CS5330 CS 5330 Pattern Recognition and Computer Vision

# Mid-Term 2nd Round: Image Stitching using a 2x2 Grid of Object Images

**Objective**:

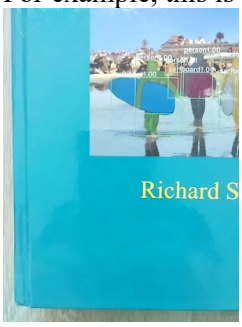
You are required to create a Python program that stitches together 4 images taken in a 2 x 2 grid format of the same object into a single image. The images may overlap slightly, and the goal is to find the correct placement for each image, stitch them into one seamless image, and save the final result.

First, Download the starter code and datasets from the provided GitHub repository.

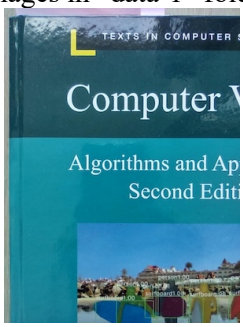
- https://github.khoury.northeastern.edu/jelee0408/CS5330-Mid-Term-Fall2024.git

**Input:**

- The input will consist of **4 images** stored in the same folder (e.g., data-1 folder), taken in a 2 x 2 grid for a single object. The images may have some overlaps.
- Each input folder will contain exactly 4 images with varying overlaps.
- For example, this is 4 images in "data-1" folder:

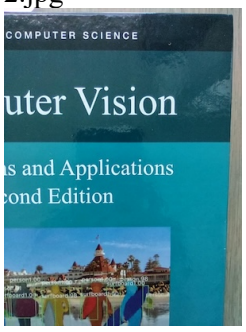



1.jpg




2.jpg


3.jpg




4.jpg

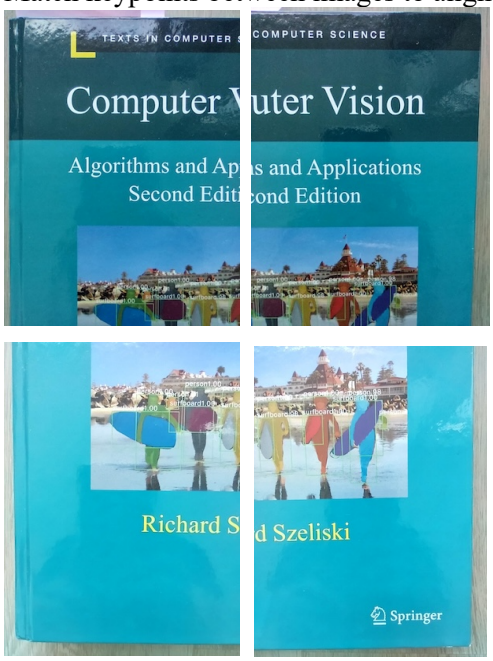
**Output:**

- The program should output a single, **stitched image** with the correct alignment.
- The final stitched image should be saved as [input_folder_name].jpg. For example, if the folder name is data-1, the stitched image should be saved as *data-1.jpg*.
- Crop any unnecessary information, such as black borders, and optimize the quality of the stitched image as much as possible.

**Steps:**

1. **Starter Code and Dataset:**
   - Download the starter code and datasets from the provided GitHub repository.
   - https://github.khoury.northeastern.edu/jelee0408/CS5330-Mid-Term-Fall2024.git

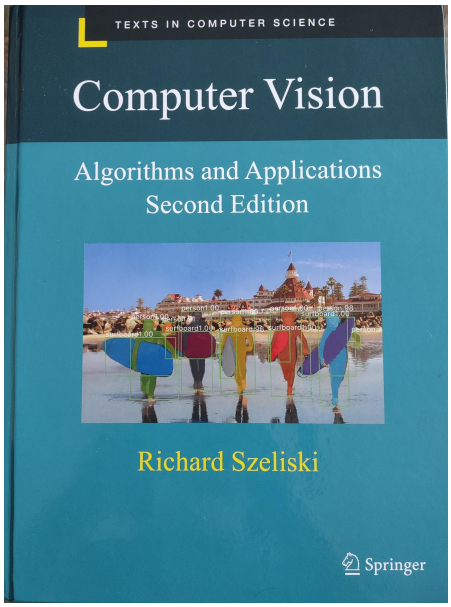
2. **Image Positioning (Step 1):**
   - Find the correct position of each image in the 3x3 grid using keypoint matching techniques. You may use any feature detection and matching algorithms, such as **SIFT** (Scale-Invariant Feature Transform) or **ORB** (Oriented FAST and Rotated BRIEF).
   - Match keypoints between images to align them correctly.




3. **Image Stitching (Step 2):**
   - Use any stitching method to combine the 9 images into one, but it is strongly recommended to use the same stitching method implemented in the mini-project.
   - You may start stitching from any image, and the stitching should account for potential image overlaps.

4. **Post-Processing (Step 3):**
   - After stitching, crop any unnecessary areas (e.g., black borders) to make the final image as clean and accurate as possible.

5. **Saving the Final Image:**
   - Save the final stitched image as [input_folder_name].jpg.
   - Here is the expected output (ideally)




**Documentation:**

- Provide a README.md file that includes:
    - A description of your approach and solution.
    - Instructions on how to run the code.
    - A brief explanation of the keypoint matching algorithm and stitching method used.

**Submission:**

Submit the following files as a group on Canvas:

1. tile-quiz.py (your main Python file containing the solution).
2. README.md (detailed documentation of your solution and execution).
3. [Option] Any additional files required to run the code (if applicable).
4. [Option] If your group uses GitHub, provide the repository URL for reference. However, the repository will not be used for grading, only as a reference.

**Datasets:**

- Three datasets are provided, each containing a folder with 4 images arranged in a 2x2 grid.
    - “data-1” : CS textbook cover 1
    - “data-2” : magazine cover 2
    - “data-3” : book cover 3
- If you need more dataset, you are encouraged to create your own dataset by taking images in a 2x2 grid pattern. However, this is not required.
- The instructor may test your code with additional, unseen datasets.

**Grading Criteria (Total 70 Points):**

1. **Correct Image Positioning (20 points):**
   - Use of feature matching algorithms (SIFT, ORB, etc.) to correctly align the images.

2. **Stitching Quality (20 points):**
   - Correct implementation of image stitching, producing a seamless final image.

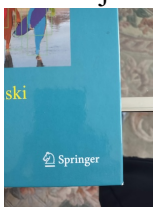
3. **Post-Processing (5 points):**
   - Removing unnecessary borders and ensuring the final image looks polished.

4. **Code Quality (10 points):**
   - Clean, readable, and well-documented code.

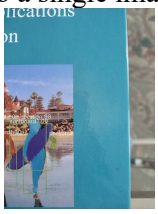
5. **README Documentation (5 points):**
   - Clear explanation of the solution and how to execute it.

6. **Final Overall Image Quality (10 points):**
   - Quality of the final stitched image, considering overlaps, cropping, and alignment.

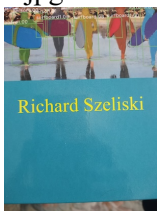
7. **Extra work (10 points):** If you have a time, you can solve the following problem to get extra points.
   - You are going to extend your solution to stitches 9 images taken in a 3 x 3 grid format of the same object into a single image.



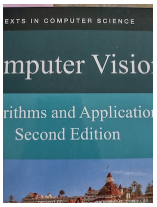
1.jpg


2.jpg


3.jpg

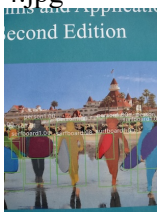


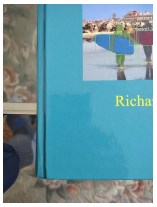
4.jpg



5.jpg


6.jpg


7.jpg


8.jpg


9.jpg

   - Datasets:
     - “extra-1”: 9 images for CS textbook cover
     - “extra-2”: 9 images for ACM magazine cover
   - You can create a new solution python file, i.e., tile-quiz-extra.py and submit it with tile-quiz.py file on Canvas. No extra time will be given.