Giorgio Mendoza
CS549-E24-E01
Dr. Alexandros Lioulemes

Week 7 Lab

Part 1:

Objective:

The objective for part one was to render a triangular pyramid to illustrate the concept of rendering a 3D object onto a 2D image.

Code Description:

The script starts by defining a hardcoded camera matrix and distortion coefficients from the previous assignment for simplicity. These are critical for accurately projecting 3D coordinates onto a 2D image based on the specific camera's intrinsic properties.

The vertices of the triangular pyramid are defined in a 3D space and this includes the corners of the base and the apex of the pyramid. The script also defines points to represent the axes originating from the base of the pyramid, helping to visualize the orientation of the 3D object in the image.

After loading the image and converting it to grayscale, the cv.findChessboardCorners API can be used to detect the corners of the chessboard in the image. These corners are refined using cv.cornerSubPix to increase their accuracy, useful for precise pose estimation.

The cv.solvePnP API is employed to compute the rotation and translation vectors that map the 3D points onto the 2D image. This function is important since it determines how the 3D model should be oriented and positioned relative to the camera.

The script uses the cv.SOLVEPNP_P3P algorithm, which is used for cases where exactly four point correspondences are provided. Then the 3D points of the pyramid and the axes are projected onto the 2D plane of the image using cv.projectPoints, which utilizes the previously calculated pose.

As shown below, a draw function was used to visualize the pyramid in both pictures. However, the second picture includes an axis to get a better understanding of the orientation of the pyramid with respect to the chessboard to get a better visual understanding (red for X, green for Y, and blue for Z).

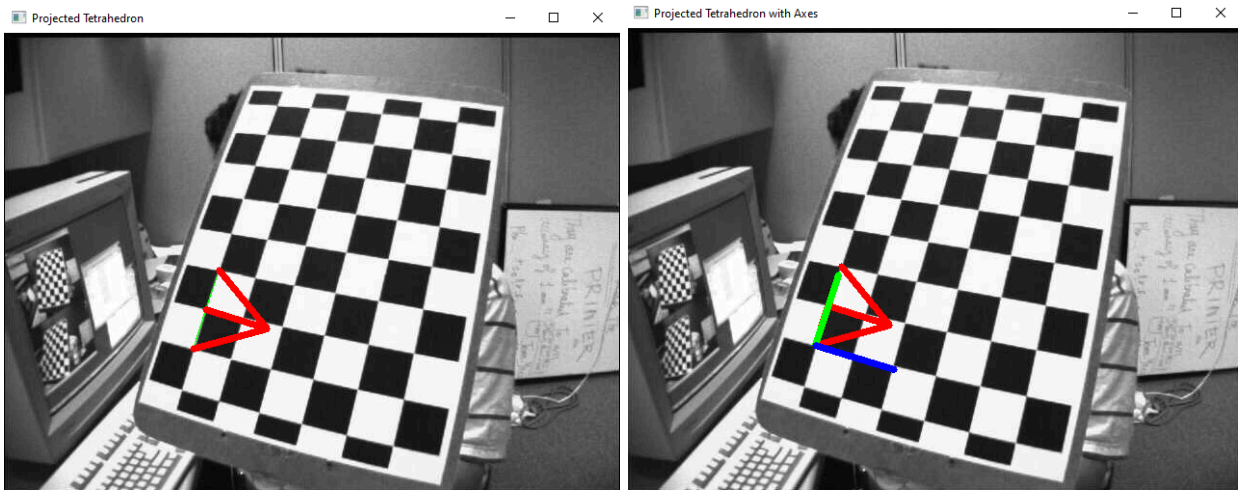

Figure 1. Triangular Pyramid Render on Chessboard

Part 2:

Objective:

Part two consists of generating a depth map using the two stereo images shown below.

Figure 2. Reference Stereo Images

Code Description:

In the second part of the project, a process was developed to create a disparity map from stereo images, which is pivotal in computer vision for deriving depth information from two-dimensional images. The method involves several key steps implemented in Python using OpenCV.

The process began with loading the left and right stereo images in grayscale format for easier processing. Utilizing OpenCV's StereoBM (Block Matching) class, a stereo block matching algorithm was applied to these images. This algorithm compares blocks of pixels between the two images to find matching blocks, which indicate similar features at different horizontal positions. The algorithm was configured with specific parameters, including numDisparities set to 160 to define the number of disparity levels considered, and blockSize set to 27, affecting the granularity of the matching process. Additional settings like texture threshold and uniqueness ratio were adjusted to optimize the algorithm's accuracy.

Following the computation, the disparity map underwent post-processing to enhance its quality. This included applying a median blur to reduce noise and smooth out the results, and speckle filtering to remove small noise points that do not conform to the larger area's disparity characteristics. For better visualization, the disparity map was normalized to span the full range of displayable grayscale values, enhancing the contrast and making the depth differences more apparent.

Finally, the normalized disparity map was displayed using Matplotlib with a color bar added to indicate the range of disparity values, providing a visual interpretation of depth information.

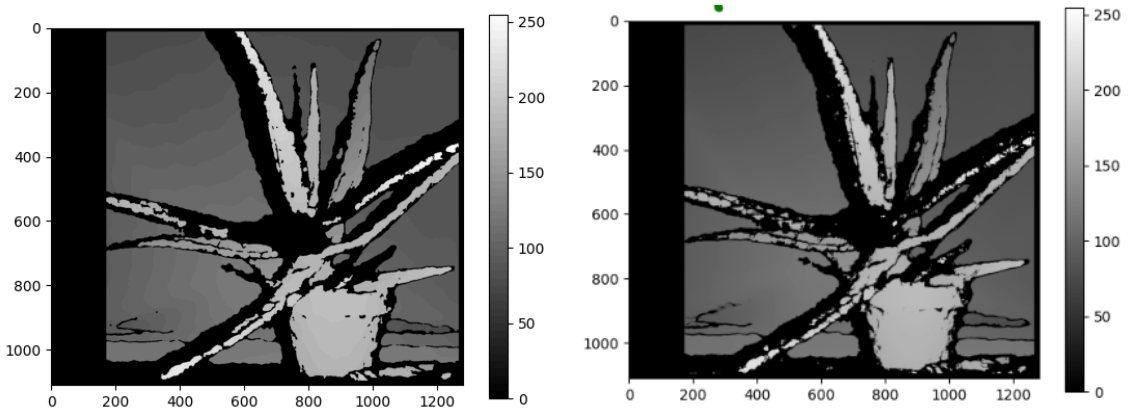

Figure 3. Best Results

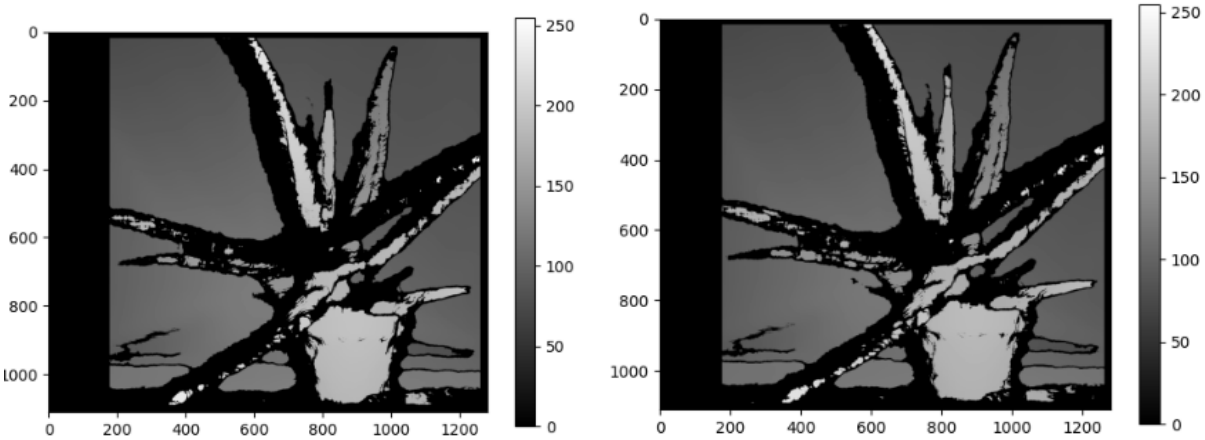

Figure 4. Results w/o using texture threshold, uniqueness ratio or median blur

The best results were achieved by fine-tuning the texture threshold, uniqueness ratio, and median blur. The texture threshold identifies areas with sufficient texture for reliable matching, while the uniqueness ratio ensures the selected match is distinctly better than the second-best. A median blur applied post-calculation smooths the disparity map, removing minor noise without blurring object edges. These adjustments, along with optimal disparities and block size, resulted in a cleaner and more uniform depth image compared to others, showcasing the effectiveness of these settings in enhancing disparity map quality.

Part 3:

Objective:

Part 3 of the project involved analyzing epipolar geometry by using the images below. The task focused on the respective pair of images to determine their epipoles and epipolar lines. The process included feature detection, matching across images, and computing the fundamental matrix, essential for defining epipolar lines and epipoles. The selection of widely distributed correspondence points across different continents was crucial for a robust fundamental matrix.

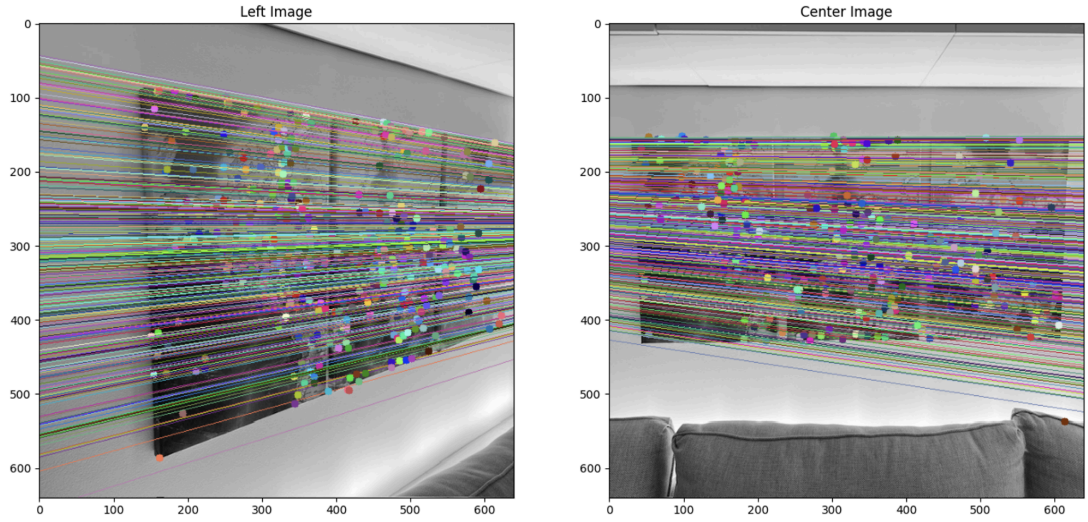

Figure 5. Left & Center Reference Images

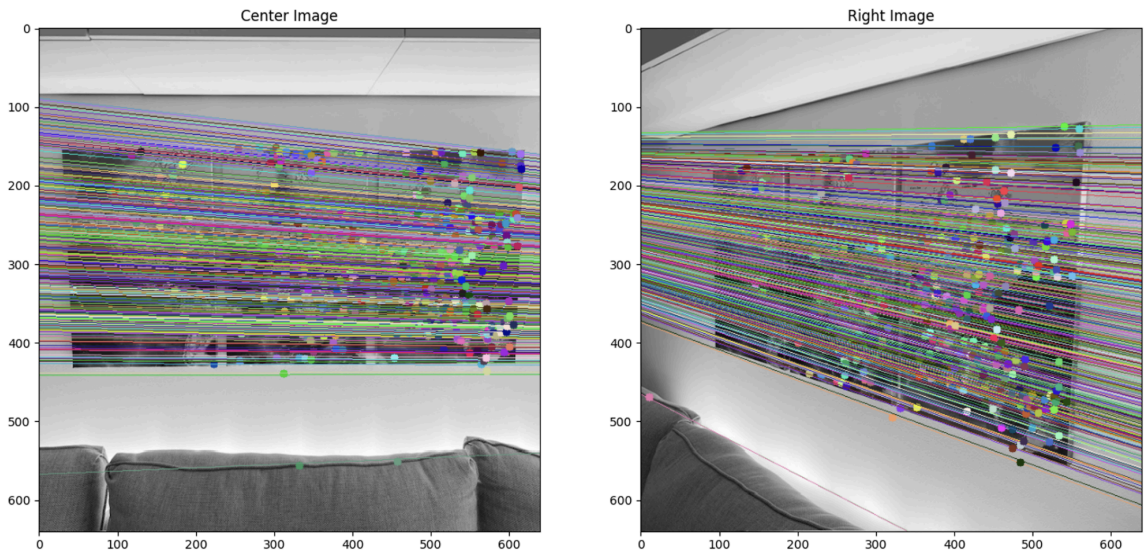

Figure 6. Center & Right Reference Images

Code Description:

For this part, the process begins by loading pairs of images in grayscale, which simplifies the detection of features by focusing solely on intensity variations. A SIFT detector is then initialized to identify keypoints and compute descriptors for these keypoints in both images of each pair. These keypoints represent distinctive features in the images that can be matched across the two views.

After detecting the keypoints, a FLANN-based matcher, which uses a KD-Tree, is employed to efficiently find the best matches between the two sets of descriptors. Lowe's ratio test is applied to these matches to ensure that only the most reliable matches (those significantly better than the second-best match) are retained.

With these filtered matches, the fundamental matrix is computed using the cv.findFundamentalMat function. This matrix is important as it encapsulates the epipolar geometry, determining how points in one image relate to lines (epipolar lines) in the other image.

Using the fundamental matrix, the script calculates and draws epipolar lines on each image of the pair. This is done by projecting points from one image onto the other as lines using the cv.computeCorrespondEpilines function, which computes the corresponding epipolar lines for a set of points in an image given the fundamental matrix.

The results are displayed with the drawn epipolar lines, which visually represent the intrinsic geometric relationships between the two views. The process is repeated for both image pairs to demonstrate the consistency and reliability of the epipolar geometry derived from the fundamental matrix.

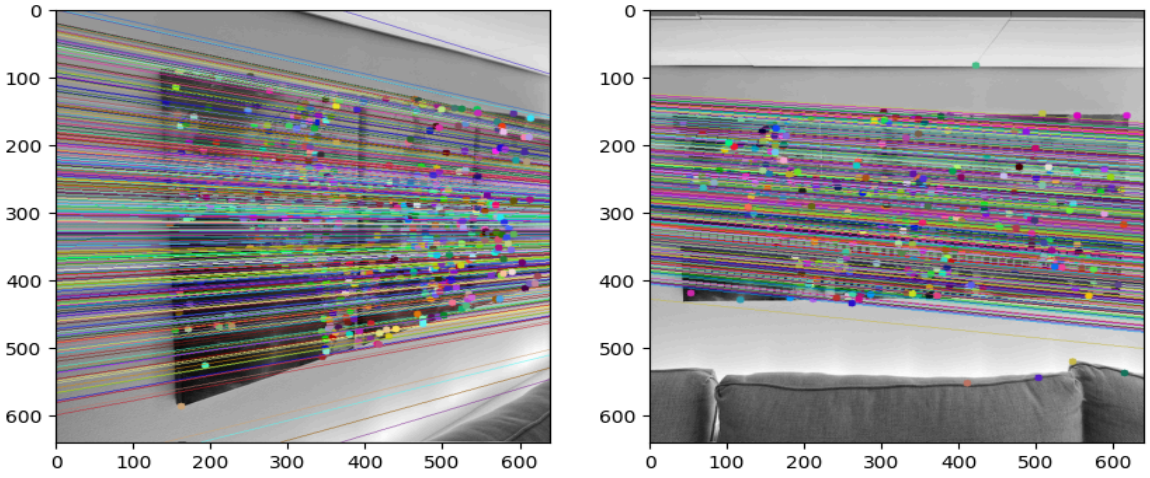

Figure 7. Left & Center m.distance < 0.97, search_params = dict(checks=55)

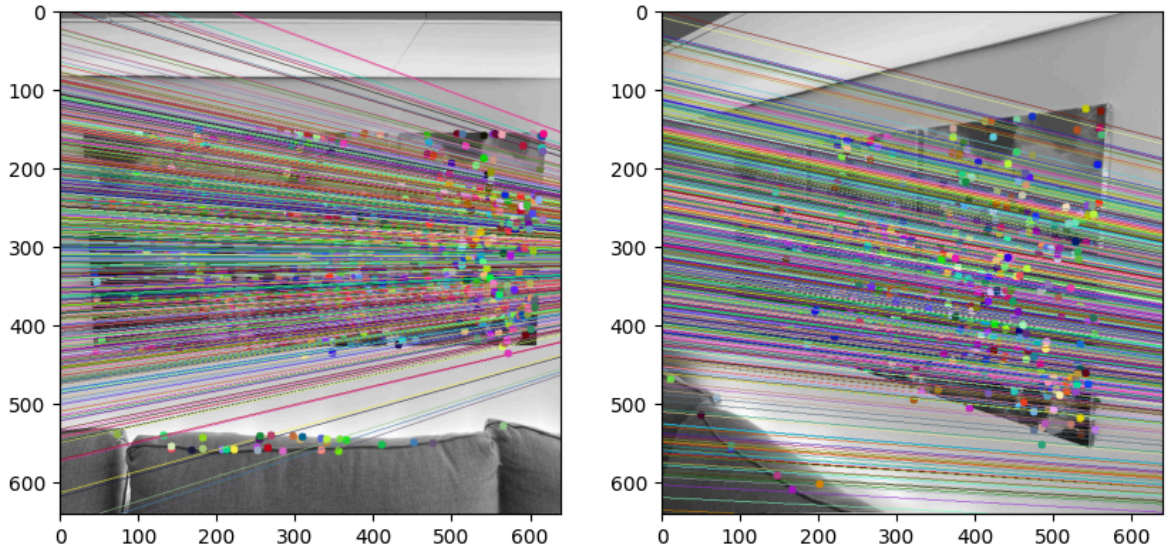

Figure 8. Center & Right Reference Images m.distance < 0.97, search_params = dict(checks=55)

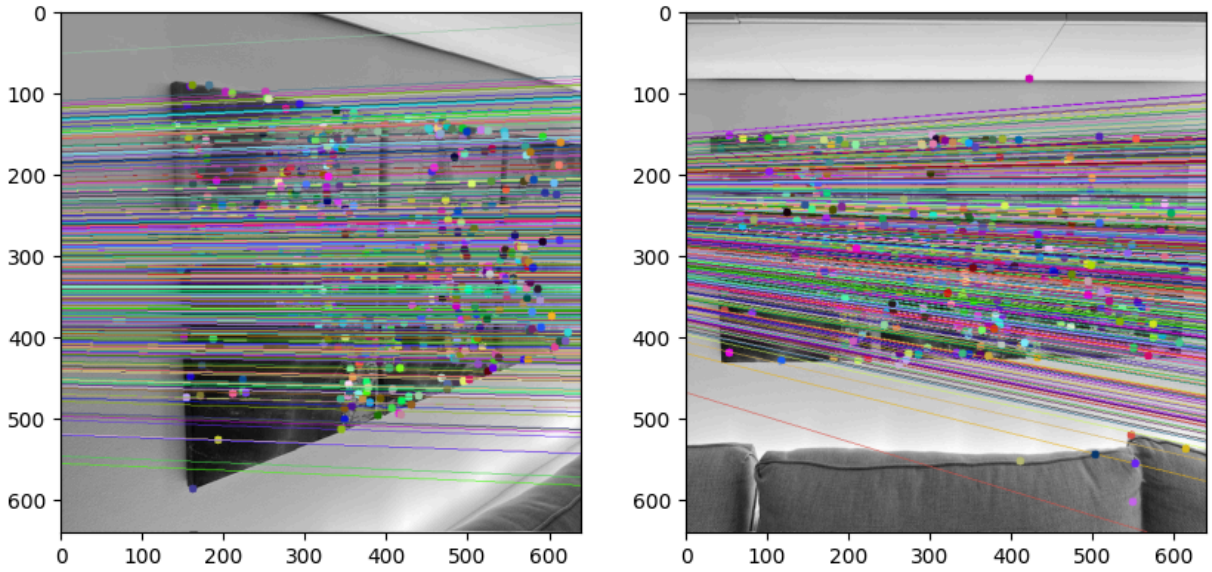

Figure 9. Left & Center m.distance < 0.97, search_params = dict(checks=70)

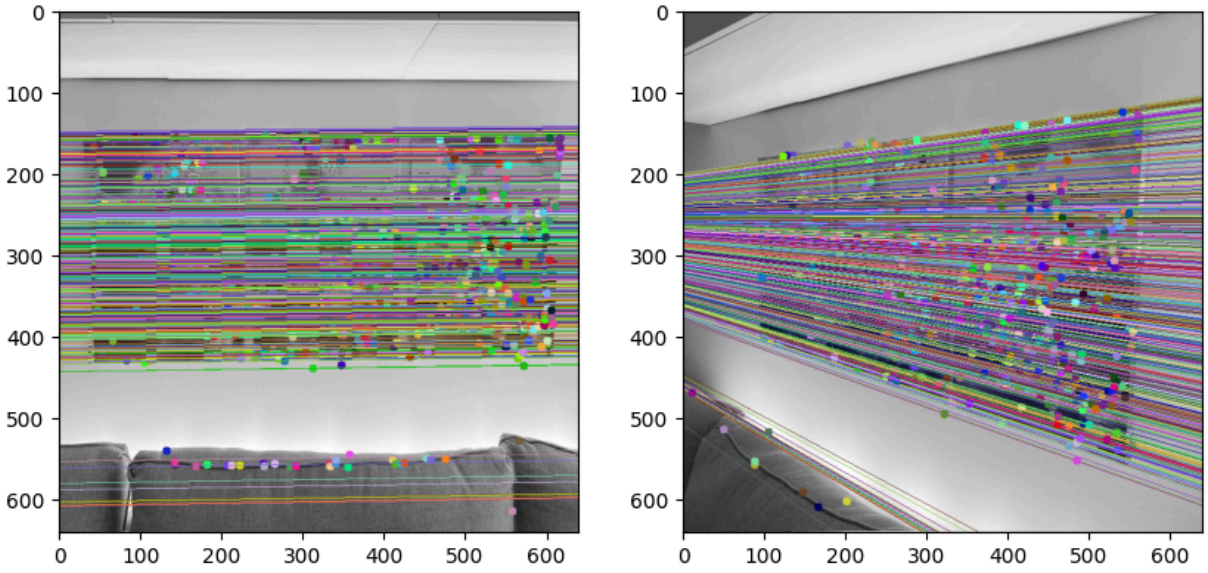

Figure 10. Center & Right m.distance < 0.97, search_params = dict(checks=70)

The results presented are intriguing, as both pairs of images utilized the same m.distance < 0.97 parameter, suggesting a standardized approach to match quality that focuses on ensuring only the most significant matches are considered. This strategy helps reduce false positives, capturing potential matches necessary for less distinct features.

However, both pairs required a different value for the checks parameter, highlighting variations in image complexity or texture:

- **Left & Center Pair (checks=55)**: The lower checks value for this pair suggests that these images either have more distinctive features or are more similar, allowing for accurate matches with fewer checks.
- **Center & Right Pair (checks=70)**: Conversely, the higher checks value indicates these images might be more challenging due to factors like less distinctive features or greater viewpoint differences. Increasing checks ensures a thorough search, improving the chances of finding correct matches.

Adjusting the checks parameter for each pair underscores the importance of customizing feature matching to specific image traits. This tailored approach ensures robust matching and accurate calculations of epipolar geometry, which are crucial for effective stereo vision analysis.