Computer Vision Final Project

“Image Patching for Corrupted Image Restoration “

İsmail Deha Köse

ICT for Internet and Multimedia

Introduction:

The aim of this project is to restore a corrupted image by overlaying patches obtained from reference images. The code utilizes computer vision techniques, such as keypoint detection, feature matching, and homography estimation, to align and blend the patches with the corrupted image. The report provides an overview of the project details, how the program works, and its potential applications.

Project Details:

The program uses the following libraries and tools:

- OpenCV: an open-source computer vision library that provides a wide range of tools for image and video processing.
- C++: a general-purpose programming language that is widely used for developing high-performance applications.

The program consists of the following functions:

- `getNumPatches`: Prompts the user to enter the number of patches and returns the value.
- `loadImages`: Loads the corrupted image and the patches into `cv::Mat` objects and stores them in a `std::vector`.
- `convertToGrayscale`: Converts the image and patches to grayscale.
- `detectAndCompute`: Detects keypoints and computes descriptors for the image and patches using the SIFT algorithm.
- `computeMatches`: Computes matches between the image and patches using the BFMatcher class provided by OpenCV.
- `refineMatches`: Refines the matches by selecting the matches with distance less than ratio * min_distance.
- `findHomography`: Finds the homography matrix using RANSAC.
- `overlayPatches`: Overlays the patches over the image to fix the corrupted regions.
- `displayResult`: Displays the result.

Step 1: Loading the Corrupted Image and Patches The algorithm begins by loading the corrupted image and the corresponding patches into memory using the OpenCV library.

Step 2: Converting to Grayscale To simplify the subsequent operations, both the image and patches are converted to grayscale using the `cv::cvtColor` function.

Step 3: Keypoint Detection and Descriptor Computation the Scale-Invariant Feature Transform (SIFT) algorithm is employed to detect keypoints and compute descriptors for both the grayscale image and patches. This is accomplished using the `cv::SIFT::create()` function.

Step 4: Match Computation A brute-force matcher based on the Euclidean distance is used to compute matches between the descriptors of the patches and the image. The `cv::BFMatcher` is utilized for this purpose.

Step 5: Match Refinement To improve the quality of matches, a threshold-based selection process is performed. Only matches with distances below a certain ratio of the best match are considered as good matches.

Step 6: Homography Estimation Using the RANSAC algorithm, homographies are estimated between the keypoints of the patches and the keypoints of the image. The `cv::findHomography` function is employed for this step.

Step 7: Overlaying the Patches The patches are overlaid onto the image by warping them according to the estimated homographies. This process is accomplished using the `cv::warpPerspective` function.

Step 8: Displaying the Result The final result, which includes the repaired image with the patches overlaid, is displayed to the user.

How It Works:

The code begins by prompting the user to enter the number of patches to be used for restoration. Then, it loads the corrupted image and the patches from the specified file paths. After converting them to grayscale, it applies the SIFT algorithm to detect keypoints and compute descriptors for both the image and the patches.

Next, the code performs feature matching between the image and the patches using a brute-force matcher. The matches are refined by applying a distance ratio test to filter out the unreliable matches. The refined matches are used to estimate the homography matrices for each patch using the RANSAC algorithm.

Finally, the patches are overlaid on the corrupted image by warping each patch according to its corresponding homography matrix. The patches are blended with the image, replacing the corrupted regions. The resulting restored image is displayed to the user.

Conclusion:

In this project, we successfully implemented an image patching algorithm using homography estimation. The algorithm effectively repaired corrupted regions in an image by overlaying relevant patches. By leveraging the Scale-Invariant Feature Transform (SIFT) for keypoint detection and descriptor computation, along with refining matches and estimating homographies, the algorithm accurately aligned the patches with the corrupted regions..The program can also be extended to work with images that have been subject to transformations, such as the "_t" patches in this Project, and also new and different algoritms.

References:

- OpenCV documentation: [https://docs.opencv.org/4.x/]