**Assignment 3**

Augmented Reality with Planar Homographies

Stereo Vision

A blue and yellow logo

Description automatically generated

Omar Salah Abdelkader 6809

Aly Mohamed Ali 6835

Youssef Yosry Mohamed 6953

**Part 1: Image Correspondences and Stitching**

*A) SIFT Descriptor and Image Stitching:*   
We initiated the process by extracting distinctive features using the SIFT descriptor from the book image and video screenshot. Leveraging established functions, we performed forward and inverse warping, seamlessly stitching the book image into the video frame.

A screenshot of a computer program

Description automatically generated

*B) Cropping and Aspect Ratio Transformation:*   
Ensuring visual coherence, we cropped the Kung Fu Panda video frames, preserving the middle segment. Subsequently, resizing was applied to maintain a consistent aspect ratio with the book frame. To maintain visual harmony, we employed the crop\_and\_resize\_image function. This function efficiently crops a specified segment from Kung Fu Panda video frames, preserving the middle portion. Subsequently, it resizes the cropped segment to match the consistent aspect ratio of the book frame, ensuring seamless integration.

A screenshot of a computer program

Description automatically generated

*C) Homography Matrix Application:*   
The homography matrix played a pivotal role in aligning the book and video images. Through forward and inverse warping, a harmonious integration was achieved, creating a visually appealing synthesis.



*D) Forward and inverse warp:*The forward\_warp function iterates through each pixel in the input image (img), applies the given homography matrix (H), and maps the transformed pixel position to the corresponding location in a new image (new\_img). The optional offset parameters allow adjustment for image alignment. This function effectively warps the input image using the specified homography. On the other hand, the inverse\_warp function performs an inverse warp operation. It initializes an empty stitched image (stitched\_image) and iterates through its pixels. For each pixel, it checks if the pixel value is zero, indicating that it has not been filled by the forward warp. If so, it uses the single\_inverse\_warp function to calculate the pixel value by applying the inverse homography (H\_inv) to map the pixel back to its original position in the input image (img). This process completes the stitching by filling in any gaps in the warped image.

A screen shot of a computer code

Description automatically generated

*A screen shot of a computer program

Description automatically generated*

*E) Iterative Processing:* Breaking down the video into frames, we applied the homography matrix iteratively using a dedicated function, "do\_all\_the\_stuff." This function seamlessly combined SIFT correspondences and homography application, accommodating variations in video length by cropping a section for optimal alignment. The function performs image stitching using SIFT keypoints and their descriptors to find matching points between two input images. It estimates the homography matrix H that aligns the keypoints in the right image to those in the left image. The stitched image is generated by warping the right image using the computed homography, overlaying it onto the left image, and handling any necessary image offsets. The resulting stitched image is returned.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of text

Description automatically generated

*F) Storage Optimization:* Due to computational constraints, frames were saved externally in Google Drive, allowing efficient memory utilization and smoother processing.

*G) Video Conversion and Audio Integration:* The transformed frames were compiled into videos, with audio extracted from Kung Fu Panda frames. To enhance visual aesthetics, frames were meticulously cropped to eliminate unwanted black pixels.

**Part 2: Sterio Vision**

*A) Block maching:*The block matching algorithm is a technique used in stereo vision to estimate disparities between corresponding pixels in left and right images. Disparities represent the horizontal shift required to align pixels in the two images. In this implementation, the function block\_matching takes two grayscale images, left\_image and right\_image, along with a specified window size and block type ("SAD" for Sum of Absolute Differences or "SSD" for Sum of Squared Differences). The images are padded to handle border cases, and for each pixel in the left image, the algorithm searches for the corresponding pixel in the same row of the right image. The pixel with the minimum difference in intensity is selected, and its horizontal position represents the estimated disparity. The output is a disparity map, where each pixel indicates the disparity value for the corresponding pixel in the left image. The disparity map provides valuable information about the scene's depth, helping reconstruct three-dimensional structures from stereo image pairs.

A screen shot of a computer program

Description automatically generated

*B) Dynamic Programming:*

1. cost function  
The cost function computes and updates the cost in a dynamic programming array D for stereo matching. It calculates the cost based on squared intensity differences between corresponding pixels in left and right grayscale images. The cost is initialized for the top-left corner and determined as the minimum among three possibilities (diagonal, left, and above) for other positions, with intensity differences scaled by dividing by 4. This function is integral to dynamic programming for stereo disparity mapping.

A computer screen shot of white text

Description automatically generated

2. Cost Computation  
These nested loops iterate over each row of the dynamic programming array D. The outer loop goes through each row, while the inner loops update the cost values within the array using the cost function. The first two inner loops handle the first column and first row of the array, computing costs for individual pixels. The subsequent nested loops cover the remaining interior positions, calculating costs based on the minimum of three possible paths (diagonal, left, and above) using the cost function. This process populates the entire dynamic programming array for stereo matching.

A screen shot of a computer program

Description automatically generated

3. Disparity Map Backtracking Algorithm  
This algorithm performs backtracking on the dynamic programming array to extract disparity values for each pixel in the stereo image pair. It starts from the bottom-right corner of the array and iteratively selects the minimum cost path among three possibilities (diagonal, left, and above). The corresponding disparity values are assigned to the left and right disparity maps. This process continues until reaching the top-left corner of the array, completing the construction of the disparity maps for stereo matching.

A computer screen shot of a code

Description automatically generated

Bonus

To draw the graph then we need to save the indices of while backtracking, but we need to work on single line only at a given time.

If a pixel in left is skipped then that means reducing the index of i by one and since it's drawn on the vertical axis that will result in drawing a vertical line, same can be observed for the horizontal line, and as the diagonal line we reduce both the indices in the left and right images which means drawing a diagonal line.

A computer screen shot of a program code

Description automatically generated

A computer screen shot of text

Description automatically generated

A graph with a line

Description automatically generated