Image Processing

HW2

Lian Natour 207300443

Mohammad Mhneha 315649814

Problem 1:
Image 1:
Histogram equalization was used for the first image to enhance its contrast. This method redistributes the image's intensity levels, making the dark regions lighter and the light regions slightly darker, which can help in revealing hidden details in both shadows and highlights.
here is the fixed image we got :

Image 2:
We used gamma correction with a gamma value of 2.1 on Image ID 2 to reduce the brightness of the streetlight and enhance details in the darker areas, making the night scene clearer without altering the overall atmosphere.
here is the fixed image we got :

Image 3:
For the third image, no adjustments were made. An evaluation of the image indicated that it was already well-balanced in terms of exposure and contrast. Any further changes could have compromised the image quality rather than enhancing it.

Problem 2 :

Function: get_transform

The get_transform function calculates the transformation matrix required to align one image with another based on matched points between them. It supports both affine and projective transformations. For affine transformations, it uses three pairs of points to create a 2x3 matrix with cv2.getAffineTransform. For projective transformations, it uses four or more point pairs to compute a 3x3 matrix with cv2.findHomography. This function ensures the points provided are adequate for the transformation type and raises an error if they are not. It is crucial for stitching different puzzle pieces together accurately.

Function: inverse_transform_target_image

This function applies an inverse transformation to position a puzzle piece accurately on the main canvas. It checks if the transformation is affine or projective and uses OpenCV functions cv2.warpAffine or cv2.warpPerspective respectively to adjust the image. This ensures each puzzle piece fits precisely within the overall puzzle layout.

Function : stitch

The stitch function merges two images by carefully blending them together. It first identifies the black (empty) areas in the first image and replaces these parts with non-black pixels from the second image. Where both images have pixel values, it averages them to smooth the transition, preventing harsh edges. This blending ensures that the merged image appears seamless , crucial for reconstructing the entire puzzle accurately.

The main script automates the assembly of puzzles by processing images from specified directories. It begins by setting up directories for each puzzle type, then reads and organizes the images for transformation. Each image is aligned using calculated transformation matrices and then merged into a single final image using a stitching function. Errors are handled and reported during processing, and successful completions are confirmed with a saved final puzzle image. The script ensures that each puzzle piece fits perfectly within the designated canvas, producing a coherent and completed puzzle.