cs512 f24 - assignment 1 (image formation)

Due by: 9/22/2024

General Instructions:

In this assignment, you will practice working with geometric transformations in computer vision, focusing on representation in homogeneous coordinates (2D and 3D), image projection in camera coordinates, and transformations such as translation, rotation, and scaling. You will be using Python, along with NumPy and OpenCV, to implement these concepts.

Your tasks include:

- Writing Python functions to perform various transformations.
- Applying these transformations to points and images.
- Computing transformation matrices manually before using OpenCV functions like `warpAffine`.

For each question, follow the instructions closely, and be sure to test your code with sample inputs.

Submission instructions:

- Follow the submission instructions of assignment 0.
- Submit a fully executed Python notebook with no gaps in execution. To receive full credit, please ensure the following:
  - The notebook includes all cell outputs and contains no error messages.
  - It is easy to match each problem with its corresponding code solution using clear markdown or comments (e.g., "Problem 2").
  - Re-run your notebook before submitting to ensure that cells are numbered sequentially, starting at [1].

Questions:

1. **Homogeneous Coordinates Representation (2D)**: Let $\mathbf{x} = (2, 3)$ be a point in 2D Cartesian coordinates.
   - Convert the point $\mathbf{x}$ into homogeneous coordinates $\mathbf{x_H}$.
   - Find another point in 2D homogeneous coordinates that is equivalent to the same point $\mathbf{x}$ (use a different scale).
   - Convert both homogeneous coordinates back to Cartesian coordinates and verify they represent the same point.

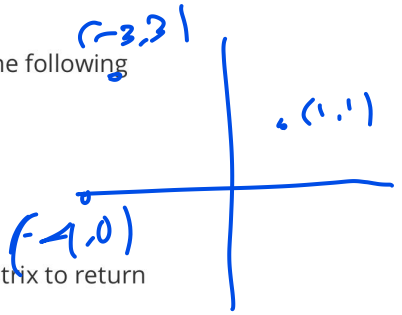
2. **Homogeneous Coordinates Representation (3D)**: Let $\mathbf{y} = (4, 5, 6)$ be a point in 3D Cartesian coordinates.
   - Convert $\mathbf{y}$ to its homogeneous coordinates representation.
   - Now, consider the homogeneous point $\mathbf{y_H} = (4, 5, 6, 1)$. Scale it by a factor of 2 and convert back to Cartesian coordinates.
   - Verify the result matches the original point $\mathbf{y}$.

3. **Affine Transformations in 2D**: Given a point $\mathbf{p} = (1, 2)$ in 2D, apply the following transformations in the specified order:
   - Scale by a factor of 3.
   - Rotate counterclockwise by 45°.
   - Translate by $(2, 3)$.

   Compute the transformation matrix in homogeneous coordinates for each step. Then, apply the combined transformation to the point $\mathbf{p}$. Finally, verify your result by applying the transformations using NumPy.

4. **Inverse Transformations (2D)**: Let $\mathbf{q} = (3, 4)$ be a point in 2D, and assume the following transformations have been applied:
   - Scale by 2.
   - Rotate by 30°.
   - Translate by $(5, 5)$.

   Compute the matrix that reverses these transformations. Apply the inverse matrix to return $\mathbf{q}$ to its original position. Verify the result using NumPy.



5. **Transformations Between 3D Coordinate Systems**: Consider two 3D coordinate systems:
   - System A has its origin at $(1, 1, 1)$ and is aligned with the world axes.
   - System B has its origin at $(2, 2, 2)$ and is rotated by 90° around the Z-axis.

   A point $\mathbf{r} = (3, 3, 3)$ is given in System A. Compute the transformation matrix to convert this point to System B's coordinates and apply it to $\mathbf{r}$.

6. **Projection in Camera Coordinates (3D to 2D)**: Let a point in 3D camera coordinates be $\mathbf{P} = (1, 2, 5)$. You are given the following camera intrinsic matrix K:

$$K = \begin{pmatrix} 1000 & 0 & 500 \\ 0 & 1000 & 500 \\ 0 & 0 & 1 \end{pmatrix}$$

   - Project the point $\mathbf{P}$ onto the 2D image plane using the intrinsic matrix K.
   - Compute the 2D pixel coordinates of the projected point.
   - Explain the meaning for the elements of K.

7. **General Camera Model (3D World Points to 2D Image Points)**: Given the following camera parameters:

- Intrinsic matrix K: $K = \begin{pmatrix} 1200 & 0 & 640 \\ 0 & 1200 & 360 \\ 0 & 0 & 1 \end{pmatrix}$
- Camera rotation matrix $R = \text{identity matrix}$.
- Camera translation $\mathbf{t} = (0, 0, -10)$.

Transform a 3D world point $\mathbf{X} = (2, 3, 4)$ from world coordinates to 2D image coordinates. You will need to:

- Compute the extrinsic matrix from the rotation and translation.
- Project the point onto the image plane.
- Verify the 2D coordinates in pixels.

8. **Image Transformation using `cv2.warpAffine` (2D)**: Load an image using OpenCV. You will apply a combination of transformations (translation, rotation, and scaling) to the image.
   - First, compute the affine transformation matrix manually using translation $(50, 30)$, rotation by 30°, and scaling by a factor of 1.5.
   - Use OpenCV's `cv2.warpAffine` function to apply the transformation to the image.
   - Display both the original and transformed images.

9. **Order of Transformations**: Let $\mathbf{s} = (1, 1)$ be a point in 2D.
   - Apply the following transformations in the order specified:
     1. Rotate by 90°.
     2. Scale by a factor of 2.
     3. Translate by $(-1, 1)$.

   Now reverse the order of the transformations and apply them again. Observe and explain how the final position of $\mathbf{s}$ changes with different transformation orders.