שאלה 1

בתמונה 1 התיקון שלדעתנו הכי טוב הוא איזון היסטוגרמה כיוון שהיא משפרת את הניגודיות בתמונה ומדגישה חלקים בתמונה "שנבלעו" ברקע מסביבם ולא ראינו אותם.

בתמונה המתוקנת ניתן לראות יותר מכתשים והרים כיוון שפיזרנו את דרגות האפור (בתמונה המקורית היא ברובה סביב דרגת אפור מסוימת) של התמונה ולכן ניתן לראות יותר פרטים ושינויים בקרקע.

בתמונה 2 התיקון שלדעתנו הכי טוב הוא תיקון גמא כיוון שערך הגמא תבהיר את האזורים הכהים מבלי להגדיל יותר מידי את הבהירות של האזורים הבהירים.

כאשר בוחרים בגמא קטן מ1 הוא מכהה אזורים בהירים ומבהיר אזורים כהים.

כאשר בוחרים גמא גדול מ1 הוא מבהיר את כל התמונה בדגש על האזורים הכהים.

בחרגנו בגמא = 2.2 בשביל לאפשר להבהיר את האזורים הכהים ברקע בלי לפגוע באיזון האור הכולל של התמונה ובתמונה המתוקנת נשמר המראה הטבעי, בנוסף גמא = 2.2 הוא ערך נפוץ לשיפור בהירות בתמונות כהות.

בתמונה 3 החלטנו שאין צורך לבצע תיקונים בתמונה כיוון שאין בעיות של פיזור לא אחיד של גוונים או בעיות של ניגודיות וחשיפה.

אחרי שביצענו מגוון של שינויים בתמונה הגענו למסקנה שתמונת המקור נראת יותר טוב ויותר "אמיתית" מהתמונות "המתוקנות" לכן בחרנו להשאיר אותה כמו שהיא.

(תמונת התוצאה שהיא בעצם המקורית היא באפור

כיוון שimread קורא אותה בgrayscale)

שאלה 2

get_transform(matches, is_affine)

הפונקציה מקבלת מערך של נקודות התאמה בין שני סטים של תמונות (נקודות מקור ויעד) ומבצעת חישוב של מטריצת טרנספורמציה שתתאים לשתי קבוצות הנקודות. אם מדובר בטרנספורמציה אפינית (is_affine=True), היא מחשבת מטריצה אפינית באמצעות הפונקציה cv2.estimateAffine2D. אחרת, היא מחשבת מטריצת הומוגרפיה עם cv2.findHomography. מטריצה זו משמשת בהמשך לשינוי צורת התמונה כך שתתאים לתמונת הפאזל.

```
Starting puzzle_homography_1
Source Points: [[478 145]
 [566 331]
 [308 332]
 [454 326]]
Destination Points: [[ 60 119]
 [186  47]
 [221 219]
 [196 120]]
Transformation Matrix: [[-7.49576352e-02  5.48333359e-01  1.37345131e+01]
 [-5.14264389e-01 -1.61298766e-01  3.83076015e+02]
 [ 1.52404940e-04 -7.99755529e-04  1.00000000e+00]]
Source Points: [[308 332]
 [295 123]
 [306 452]
 [436 380]]
Destination Points: [[180 192]
 [190  36]
 [170 278]
 [275 240]]
Transformation Matrix: [[-2.67955838e-02 -1.88131471e-02  1.11828939e+02]
 [-3.36596592e-01  4.25559276e-01  6.62046675e+01]
 [-1.63111580e-03  1.29835917e-04  1.00000000e+00]]
```

```
Starting puzzle_affine_2
Source Points: [[440 448]
 [306 239]
 [451 209]]
Destination Points: [[ 59  29]
 [235  79]
 [173 173]]
Transformation Matrix: [[-5.31332848e-01 -5.01442098e-01  5.17432513e+02]
 [ 5.28652586e-01 -5.78179170e-01  5.54171304e+01]]
Source Points: [[492 376]
 [377 326]
 [508 243]]
Destination Points: [[227 149]
 [195 248]
 [127 150]]
Transformation Matrix: [[-4.62255359e-02  7.46318733e-01 -3.08728798e+01]
 [-8.14973594e-01 -1.05560733e-01  5.89657844e+02]]
Source Points: [[306 239]
 [258 366]
 [360 420]]
Destination Points: [[ 61 135]
 [ 96  29]
 [  7  31]]
Transformation Matrix: [[-8.48642738e-01 -4.51563103e-02  3.31477036e+02]
 [ 1.55795703e-01 -7.75762254e-01  2.72733694e+02]]
Source Points: [[306 239]
 [258 366]
 [429 245]]
Destination Points: [[106 163]
 [ 11 121]
 [ 98 259]]
Transformation Matrix: [[-2.80344459e-02 -7.58627192e-01  2.95890439e+02]
 [ 7.82198755e-01 -3.50744861e-02 -6.79700170e+01]]
```

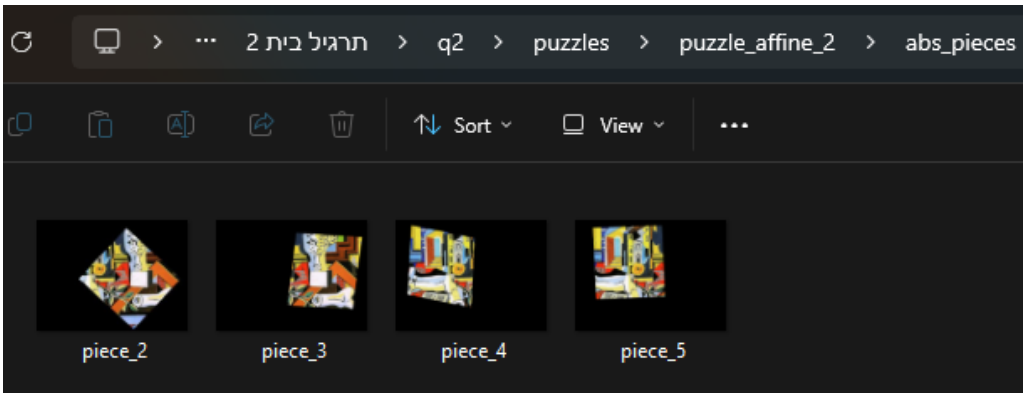
```
Starting puzzle_affine_1
Source Points: [[263  94]
 [492 415]
 [552 138]]
Destination Points: [[256 115]
 [ 16 276]
 [235 297]]
Transformation Matrix: [[ 4.61828692e-02 -7.80610209e-01  3.17231265e+02]
 [ 6.20826430e-01  5.86627647e-02 -5.37916510e+01]]
```

stitch(img1, img2)

הפונקציה מחברת שתי תמונות בגודל זהה לכדי תמונה אחת. היא עושה זאת על ידי לקיחת הערכים המקסימליים של הפיקסלים בשתי התמונות בכל נקודה (Blend), כך שהתוצאה תהיה תמונה שמשלבת את שתי התמונות.
השימוש בפונקציה מתאים במיוחד למקרים שבהם התמונות עשויות לחפוף חלקית.

inverse_transform_target_image(target_img, original_transform, output_size)

הפונקציה אחראית להחזיר תמונה שעברה טרנספורמציה למיקומה המקורי. היא בודקת אם מטריצת הטרנספורמציה שהתקבלה היא אפינית או הומוגרפית, ובהתאם לכך מחשבת את מטריצת הטרנספורמציה ההפוכה.
הפונקציה משתמשת ב-cv2.warpAffine או ב-cv2.warpPerspective כדי ליישם את הטרנספורמציה ההפוכה ולהתאים את התמונה לגודל שצוין (output_size).



הרצת הקוד הראשי(main)

החלק המרכזי בקוד מטפל ברשימת פאזלים שמוגדרים בתיקיות. עבור כל פאזל, הוא טוען את החלק הראשון כבסיס לפתרון. לאחר מכן, הוא עובר על יתר החלקים בתיקייה, מחשב עבורם את מטריצת הטרנספורמציה על סמך הנתונים מקובץ ההתאמות, ומבצע טרנספורמציה הפוכה על כל חלק כדי למקם אותו במקום הנכון בפאזל. לבסוף, כל חלק חדש מחובר לתמונה הראשית באמצעות הפונקציה stitch. התוצאה היא תמונה מלאה של הפאזל שנשמרת בקובץ solution.jpg.