Image Processing - Exercise 4

Hadar Tal, hadar.tal, 207992728

Introduction

The goal of the exercise is to blend two images of different resolutions seamlessly, creating a single composite image where the high-resolution details are preserved while incorporating the overall structure of the low-resolution image. The main techniques employed to achieve this include image pyramid construction, feature matching using the Scale-Invariant Feature Transform (SIFT) algorithm, homography estimation using RANSAC, and image blending using Laplacian pyramids. Overall, these techniques enable the creation of a visually pleasing composite image that seamlessly integrates the content of the input images at different resolutions.

Algorithm

Image Preprocessing:
Initially, the algorithm begins by preprocessing the input images. This involves opening the images and converting them to grayscale to simplify subsequent processing steps.

Pyramid Construction:
The algorithm constructs Gaussian and Laplacian image pyramids for both the high-resolution and low-resolution images. To optimize computational efficiency and maintain robust feature detection, Feature Extraction and Feature Matching are executed on the second level of the Gaussian pyramid. This approach leverages the relatively smaller difference in resolution at this level, ensuring a balance between preserving key details and reducing computational complexity, thereby enhancing the overall performance of the algorithm.

Feature Extraction (SIFT):
Inputs: Grayscale images.
Outputs: Detected keypoints and computed descriptors for both images.
SIFT Detection: Utilizes the Scale-Invariant Feature Transform (SIFT) algorithm to detect keypoints in both images. These keypoints represent distinctive image features invariant to scale, rotation, and illumination changes.
SIFT Descriptor Computation: Computes descriptors for each detected keypoint to characterize its local neighborhood. These descriptors encode information about the keypoints' surrounding texture and gradient orientations.

Feature Matching:
Once keypoints and descriptors are computed, the algorithm matches them between the high-resolution and low-resolution images to identify corresponding keypoints.
Inputs: Detected keypoints and computed descriptors for both images.
Outputs: Matched keypoints between the high-resolution and low-resolution images.

Homography Estimation:
Based on the matched keypoints, the algorithm estimates a homography matrix using the Random Sample Consensus (RANSAC) algorithm. This homography matrix aligns the high-resolution image with the low-resolution image.
Inputs: Matched keypoints between the high-resolution and low-resolution images.
Outputs: Homography matrix representing the geometric transformation.

Mask Generation:
The algorithm generates a binary mask to delineate the regions of interest in both images. This mask is derived from the transformed high-resolution image.
Inputs: Transformed high-resolution image.
Outputs: Binary mask delineating regions of interest in both images.

Image Blending:
Finally, the Laplacian pyramids of both images are blended according to the generated mask. This blending process combines the details from the high-resolution image with the overall structure of the low-resolution image, resulting in a seamless composite image. While a simple approach could involve directly pasting the transformed high-resolution image over the low-resolution one, blending via Laplacian pyramids produces a more visually pleasing result by smoothly integrating the details and structure of both images. Inputs to this process include the Laplacian pyramids of both images and the generated mask, while the output is the blended composite image.

Implementation Details

The implementation of the algorithm utilizes the OpenCV library for various image processing tasks, including reading images, color conversions, feature detection, and matching. Initially, Gaussian and Laplacian image pyramids are constructed for both the high-resolution and low-resolution images through iterative Gaussian blurring and downsampling. Feature extraction is performed using the Scale-Invariant Feature Transform (SIFT) algorithm to detect key points and compute descriptors representing local image features. These descriptors are then matched using a brute-force matcher with a ratio test to find correspondences between features in the two images. Subsequently, the matched keypoints are used to estimate a homography transformation between the images.

The implementation utilizes the OpenCV library extensively for image processing tasks, including reading images (cv2.imread), color space conversions (cv2.cvtColor), Gaussian blur (cv2.GaussianBlur), feature detection and description (cv2.SIFT_create), keypoint matching (cv2.BFMatcher), and homography estimation (cv2.findHomography).
Prior to utilizing OpenCV functionalities, I independently implemented a comprehensive set of image processing algorithms, including a multi-scale Harris corner detector for keypoint detection, a custom descriptor generation method using backward warping windows, and a matching algorithm based on Euclidean distances of wavelet transforms. Additionally, I implemented RANSAC for robust estimation of geometric transformations and image back-warping for aligning images. While this custom implementation was functional, it exhibited longer runtime and required manual parameter tuning for each image, leading to decreased efficiency and adaptability compared to OpenCV's optimized functions. Consequently, I transitioned to utilizing OpenCV's library, which provided more efficient and robust implementations of these algorithms, enhancing overall performance and ease of use. After the “ex4” function, I submitted the independent implementations of the custom image processing algorithms.

Within the algorithm, numerous hyperparameters and thresholds play crucial roles in the image processing pipeline.
Scale for Feature Matching (scale_for_feature_matching): This parameter specifies the scale at which the algorithm performs feature matching across the image pyramids. Opting for a larger value can speed up the computation process, albeit at the risk of overlooking more subtle details. Conversely, selecting a smaller value can amplify the impact of resolution differences, potentially resulting in the generation of incorrect matches later in the process.
Gaussian Kernel Size (gaussian_kernel_size): This parameter defines the size of the Gaussian kernel used for image blurring operations. A smaller kernel size will result in less smoothing and preserve finer details but may also introduce noise, while a larger kernel size will produce stronger smoothing and reduce noise at the cost of potentially blurring important features.
Ratio Threshold for Feature Matching (ratio_1nn_to_2nn): This threshold determines the ratio between the distances of the first and second nearest neighbors in feature matching. A smaller value will result in more stringent filtering of matches, potentially removing outliers but also risking the exclusion of valid matches. Conversely, a larger value may include more matches but also increase the likelihood of incorrect matches.
Maximum Error Threshold for Homography Estimation (max_error_for_homography): This threshold specifies the maximum allowable error for estimating the homography matrix using the RANSAC algorithm. A smaller value will result in a more strict selection of inliers, leading to a more accurate homography but potentially discarding valid correspondences. On the other hand, a larger value may include more points in

the homography estimation process but could also introduce more outliers and reduce the accuracy of the transformation.

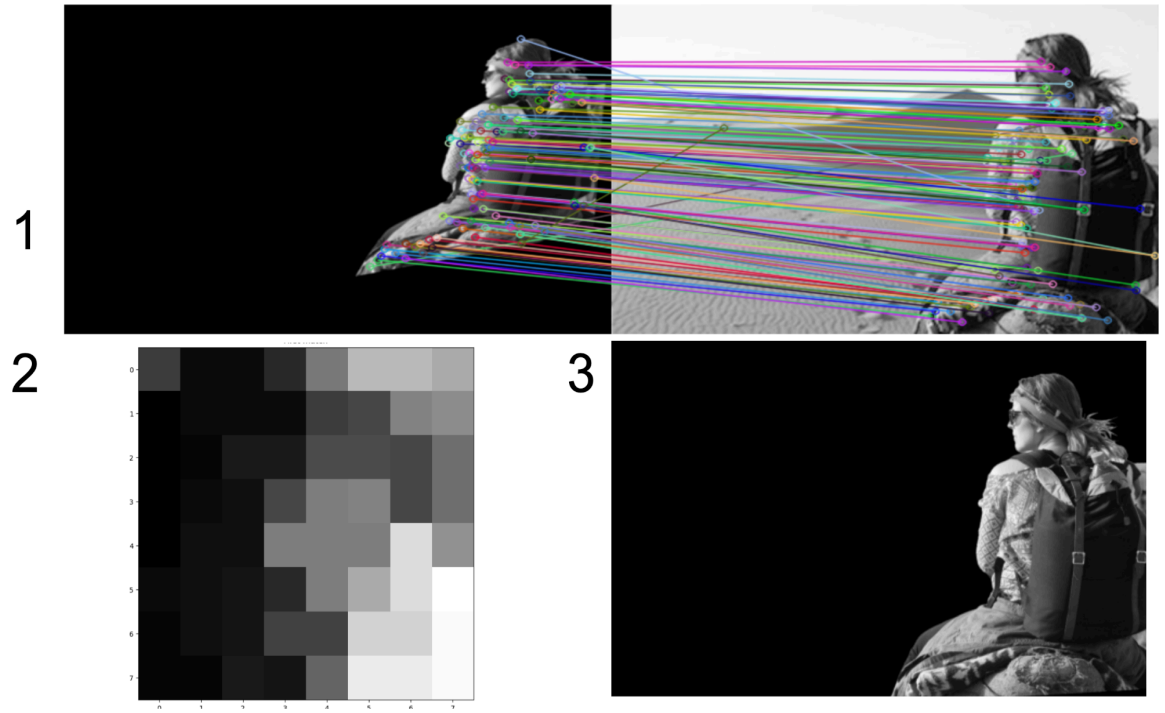
During implementation, I encountered several challenges, making this task quite demanding. One significant challenge was my decision not to rely solely on existing packages, opting instead to implement many functionalities from scratch. This approach required a considerable amount of time and effort. However, I'm pleased with this decision as it provided me with valuable learning opportunities. Despite eventually integrating some package-based solutions into the final algorithm, I gained a deeper appreciation for the functions we often take for granted in editing software. For instance, creating MOPS descriptors at gradient angles proved challenging, highlighting the complexity underlying seemingly simple operations like image rotation. This experience underscored the intricacies involved in image processing tasks and enhanced my understanding of the underlying concepts.

Visual Results

These visual results represent the output for the two sets of images. One flaw observed in the final results is the presence of thick black lines around the high-resolution image after applying the homography transformation. This artifact occurred due to boundary effects during the warping process, where pixels outside the original image boundaries were set to zero, resulting in the black lines.

In visualization (1) below, each image represents one of the input images. The colored circles denote the detected keypoints, with different colors indicating different keypoints. The lines connecting keypoints between the two images represent matched keypoints. The majority of these lines form a consistent pattern or structure across the images, indicating successful matching. However, there are also some outliers where the lines connect keypoints that do not correspond well between the images. These outliers are represented by scattered or irregular lines across the visualization. Overall, this visualization provides insight into the quality of the keypoint matching process, showcasing both successful matches and potential outliers.

I also provide an example of a MOPS (2) (Maximally Stable Oriented Pyramids) descriptor, which consists of an 8x8 image representing the local features extracted from the Gaussian pyramid at two scales above, capturing the corners' area by considering the angle of the gradient.
In Figure 3, the outcome of applying the homography to the high-resolution image is illustrated.



Conclusion

The project also underscored the beauty and power of applying mathematical concepts to image processing tasks. Techniques such as linear transformations and homography played a crucial role in aligning images and extracting meaningful information. Moreover, understanding the gradients within images allowed for the detection of key features, while concepts like Fourier analysis enabled the exploration of frequency components. By delving into these mathematical principles, I gained a deeper appreciation for the intricate relationship between mathematical concepts and their application in image analysis and manipulation.
Additionally, the project highlighted the immense power and efficiency offered by libraries like OpenCV in handling complex image processing tasks. While I initially attempted to implement certain algorithms from scratch for learning purposes, the eventual reliance on OpenCV demonstrated the significant time-saving benefits and robust functionality provided by established libraries. This experience underscored the importance of striking a balance between learning fundamental concepts through manual implementation and leveraging the capabilities of existing libraries to streamline development and achieve practical results more effectively.