלא ברורה):

בחרנו לבצע תיקון Histogram equalization.
תיקון זה מסייע בהבחנה בין גוונים דומים.
תיקון זה בעצם "מרווח" את היסטוגרמת התמונה ומסייע להבחין בין הצבעים בתמונה.

בחרנו תיקון זה כיוון שהתמונה המקורית הייתה לא ברורה ורוב הפיקסלים שלה היו בעלי ערכי אפור קרובים, לכן היה קשה להבחין מה רואים בתמונה.

לאחר התיקון, נראה כי מדובר בתמונה של כוכב וניתן לזהות מכתשים בתמונה, שקודם לא היה ניתן להבחין.
התיקון הגדיל את ההבדלים בין גווני האפור השונים בתמונה וסייע להבדיל ביניהם.

תמונה 2 (תמונת הרחוב):

בחרנו לבצע תיקון Gamma correction.
תיקון זה מתבצע כיום כמעט בכל תמונה באופן אוטומטי כיוון שנדרש לבצע תיקון זה לטובת הצגת התמונה במסך. זאת כיוון שמסכים רבים אינם מציגים באופן פרופורציונלי ישירות לקלט ולכן התיקון מתקיים בערכי הפיקסלים כבר במועד הצילום.

במקרה זה, הוספנו את התיקון לתמונה קיימת כיוון שהתיקון מסייע בשיפור ניגודיות.
תמונת הרחוב שצולמה הייתה בעיקר חשוכה מאוד או בהירה מאוד (כלומר בעלת ניגודיות גבוהה).
התיקון סייע בכך שהוריד את ניגודית התמונה ויצר מעבר "חלק" יותר של הפיקסלים בתמונה.

תמונה 3 (אישה בשמלה):

בחרנו לבצע תיקון Brightness and contrast stretching.
תיקון זה מסייע לתמונות שערכי הפיקסלים שלהן מופיעים בטווח צר.
התיקון מבצע "מתיחה" של ערכי הפיקסלים ובכך משתמש בקשת רחבה יותר של ערכי פיקסלים.
דבר זה מגדיל את ניגודיות התמונה ועוזר להבחין בפרטים נוספים בתמונה.

בתמונה המקורית הופיעו הרבה פיקסלים בערכים גבוהים (בהירים) ואחרי התיקון בהירות התמונה ירדה.
קשה להגיד האם התיקון היה חיוני לתמונה זו.
נראה כי למרות שהתמונה המקורית הייתה בהירה, היה ניתן בכל זאת לזהות את כל פרטי התמונה בצורה ברורה.

שאלה 2:

1. כדי לקבל מטריצות נכונות, עברנו על כל חלק בפאזל והשווינו עם התמונה הראשונה.
חיפשנו נקודות במרחק רב ככל האפשר בין אחת לשניה, כדי שמטריצת הטרנספורמציה תהיה מדויקת ככל האפשר.
עבור חלקים שעברו טרנספורמציה אפינית, השתמשנו ב 3 נקודות השוואה, ואילו בטרנספורמציה הומוגרפית השתמשנו ב 4 נקודות השוואה.
2. לכל אחת מהתמונות קראנו לפונקציה prepare puzzle, שהחזירה לנו את כל המידע הדרוש לתחילת עבודה על החלקים בפאזל.
3. פונקציית get_transform :

```
def get_transform(matches, is_affine):
    src_points, dst_points = matches[:, 0], matches[:, 1]
    # Add your code here
    # Use the source and destination points to estimate the transform matrix
    if is_affine:
        T, _ = cv2.estimateAffine2D(src_points, dst_points)
    else:
        T, _ = cv2.findHomography(src_points, dst_points)
    return T
```

הפונקציה מבחינה בין חלקי פאזל שעברו טרנספורמציה אפינית לבין טרנספורמציה הומוגרפית.
הפונקציה מחזירה את מטריצת הטרנספורמציה של החלק לעומת התמונה הראשונה.
4. פונקציית inverse_transform_target_image :

```
def inverse_transform_target_image(target_img, original_transform, output_size, is_affine):
    # Add your code here
    # Find the Inverse Matrix and inverse the image accordingly
    if is_affine:
        inverse_transform = cv2.invertAffineTransform(original_transform)
        warped = cv2.warpAffine(target_img, inverse_transform, output_size, output_size, flags=cv2.INTER_LINEAR)
    else:
        inv_transform = np.linalg.inv(original_transform)
        warped = cv2.warpPerspective(target_img, inv_transform, output_size, flags=cv2.INTER_LINEAR)
    return warped
```

גם פונקציה זו מפרידה בין אפינית להומוגרפית.
הפונקציה מוצאת את ה inverse של הטרנספורמציה שמצאנו בפונקציה get_transform ואז מפעילה את המטריצה על החלק, ומחזירה את החלק כשהוא מותאם לגודל של הקנבס ועבר טרנספורמציה לצורה המקורית(כמו תמונה 1).
5. פונקציית stitch:

```
def stitch(image1, image2):
    # Add your code here
    # Compare each pixel in image1 and image2 and take the pixel with the max value
    stitched_image = cv2.max(image1, image2)
    return stitched_image
```

הפונקציה מקבלת 2 חלקים ומחברת ביניהם לכדי תמונה אחת.
האופן שבו זה מתבצע זה ע"י לקיחת הערך המקסימלי מבין 2 התמונות עבור כל פיקסל.
במצבים שבהם אין חפיפה בין 2 התמונות - הפיקסל שנבחר הוא הפיקסל של התמונה שקיימת.
במצבים שבהם יש חפיפה בין 2 התמונות - הפיקסל שנבחר הוא אותו פיקסל(בהנחה והטרנספורמציות היו מדויקות לחלוטין).

6. כדי לקבל את התמונה המלאה, נבצע stitch בין התמונה הראשונה לשנייה.
נשתמש בתוצאה ונעשה stitch בינה לבין התמונה השלישית וכ'ו...
התוצאה הסופית לאחר stitch בין כל חלקי הפאזל זו התמונה השלמה

```
# To create the final puzzle we will start with image1 and then stitch the other images
final_puzzle = img1
for i, file_name in enumerate(os.listdir(pieces_pth)):
    # Image 1 is already part of the final puzzle so we will skip it
    if (file_name == 'piece_1.jpg'):
        continue
    T = get_transform(matches[i - 1], is_affine)
    curr_img = cv2.imread(os.path.join(pieces_pth, file_name))
    transformed_image = inverse_transform_target_image(curr_img, T, output_size: (weight, height), is_affine)

    # save the inverted image
    output_path = os.path.join(edited, f'{file_name.split('.')[0]}_relative.jpg')
    cv2.imwrite(output_path, transformed_image)

    # Stitch the inverted image with the final puzzle
    final_puzzle = stitch(final_puzzle, transformed_image)

sol_file = f'solution.jpg'
cv2.imwrite(os.path.join(puzzle, sol_file), final_puzzle)
```

התמונות שקיבלנו :