Computer Vision Programming Exercises: 2D Transformations and Image Homographies

Instructor: PhD Thái Đình Kim
Khoa học Computer Vision, Spring 2025
VNU-IS March 16, 2025

Lectures 7 & 8: 2D Transformations and Image Homographies
`https://vnuis.edu.vn/`

This document provides programming exercises for students to practice 2D transformations and image homographies in computer vision using Python and OpenCV.

Mục lục

General Instructions

- **Objective:** These exercises cover 2D transformations (Lecture 7) and image homographies (Lecture 8), helping students understand and implement fundamental computer vision techniques using Python and OpenCV.
- **Tools:** Use Python 3.x, OpenCV (`cv2`), NumPy, and Matplotlib. Install via:

```
pip install opencv-python numpy matplotlib
```

- **Input Images:** Use a single image (e.g., `lena.jpg`) for Lecture 7 exercises, and two overlapping images for Lecture 8 exercises. Sample images can be sourced online if needed.
- **Expected Output:** Transformed images (Lecture 7) or stitched panoramas (Lecture 8), with cropped areas filled with black or blended appropriately.
- **Submission:** Submit Python code and a report with input/output images and brief explanations for each exercise.

1 Lecture 7: 2D Transformations

Programming Exercises

Exercise 1: Translation

- **Objective:** Translate an image 100 pixels to the right and 50 pixels downward.
- **Input Image:** A single image (e.g., `lena.jpg`).
- **Expected Output:** The translated image, with cropped areas filled with black.
- **Algorithm Applied:** Translation is represented by:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $t_x = 100$, $t_y = 50$. OpenCV uses a 2×3 matrix with `cv2.warpAffine`.

Exercise 2: Rotation

- **Objective:** Rotate an image by 45 degrees around its center.
- **Input Image:** A single image (e.g., `lena.jpg`).
- **Expected Output:** The rotated image, with cropped areas filled with black.
- **Algorithm Applied:** Rotation is represented by:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $\theta = 45°$. OpenCV's `cv2.getRotationMatrix2D` generates a 2×3 matrix for `cv2.warpAffine`.

Exercise 3: Scaling

- **Objective:** Scale an image up by a factor of 1.5 in both x and y directions.
- **Input Image:** A single image (e.g., `lena.jpg`).
- **Expected Output:** The scaled image, with cropped areas filled with black.
- **Algorithm Applied:** Scaling is represented by:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $s_x = s_y = 1.5$. OpenCV uses a 2×3 matrix with `cv2.warpAffine`.

Exercise 4: Shearing

- **Objective:** Apply a shearing transformation along the x-axis with a shear factor of 0.5.
- **Input Image:** A single image (e.g., `lena.jpg`).
- **Expected Output:** The sheared image, with cropped areas filled with black.
- **Algorithm Applied:** Shearing is represented by:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $s = 0.5$. OpenCV uses a 2×3 matrix with `cv2.warpAffine`.

Exercise 5: General Affine Transformation

- **Objective:** Apply an affine transformation based on 3 point correspondences.
- **Input Image:** A single image (e.g., `lena.jpg`).
- **Expected Output:** The affine-transformed image, with cropped areas filled with black.
- **Algorithm Applied:** Affine transformation is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  Use `cv2.getAffineTransform` with 3 point pairs, then apply with `cv2.warpAffine`.

Exercise 6: Affine Transformation with Least Squares

- **Objective:** Determine an affine transformation from multiple point correspondences using Least Squares.
- **Input Image:** A single image (e.g., `lena.jpg`).
- **Expected Output:** The affine-transformed image, with cropped areas filled with black.
- **Algorithm Applied:** Solve:

$$x' = ax + by + t_x, \quad y' = cx + dy + t_y$$

  Construct $Ax = b$, where A is $2n \times 6$, $x = [a, b, t_x, c, d, t_y]$, and solve using `np.linalg.lstsq`.

2 Lecture 8: Image Homographies

Programming Exercises

Exercise 1: Applying a Homography Transformation

- **Objective:** Apply a manually defined homography matrix to warp an image.
- **Input Image:** A single image (e.g., `lena.jpg`).
- **Expected Output:** The warped image, with cropped areas filled with black.
- **Algorithm Applied:** Homography is:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  Apply using `cv2.warpPerspective`.

Exercise 2: Computing Homography with DLT

- **Objective:** Compute a homography matrix H using DLT from 4+ point correspondences.
- **Input Images:** Two overlapping images.
- **Expected Output:** The second image warped onto the first's perspective.
- **Algorithm Applied:** Solve $Ah = 0$, where:

$$A_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

  Use SVD, reshape h to H, and apply with `cv2.warpPerspective`.

Exercise 3: Feature Detection and Matching

- **Objective:** Detect and match feature points between two images using SIFT.
- **Input Images:** Two overlapping images.
- **Expected Output:** Visualization of matched feature points.
- **Algorithm Applied:** Use `cv2.SIFT_create()`, `cv2.BFMatcher`, and `cv2.drawMatches` with a ratio test.

Exercise 4: Estimating Homography with RANSAC

- **Objective:** Estimate a homography matrix H using RANSAC.
- **Input Images:** Two overlapping images.
- **Expected Output:** The second image warped onto the first's perspective.
- **Algorithm Applied:** Use `cv2.findHomography` with `cv2.RANSAC`, then apply with `cv2.warpPerspective`.

Exercise 5: Creating a Panorama

- **Objective:** Stitch two images into a panorama using homography and RANSAC.
- **Input Images:** Two overlapping images.
- **Expected Output:** A panoramic image.
- **Algorithm Applied:** Detect features (SIFT), estimate H (RANSAC), warp with `cv2.warpPerspective`, and blend images.

Extensions

- Experiment with different transformation parameters or feature detectors (e.g., ORB).
- Extend panorama stitching to multiple images.
- Implement advanced blending techniques (e.g., multi-band blending).

References

- Lecture 7: 2D Transformations, VNU-IS, Spring 2025.
- Lecture 8: Image Homographies, VNU-IS, Spring 2025.
- OpenCV Documentation, `https://docs.opencv.org/`.
- Hartley, R., & Zisserman, A., "Multiple View Geometry in Computer Vision".