AIN433 - Report on Computer Vision Lab Assignment 3

Student Name: Emre Çoban
Student ID: 2200765028
Date: 18.12.2023
Email: b2200765028@cs.hacettepe.edu.tr

1 Introduction

The field of Computer Vision is characterized by techniques designed to empower machines in the interpretation and comprehension of visual information. Within the AIN 433 course, an immersive exploration of Panorama Image Stitching is undertaken in Programming Assignment 3. This application entails the seamless integration of a set of sub-images to create a cohesive and visually compelling panorama.

The primary objective of this assignment is to employ advanced computer vision techniques for the intricate task of panorama stitching. By utilizing keypoint descriptors, specifically the Scale-Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF (ORB), the aim is to have distinctive points in the images identified. These techniques are crucial for overcoming challenges such as viewpoint changes, scale differences, and variations in lighting within the dataset.

The comprehensive approach encompasses not only keypoint extraction but also matching features, homography calculation, and merging by transformation. Feature matching involves the association of keypoint descriptors across different images, establishing correspondences essential for understanding the scene's structure. Homography calculation is employed to model the geometric transformation between images, ensuring accurate alignment. Finally, merging by transformation involves the application of the calculated homography to seamlessly integrate sub-images into the final panoramic composition.

Originally intending to compare SIFT with SURF, licensing considerations led to the selection of ORB as an alternative method for comparison in this assignment. Subsequent sections will provide a detailed explanation of each code component, showcase visual results for panoramas in the dataset, and delve into insights related to code quality and performance.

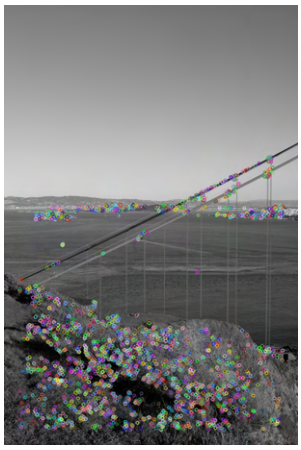
By completing this assignment, practical experience in the application of computer vision techniques is sought, with the ultimate goal of achieving the seamless creation of panoramic images from diverse sub-images.

1.1 Keypoint Extraction

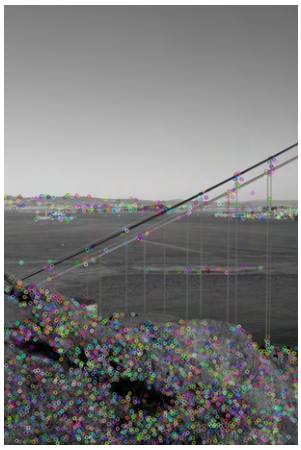
Keypoint extraction plays a pivotal role in Panorama Image Stitching, requiring the identification of distinctive points for subsequent matching and alignment. In this assignment, two renowned keypoint description methods, the Scale-Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF (ORB), are employed.

To offer flexibility in selecting the keypoint extraction method, a unified function has been implemented. This function enables the use of either SIFT or ORB based on the specified method parameter. It is worth noting that originally, the plan was to compare SIFT with another method, namely SURF. However, due to licensing considerations, ORB was chosen as an alternative method for comparison in this assignment.

The keypoints detected by SIFT and ORB for the same image are visualized in Figure 1, and a quantitative comparison is provided in Table 1.



(a) ORB Keypoints



(b) SIFT Keypoints

Figure 1: Comparison of ORB and SIFT Keypoints

Method	No. of Keypoints	Extraction Time	No. of Matches	Matching Time
SIFT	2735	0.0651	835	0.0220
ORB	4501	0.0100	556	0.0295

Table 1: Comparison of SIFT and ORB Keypoint Extraction and Matching

1.2 Scale-Invariant Feature Transform (SIFT)

SIFT stands out as a widely-utilized keypoint extraction method celebrated for its robustness in handling changes in scale, rotation, and illumination. The underlying mechanism involves identifying keypoint locations across various image scales and generating a descriptor for each keypoint that maintains invariance to scale and orientation changes. SIFT's effectiveness in scenarios with varying scale and rotation makes it particularly well-suited for applications demanding robust feature detection.

In the context of performance, SIFT is known for its computational intensity, and while it may be slower compared to some alternatives, its ability to provide reliable keypoints in challenging conditions justifies its application in scenarios requiring scale invariance.

Advantages of SIFT:

- Robust to changes in scale, rotation, and illumination.
- Provides distinctive and stable keypoints across diverse conditions.
- Well-suited for applications demanding robust feature detection.

Disadvantages of SIFT:

- Computationally intensive, resulting in slower processing times.

1.3 Oriented FAST and Rotated BRIEF (ORB)

Oriented FAST and Rotated BRIEF (ORB) is an algorithm crafted for efficient keypoint detection and descriptor extraction, especially tailored for real-time applications. The amalgamation of the high-speed feature detection prowess of the FAST (Features from Accelerated Segment Test) algorithm with the efficiency derived from the BRIEF (Binary Robust Independent Elementary Features) descriptor characterizes the core of ORB.

The efficiency exhibited by ORB originates from the utilization of binary descriptors and the FAST algorithm, which selectively identifies keypoint candidates with adeptness. The resultant synergy of these components manifests in a rapid and effective process for extracting features.

Advantages of ORB:

- Tailored computational efficiency for real-time applications.
- Exhibits rotation invariance and reasonable scale invariance.

Disadvantages of ORB:

- Binary descriptors may offer less distinctiveness compared to SIFT.
- Limited robustness in scenarios with significant viewpoint variations.

2 Feature Matching

After keypoints are extracted from different images using SIFT or ORB, the next step in Panorama Image Stitching involves feature matching. This process aims to establish correspondences between keypoints from different images, facilitating the subsequent steps of homography calculation and image stitching.

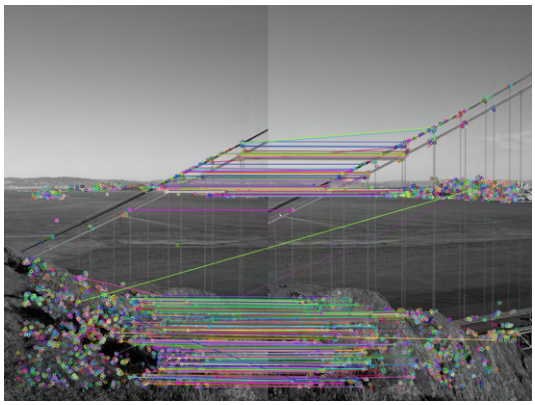
For feature matching, a brute-force matcher is employed, specifically the k-nearest neighbor method. In this method, each keypoint in the first image is matched with the k-nearest keypoints in the second image, and vice versa. The matches are then filtered based on a distance ratio test to retain only the high-quality correspondences.

The feature matching process is implemented by the following function, `match_keypoints`:

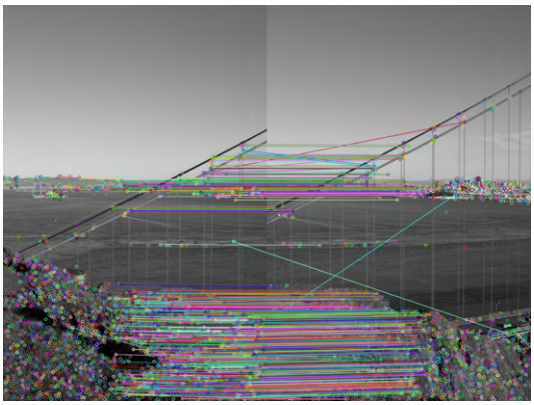
```
def match_keypoints(descriptors1, descriptors2)}
    # Create a Brute-Force Matcher
    bf = cv2.BFMatcher()

    # Use the matcher to find k=2 nearest matches for each descriptor
    matches = bf.knnMatch(descriptors1, descriptors2, k=2)

    # Filter good matches based on the distance ratio test
    good_matches = []
    for m, n in matches:
        if m.distance < 0.75 * n.distance:
            good_matches.append(m)

    # Return the list of good matches
    return good_matches
```

In this function, the Brute-Force Matcher is employed to identify the k=2 nearest matches for each descriptor. The matches are subsequently filtered based on the distance ratio test, allowing the retention of only those matches where the distance to the nearest neighbor is significantly smaller than the distance to the second nearest neighbor.

The feature matchings detected by SIFT and ORB for the same two images will now be visualized:



(a) Feature Matching with ORB Keypoints



(b) Feature Matching with SIFT Keypoints

Figure 2: Comparison of Feature matching with ORB and SIFT Keypoints

Decision to Use SIFT: Despite ORB's capability to discover more features in a shorter timeframe, the decision was taken to employ SIFT for the final generation of panoramic images. The primary consideration lay in SIFT's effectiveness in identifying a greater number of matches during the feature matching stage. This contributed to a more precise homography calculation, thereby elevating the quality of the final stitched panorama.

3 Homography Calculation

Homography is a transformative mapping between two images, capturing the geometric relationship between corresponding points. In computer vision and image stitching, the homography matrix is represented as the perspective transformation necessary to align and merge images seamlessly. It describes how one image should be warped to match the perspective of another, facilitating the creation of panoramic compositions.

The homography matrix is computed by the `calculate_homography` function, a custom implementation designed to utilize the RANSAC algorithm. Let's delve into the operational details of this function:

- **Random Sample Consensus (RANSAC):** RANSAC, a robust statistical method, is employed by the function to iteratively estimate the homography matrix. Subsets of correspondences between keypoints are randomly selected, candidate homography matrices are calculated, and the matrix that best aligns the keypoints is ultimately chosen.
- **Singular Value Decomposition (SVD):** Within each RANSAC iteration, Singular Value Decomposition is performed on a matrix constructed from source and destination points of randomly sampled matches. The homography matrix is extracted from the last column of the right singular vector using SVD.
- **Homography Normalization:** The calculated homography matrix is normalized by dividing all its elements by the value in the last row and last column. This normalization ensures that

the last element of the homography matrix is set to 1.

- **Inlier Counting:** For each candidate homography, the count of inliers is determined by comparing the transformed keypoints with the actual keypoints. The distance between the transformed and actual keypoints is measured using the Euclidean norm.
- **Updating the Best Homography:** The best homography is updated by the function if the number of inliers surpasses the current maximum. This process is repeated for the specified number of iterations to find the most robust homography matrix.

The application of this custom function is exemplified in the following code snippet:

```
homography_matrix, _ = calculate_homography(keypoints1, keypoints2,
                       matches, num_iterations=100, inlier_threshold=5.0)
```

The usage of the custom `calculate_homography` function is demonstrated in this line, where `keypoints1` and `keypoints2` represent keypoints from the first and second images, and `matches` denote the correspondences between them. The resulting `homography_matrix` is subsequently applied to align and stitch the images seamlessly, highlighting the effectiveness of the custom implementation.

4 Image Warping

The process of image stitching involves several key steps to ensure accurate alignment and seamless transitions between adjacent images. These steps are orchestrated by the `merge_images` function, as outlined below:

- **Corner Calculation:** The corners of rectangles for both input images are defined by the function. These corners, representing the extremities of each image, are crucial for determining the size of the resulting panorama.
- **Homography Transformation:** The corners of the second image are then subjected to the application of the homography matrix, calculated using the RANSAC algorithm. This transformation effectively warps the second image to match the perspective of the first image, ensuring proper alignment.
- **Bounding Box Calculation:** Based on the warped corners, the calculation of the minimum and maximum coordinates is performed to define a bounding box that encompasses both images.
- **Positive Quadrant Translation:** To ensure positive coordinates, the bounding box is translated to the positive quadrant. This translation is applied not only to the bounding box but also to the homography matrix.
- **Perspective Transformation:** The adjusted homography matrix is then used to warp the second image, resulting in a properly aligned version. This step is essential for achieving accurate alignment between images.
- **Image Placement:** The first image is placed on top of the warped second image at the appropriate location. This process involves considering the translation applied to the bounding box, contributing to the creation of a composite image.
- **Final Stitched Image:** The resulting stitched image is obtained, representing a seamless composition of the two input images. Any unnecessary border pixels are trimmed by the function to finalize the panorama, ensuring a visually appealing and coherent result.

This detailed image warping process ensures accurate alignment between adjacent images, contributing to the overall success of the image stitching algorithm.

5 Generating Panorama

The generation of a panorama involves a series of intricate steps, each contributing to the final seamless composition. The process encompasses feature extraction, keypoint matching, homography calculation, and image stitching. These steps are orchestrated by the `generate_panorama` function, ensuring a coherent and visually appealing result. Each stage is detailed below:

- **Feature Extraction** In the initial phase, keypoints and descriptors are extracted from each input image. Distinctive points in the images, along with their corresponding descriptors, are identified using the `extract_keypoints` function. This function supports both SIFT and ORB methods.
- **Keypoint Matching** The `match_keypoints` function utilizes a Brute-Force Matcher to find matches between the extracted keypoints of consecutive images. Reliable matches, filtered through the distance ratio test, form the basis for subsequent alignment.
- **Homography Calculation** The `calculate_homography` function, employing RANSAC, is utilized to calculate the homography matrix using corresponding keypoints. This step robustly estimates the geometric transformation needed to align the images.
- **Image Warping and Blending** The `merge_images` function employs the calculated homography to warp the second image and stitch it seamlessly with the first. Detailed steps include corner calculation, homography transformation, bounding box calculation, positive quadrant translation, perspective transformation, image placement, and final trimming for a visually appealing result. Intermediate step figures, which show keypoints and feature matches, are saved inside a folder called "intermediate_results."
- **Iterative Stitching** The iterative stitching process unfolds as images are processed one by one, progressively building the panorama. The homography calculation and image stitching are iteratively applied to ensure proper alignment between consecutive images.
- **Final Panorama** The final panorama is generated by iteratively stitching images, resulting in a visually coherent composition. The `generate_panorama` function automates these steps, producing a panorama that captures the entire scene seamlessly.
- **Rectifying Image** To address potential distortions, particularly the stretching of the leftmost images in the panorama as new images are added to the right, a rectification process is applied. The `detect_corner_points` function is used for the detection of corner points, followed by image rectification using `rectify_image`. These steps contribute to an improved final panorama, mitigating the issue of excessive stretching in the leftmost images and resulting in a more visually balanced composition. The code saves this new image as "rectified_result.png."

The integration of these stages in the image stitching algorithm contributes to the successful generation of panoramic compositions, providing a comprehensive view of the input images in a single, cohesive panorama. However, it's important to note that the leftmost images in the panorama might undergo stretching as new images are added to the right. This stretching occurs due to the perspective transformation applied during the homography calculation, which adjusts the images to create a seamless blend. As more images are added, the panorama dynamically expands, resulting in a larger and more detailed composition.

It's noteworthy that the algorithm is designed to work optimally with 5 to 9 images, depending on dataset characteristics. The image size may become too large after some steps, explaining why the

goldengate folder was completed with all images processed, while 5 images from the hotel, 4 images from the yard, 9 images from the carmel folder, and 7 images from the fishbowl folder were successfully handled.

6 Results

In the following section, intermediate results are presented, illustrating the key steps of the panorama generation process. Each set of images showcases keypoints, feature matches, and the progressively stitched panorama. It is important to note that limitations are encountered when handling larger datasets, typically beyond 6-7 images, due to increased size and complexity. This limitation is evident when our results are compared to ground truth panoramas.

Despite these challenges, proficiency is demonstrated in accurately merging images from their correct spatial positions, showcasing a robust understanding of the scene geometry. The process involves identifying keypoints, establishing feature matches, and calculating homography to seamlessly stitch the images together. However, as the dataset size increases, limitations arise, impacting the ability to generate panoramas comparable to ground truth.

The subsequent section features the final panoramic images generated by our method, accompanied by their corresponding rectified versions. While challenges may be faced with larger datasets, the presented results emphasize the method's proficiency in capturing and merging scenes accurately from the available input images.

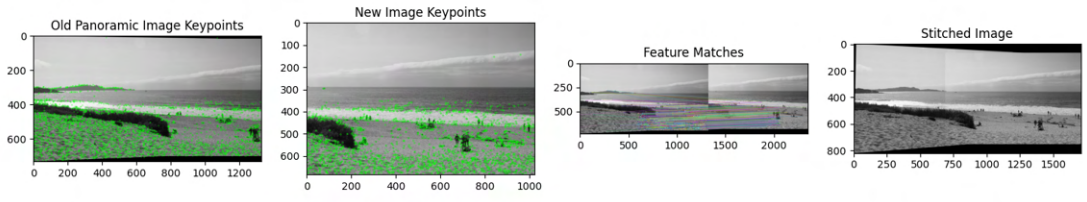


Figure 3: Intermediate Results of Carmel Folder Step 2 , Merging 3rd Image

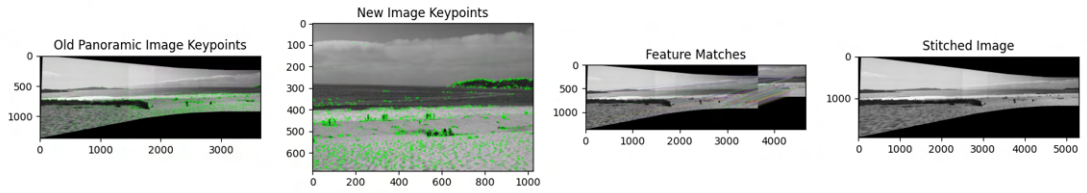


Figure 4: Intermediate Results of Carmel Folder Step 6, Merging 7th Image

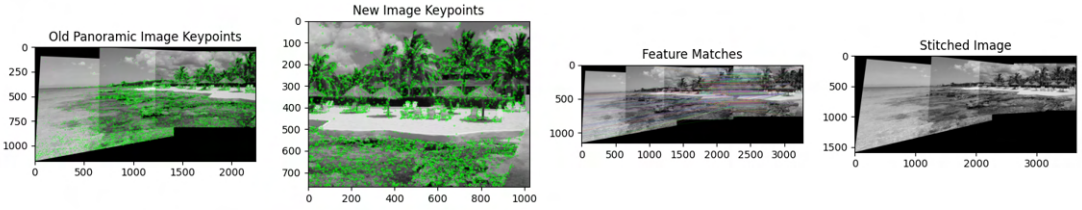


Figure 5: Intermediate Results of Hotel Folder Step 3, Merging 4th Image

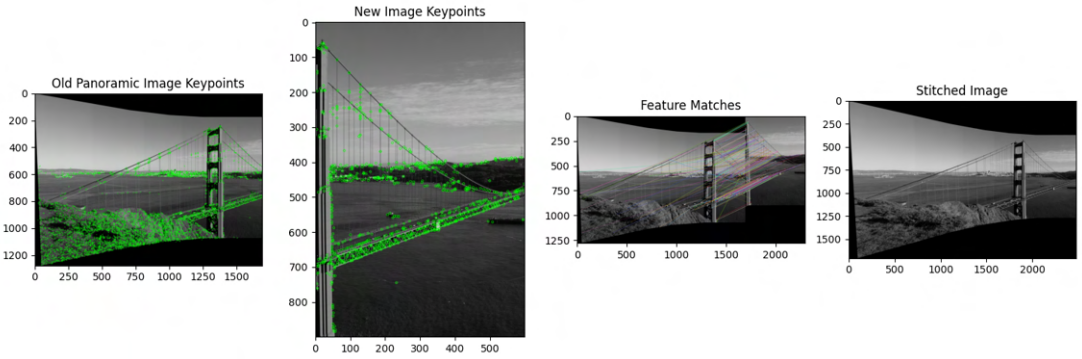
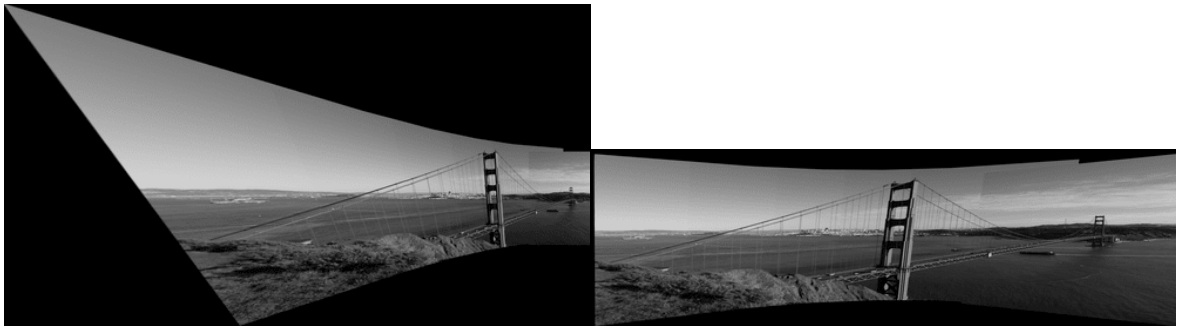
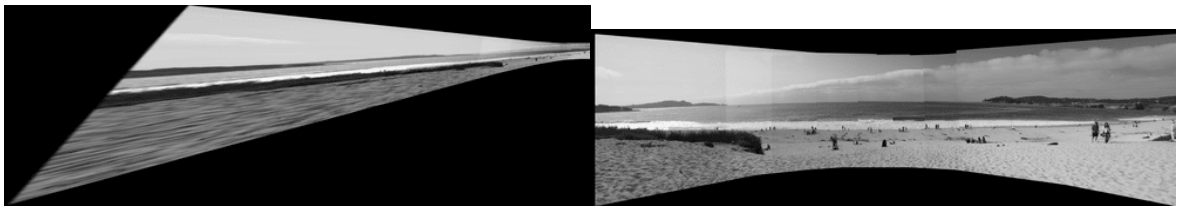


Figure 6: Intermediate Results of Goldengate Folder Step 4, Merging 5th Image



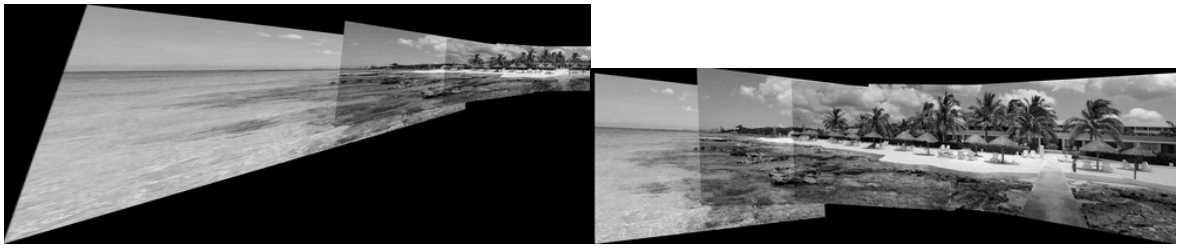
(a) Before Rectifying (b) After Rectifying

Figure 7: Panoramic Image of Goldengate Folder



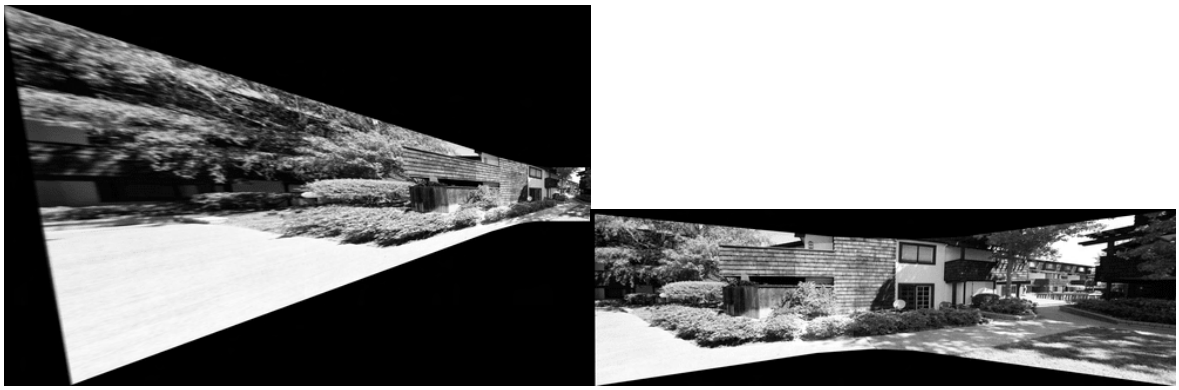
(a) Before Rectifying (b) After Rectifying

Figure 8: Panoramic Image of Carmel Folder



(a) Before Rectifying (b) After Rectifying

Figure 9: Panoramic Image of Hotel Folder



(a) Before Rectifying (b) After Rectifying

Figure 10: Panoramic Image of Yard Folder

(a) Before Rectifying (b) After Rectifying

Figure 11: Panoramic Image of Fishbowl Folder

(a) Ground Truth - Goldengate Folder (b) Ground Truth - Carmel Folder

Figure 12: Ground Truth Panoramic Images

(a) Ground Truth - Hotel Folder (b) Ground Truth - Yard Folder

Figure 13: Ground Truth Panoramic Images

Figure 14: Ground Truth - Fishbowl Folder