# Image Processing - Exercise 4

Shir Hana Stern, shirste, 325945236

## Introduction

In this project, we undertook the implementation of a stereo mosaicing algorithm. The primary objective was to develop a system capable of stitching together a sequence of overlapping images captured from a moving camera to create a panoramic view. The core of the algorithm involved several fundamental computer vision concepts. Firstly, feature detection and matching were employed to establish correspondences between key points in consecutive frames. The Scale-Invariant Feature Transform (SIFT) algorithm, renowned for its robustness to scale and rotation changes, was instrumental in this regard. These detected features acted as anchor points, guiding the subsequent steps in the process.
Secondly, the concept of transformation was pivotal. Once corresponding features were identified, a homography matrix was computed to model the geometric transformation between the frames. This matrix encapsulated the translation, rotation, and scaling required to align one image with another. By applying this transformation to each frame, we were able to warp them into a common coordinate system.
Finally, image stitching was performed to combine the warped images into a seamless panorama. This involved blending the overlapping regions of the images to create a visually coherent output.
In essence, this project involved a systematic process of feature detection, matching, homography estimation, and image warping to achieve the goal of creating panoramic images from video sequences.

## Algorithm

In general, this stereo mosaicing algorithm operates in the following stages:

**Feature Detection and Description:** The Scale-Invariant Feature Transform (SIFT) algorithm is employed to identify distinctive keypoints within each frame of the input video sequence. For each detected keypoint, a unique descriptor is extracted, capturing the local image information.
**Feature Matching:** Keypoints are matched between consecutive frames using a nearest neighbor search approach. To enhance the accuracy of matching, a ratio test is applied, which retains only those matches where the distance to the best match is significantly smaller than the distance to the second-best match.
**Homography Estimation:** The RANSAC (Random Sample Consensus) algorithm is utilized to estimate the homography matrix that relates the corresponding keypoints

between two frames. RANSAC is robust to outliers and effectively handles potential mismatches, ensuring a reliable estimation of the geometric transformation between frames.

**Image Warping:** Each frame is then warped using the computed homography matrix. This involves transforming the coordinates of each pixel in the current frame to its corresponding location in the reference frame. To seamlessly integrate the warped frames, techniques such as linear blending or pyramid blending can be employed to minimize artifacts in the final panorama.

**Panorama Generation:** The warped frames are accumulated into a final panoramic image. This process is repeated for each pair of consecutive frames in the video sequence, resulting in a series of overlapping panoramas. These individual panoramas are then stitched together to create a dynamic panoramic video.

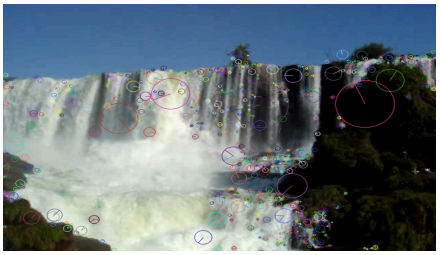
For changing viewpoint mosaic, the algorithm explores different viewpoints within each frame to create multiple panoramas.

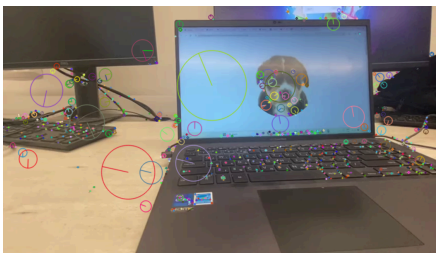
By shifting the "center" of focus within each frame and stitching together the corresponding warped strips, the algorithm effectively generates a series of panoramic images with varying perspectives. This allows the user to explore different sections of the overall scene captured in the video.

For dynamic mosaic, the algorithm segment refines the panorama generation process by intelligently selecting and combining portions of the warped frames. It iterates through the width of the canvas, effectively dividing it into vertical strips. For each strip, it analyzes the warped frames, identifying regions with a significant number of non-zero pixels, indicating the presence of meaningful visual information. By establishing a threshold for the number of non-zero pixels, the algorithm filters out frames that do not contribute substantially to the overall panorama. This selective approach helps to reduce noise, minimize artifacts, and enhance the visual clarity of the final panorama by ensuring that only the most informative portions of the warped frames are integrated into the output.

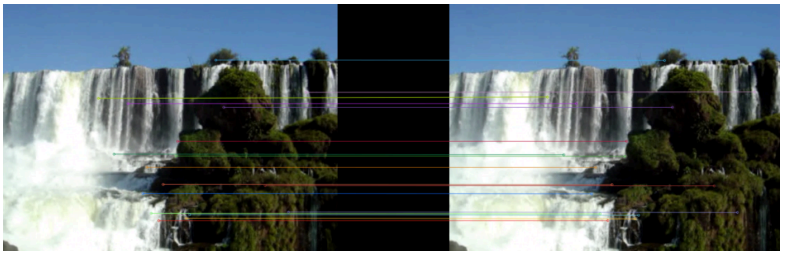
Essentially, this mechanism creates multiple "columns" within the panorama. Each column represents a slightly different viewpoint within the scene, as it focuses on a specific vertical region of each frame. This multi-column approach allows for a more comprehensive and nuanced representation of the captured scene.

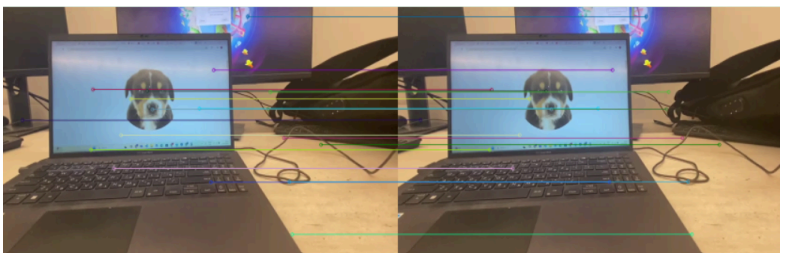
**Feature Detection and Description**





**Feature Matching**





**Feature Detection and Description:** Input: A single video frame. Output: Keypoints (locations with size and orientation) and feature descriptors (local structural data). Visualize by overlaying circles and arrows on the frame to indicate keypoints and their orientations.
**Feature Matching:** Input: Keypoints and descriptors from two consecutive frames. Output: Matches indicating corresponding points between the frames. Visualize by displaying the frames side by side with lines connecting matched keypoints.
**Homography Estimation:** Input: Matched keypoints. Output: A transformation matrix for aligning frames and a set of consistent matches.
**Image Warping:** Input: A frame and a transformation matrix. Output: The warped frame aligned with a reference frame. Visualize by displaying the original and warped frames side by side.
**Panorama Generation:** Input: Warped frames and blending settings. Output: A panoramic image. Visualize intermediate panoramas as they build, highlighting frame overlaps and blending.
**Changing Viewpoint Mosaic:** Input: Warped frames and focus regions (vertical strips). Output: Multiple panoramas with varying perspectives. Visualize by displaying panoramas side by side, marking the regions of focus.
**Dynamic Mosaic:** Input: Warped frames and a threshold for meaningful areas. Output: A refined panorama including only the most informative parts. Visualize by overlaying heatmaps on the final mosaic to highlight selected areas and showing examples of excluded regions.

# Implementation Details

**Feature Detection and Description:** (from scratch) The first frame is converted to grayscale for robust feature detection using the SIFT algorithm. The `detectAndCompute` method extracts keypoints and descriptors. Subsequent frames undergo the same process during their addition to the mosaic. (library use) OpenCV's SIFT algorithm is employed for keypoint detection. (hyper parameters) Contrast threshold: Set to `0.04` to filter out weak features. Edge threshold: Default OpenCV values were retained for robust detection in cluttered scenes.
**Feature Matching:** (from scratch) The `match` method identifies matches between descriptors of consecutive frames using a k-NN matcher with a ratio test for reliability. A maximum of 20 best matches are retained, sorted by distance. Visualizations of matched points are optionally displayed. (library use) OpenCV's BFMatcher is used for k-NN matching. (hyper parameters) Ratio threshold: Set to `0.75` based on Lowe's paper to ensure accurate matching. Maximum matches: Limited to 20 for computational efficiency and visual clarity.
**Homography Estimation:** (from scratch) Matches between keypoints are used to

estimate an affine transformation (translation + rotation) via the `cv2.estimateAffinePartial2D` function, employing the RANSAC algorithm for outlier removal. The 2x3 transformation matrix is extended to a 3x3 matrix for matrix multiplication with cumulative transformations. (library use) OpenCV's `estimateAffinePartial2D` is used. (hyper parameters) RANSAC iterations: Set to `2000` for robust outlier rejection.

**Image Warping and Stitching:** (from scratch) The `warp_and_store` method applies the cumulative transformation to the current frame using `cv2.warpAffine`. Warped frames are added to the canvas. Overlapping regions are handled by combining images pixel-wise, prioritizing color information from one frame while filling gaps with the other. (library use) OpenCV's `warpAffine` function.

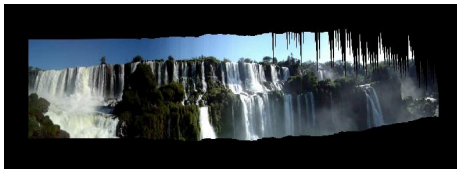
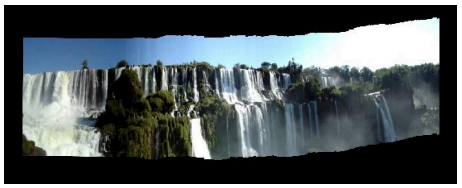
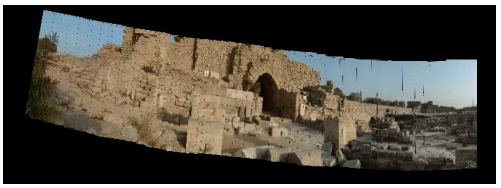
**Dynamic Mosaic Generation:** (from scratch) For dynamic mosaics, the canvas is divided into fixed-width vertical strips. A threshold determines whether a strip from the current frame is included based on its non-zero pixel coun. Masks are generated to visualize regions included (focus mask) or excluded (excluded mask). (library use) OpenCV for mask creation and visualization. (hyper parameters) Strip width: Set to `50 pixels` to balance visualization and computational overhead.

**Viewpoint Mosaic Generation:** (from scratch) Specific regions are extracted using `extract_region`, defined by corner coordinates and null proportions on either side. The regions are blended sequentially into the panorama. (library use) OpenCV for region extraction and blending. (hyper parameters) Null proportions: Set to `10%` to exclude noisy edges.

During the implementation, one significant challenge was attempting to solve the exercise directly using the provided videos. The algorithm's complexity made it difficult to debug and understand errors or identify missing components. To address this, I broke the task into smaller steps. I started with a very short video that I created and had minimal shifts between frames, allowing me to verify that the transformation matrix was logical and functioning correctly. Adding visual outputs during the process, such as matched keypoints and intermediate panoramas, also helped me ensure I was on track at each step.

Another challenge arose while building panoramas for viewpoint mosaics. Initially, the results were unclear, and I couldn't determine whether the issue stemmed from my implementation or the video inputs. Ultimately, I realized the need to adopt a window-based approach. This method allowed me to handle each panorama by focusing on a specific center point, This adjustment enabling the mosaics to reflect the intended viewpoint transformations.

# Visual Results

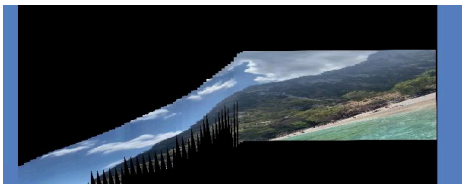
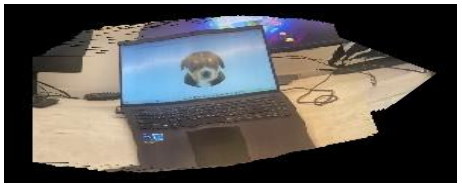
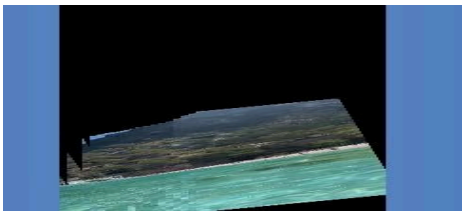
| | **dynamic mosaic:** | **viewpoint mosaic:** |
|---|---|---|
| 1) first frame: |  |  |
| 2) middle frame: |  |  |
| 3) last frame: |  |  |

The final results highlight several notable observations and some flaws. In the dynamic video, the movement of water is clearly visible, with sequential frames effectively capturing the descent of the waterfall. This showcases the algorithm's ability to track dynamic motion over time. However, the alignment is not perfect, as the stitched frames exhibit slight misalignments along the x-axis. This misalignment becomes apparent when observing the flow, where parts of the image do not transition smoothly.

In the viewpoint video, the results successfully demonstrate varying perspectives of Caesarea. Each frame contributes a unique angle, effectively creating a wide-angle panorama of the scene. Despite this, there are visible issues with the stitching process. The seams between frames are noticeable, and faint lines can be seen where adjacent images meet. These lines disrupt the overall visual continuity and are a direct result of imperfect blending and alignment.

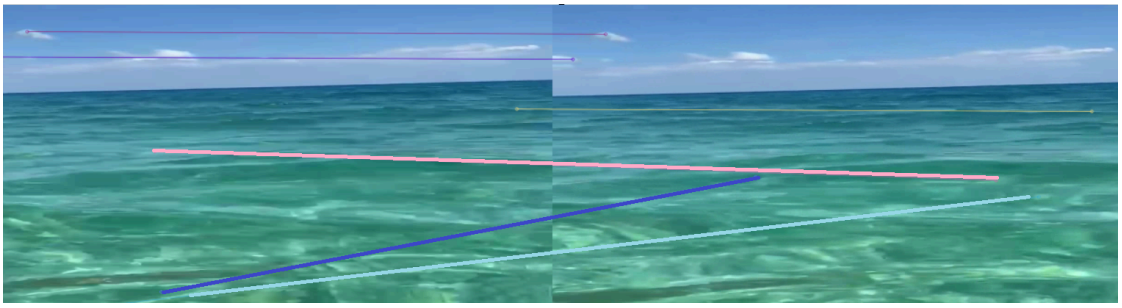
Flaws
Alignment Issues: Both videos exhibit a lack of perfect alignment, particularly noticeable along the x-axis, which affects the smoothness of transitions.
Seam Visibility: The seams between stitched frames, while subtle, are visible and detract from the quality of the final output.
Feature Matching Errors: In regions with repetitive patterns, the algorithm struggles to find accurate matching points, which may contribute to stitching errors.
These flaws highlight areas where the algorithm could be improved, such as refining the blending process, enhancing alignment techniques, or incorporating advanced methods to handle repetitive patterns and minimize visible seams.

My own videos:

| | **good mosaic (viewpoint):** | **bad mosaic:** |
|---|---|---|
| 1) first frame: |  |  |
| 2) middle frame: |  |  |
| 3) last frame: |  |  |

The primary difference between the two videos lies in how they adhere to the assumptions of the algorithm. In the "good" video, the camera movement was controlled, with only the phone moving relative to the computer screen, ensuring that the frames exhibit rigid transformations. This allowed the algorithm to function as expected.

In contrast, the "bad" video, which captures the sea, violates the algorithm's assumptions. The camera was moved erratically, introducing non-rigid features between frames. Additionally, much of the video predominantly displays the sea, resulting in matching errors due to the repetitive and similar appearance of features across frames (e.g., many points in the sea appear similar when characterized using SIFT). As a result, the algorithm struggles to find accurate correspondences, leading to suboptimal results.



As seen in the figure, the matches between the two frames do not correspond to the actual same points. This discrepancy highlights the failure of the algorithm to accurately identify true correspondences, primarily due to the repetitive and similar features in the sea region, which lead to incorrect matching.

# Conclusion

This exercise was both challenging and rewarding, offering a deep dive into the complexities of image processing. One of the most valuable lessons I learned was the importance of effective debugging, which in image processing often relies heavily on visualization. Being able to see intermediate results helped me identify issues and refine the algorithm step by step.

Another critical insight was the necessity of thoroughly planning the algorithm before diving into coding. By outlining all the details in advance, I was able to break the problem into manageable intermediate steps, making the solution more approachable and systematic.

Additionally, I gained practical knowledge in several key areas:

1. **Feature Detection with SIFT:** Understanding how to extract distinctive features from an image for subsequent analysis.
2. **Feature Matching and Refinement with RANSAC:** Learning to improve the reliability of matching points by eliminating outliers and ensuring robust correspondence.
3. **Transformation Matrix and Warping:** Computing the transformation matrix from matched points and applying it to warp images effectively.
4. **Constructing Panorama Arrays (Time vs. Perspective):**
    - **Time-based Panoramas:** Frames are stitched sequentially, emphasizing continuity over time. This approach highlights dynamic changes, such as movement in the scene.
    - **Perspective-based Panoramas:** Frames are stitched based on their spatial perspective, creating a comprehensive wide-angle view of a single scene from different vantage points.

Overall, this exercise reinforced the importance of visualization, planning, and modular problem-solving while deepening my understanding of advanced image processing techniques.