Introduction to Computer Vision (ECSE 415)

Assignment 2: Feature Matching, Image Stitching

Due date: 11:59 PM, October 11, 2024

Please submit your assignment solutions electronically via the myCourses assignment dropbox.

The submission should include a single Jupyter Notebook consisting of the solution for the entire assignment. All images and programs should be inserted in the Jupyter Notebook, not submitted separately.

Use Python to implement all operations. Unless otherwise specified, you can use OpenCV and Numpy library functions for all parts of the assignment, or you can write your own.

Students are expected to write their own code and assignments. (Academic integrity guidelines can be found at https://www.mcgill.ca/students/srr/academicrights/integrity).

Assignments received late will be penalized by 10% per day.

Submission Instructions

Submit a single Jupyter Notebook consisting of the solution for the entire assignment.

Comment your code appropriately.

Do not forget to run Markdown cells (so that the output images appear).

Do not submit input/output images separately. Images should be displayed in the Jupyter Notebook itself. Assume input images are kept in the same directory as the codes.

Make sure that the submitted code is running without error. Describe any specific requirements for running the code in your notebook.

If external libraries were used in your code please specify their names and versions in the notebook.

We are expecting you to make a path variable at the beginning of your codebase. This should point to your working local (or Google Drive) folder. Your path variable should be defined at the top of your Jupyter Notebook. While grading, we are expecting that we just have to change the path variable once and it should allow us to run your solution smoothly. Here is an example:

```python
path = '/content/drive/Mydrive/Assignment_x/images/'
image = cv2.imread(path + 'img1.png')
```

1 Harris Corner Detection (10 points)

1. **Load and Convert Image to Grayscale:**

    - Read the image ./images/mcgill_arts_building.jpg and convert it to grayscale using an appropriate image processing library.

2. **Compute Image Derivatives:**
    - Calculate the image derivatives I_x and I_y in both the x and y directions using Sobel filters or another derivative method.
    - Compute the products of the derivatives: I_x^2, I_y^2, and I_xI_y.
3. **Apply Gaussian Filtering:**
    - Smooth the derivative products I_x^2, I_y^2, and I_xI_y using Gaussian filtering to reduce noise.
4. **Compute the Corner Response:**
    - For each pixel, calculate the corner response function using the formula: $R = det(H) - \alpha \cdot (trace(H))^2$ where H is the second moment matrix formed from the smoothed derivative products, and $\alpha = 0.05$.
5. **Non-Maximum Suppression:**
    - Perform non-maximum suppression on the corner response map to identify potential corner points by keeping only the local maxima.
6. **Detect and Mark Corners:**
    - Apply a threshold to the corner response map, and mark the local maxima that exceed the threshold as detected corners.
    - Overlay the detected corners on the original image for visualization, similar to the example shown in the demo.
7. **Experiment with Thresholds:**
    - Experiment with different threshold values to control the sensitivity of corner detection.
    - Summarize your findings on the effect of varying thresholds in a markdown cell.

2 SIFT Features (20 points)

2.1 SIFT Keypoint Matching Between Two Images

1. **Compute SIFT Keypoints and Descriptors:**
    - Detect SIFT keypoints and compute descriptors for the images ./images/graf1.png and ./images/graf2.png .
2. **Match Keypoints:**
    - Use a brute-force matching method to find correspondences between the keypoints of both images.
3. **Sort Matches by Distance:**

- Sort the matched keypoints in ascending order based on their matching distance.

4. **Display the Top Ten Matches:**
   - Visualize the top ten matched keypoints, overlaying them on both images.
5. **Plot the Matching Distances for the Top 100 Matches:**
   - Create a plot with keypoint indices on the x-axis and their corresponding matching distances on the y-axis to visualize the distribution of matching distances.

2.2 Scale Invariance

1. **Compute SIFT Keypoints for the Original Image:**
   - Detect the SIFT keypoints and compute their descriptors for `./images/graf1.png`.
2. **Scale the Original Image:**
   - Create 4 scaled versions of `./images/graf1.png` using scaling factors of 0.25, 0.6, 3, and 5.
   - Display each of the scaled images.
3. **Compute SIFT Keypoints for the Scaled Images:**
   - For each scaled image, compute the SIFT keypoints and their descriptors.
4. **Match Keypoints:**
   - Perform brute-force matching between the keypoints of the original image (`./images/graf1.png`) and each of the scaled images.
5. **Sort Matches by Distance:**
   - For each pair of images (original vs. scaled), sort the matches according to the matching distance.
6. **Display the Top Ten Matches:**
   - Visualize the top ten matches between the original and each scaled image.
7. **Plot the Matching Distances:**
   - For each pair, plot the matching distance for the top 100 keypoints, using the keypoint index on the x-axis and matching distance on the y-axis.
8. **Discuss Trends:**
   - Analyze the results and discuss how increasing the scale affects the matching distances.

2.3 Rotation Invariance

1. **Rotate Image:**
   - Rotate `./images/graf1.png` by the following angles: 30°, 75°, 90°, and 180°.
   - Display the four rotated images.
2. **Compute SIFT Keypoints for Rotated Images:**
   - For each of the four rotated images, compute SIFT keypoints and descriptors.

3. **Match Keypoints:**
    - Use a brute-force matching method to match the keypoints of the original image (`./images/graf1.png`) to each of the rotated images.
4. **Sort Matches by Distance:**
    - Sort the matched keypoints for each image pair based on the matching distance in ascending order.
5. **Display the Top Ten Matches:**
    - Visualize the top ten matched keypoints for each image pair (original vs. rotated).
6. **Plot the Matching Distances for the Top 100 Matches:**
    - Create a plot showing the matching distance for the top 100 keypoints, with keypoint indices on the x-axis and corresponding matching distances on the y-axis.
7. **Discuss Trends:**
    - Analyze how increasing the angle of rotation affects the matching distances. Explain the trends observed in the plot.

2.4 Repeat 2.1-2.3 using the image pair you obtained from assignment 1.

Image Stitching (20 points)

You are given three different views of the same scene located in the folder `./images/Q3`. Follow these steps to stitch the images together:

1. **Keypoint Detection and Matching**:
    a. Compute SIFT keypoints and descriptors for images 1 and 2.

    b. Match the keypoints between images 1 and 2. Display the 20 best matching pairs.
2. **Homography Estimation and Transformation**:
    a. Use the RANSAC method to compute the homography matrix that aligns keypoints from image 1 to image 2.

    b. Apply the computed homography to transform image 1. Image 2 should remain unchanged.
3. **Image Stitching**:
    a. Stitch the transformed image 1 with the original image 2. For better visualization, average the pixel intensity in the overlap region so that you can see patterns from both images. Name this result `image_12`. Display `image_12`.
4. **Repeat the Process for Image 3**:
    a. Compute SIFT keypoints and descriptors for `image_12` and image 3.

    b. Match the keypoints between `image_12` and image 3. Display the 20 best matching pairs.

    c. Compute the homography using RANSAC to align keypoints from `image_12` to image 3. Apply this homography to transform image 3. Image `image_12` should remain unchanged.

5. **Final Stitching**:

   a. Stitch the transformed image 3 with `image_12`. Also average the pixel intensity in the overlap region. Display the final stitched image.