1. **Introduction**

   The goal of this exercise is to implement stereo mosaicing - generating a video of panoramas from different viewpoints, creating a parallax effect when played back. The input is a video panning across a scene; the output is multiple panoramas, each representing the scene as viewed from a slightly different horizontal position.

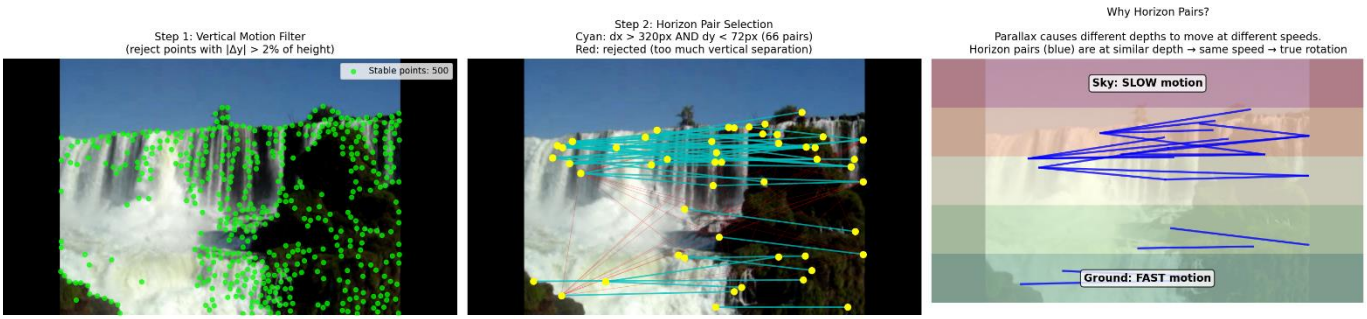
   The main techniques used are:

   - Optical flow for tracking features between consecutive frames
   - Homography estimation using horizon pairs to avoid parallax-induced errors (banana effect)
   - Homography accumulation to align all frames to a common reference
   - Stitching and Barcode blending with Laplacian pyramid blending to eliminate blinking artifacts and create seamless transitions between viewpoints

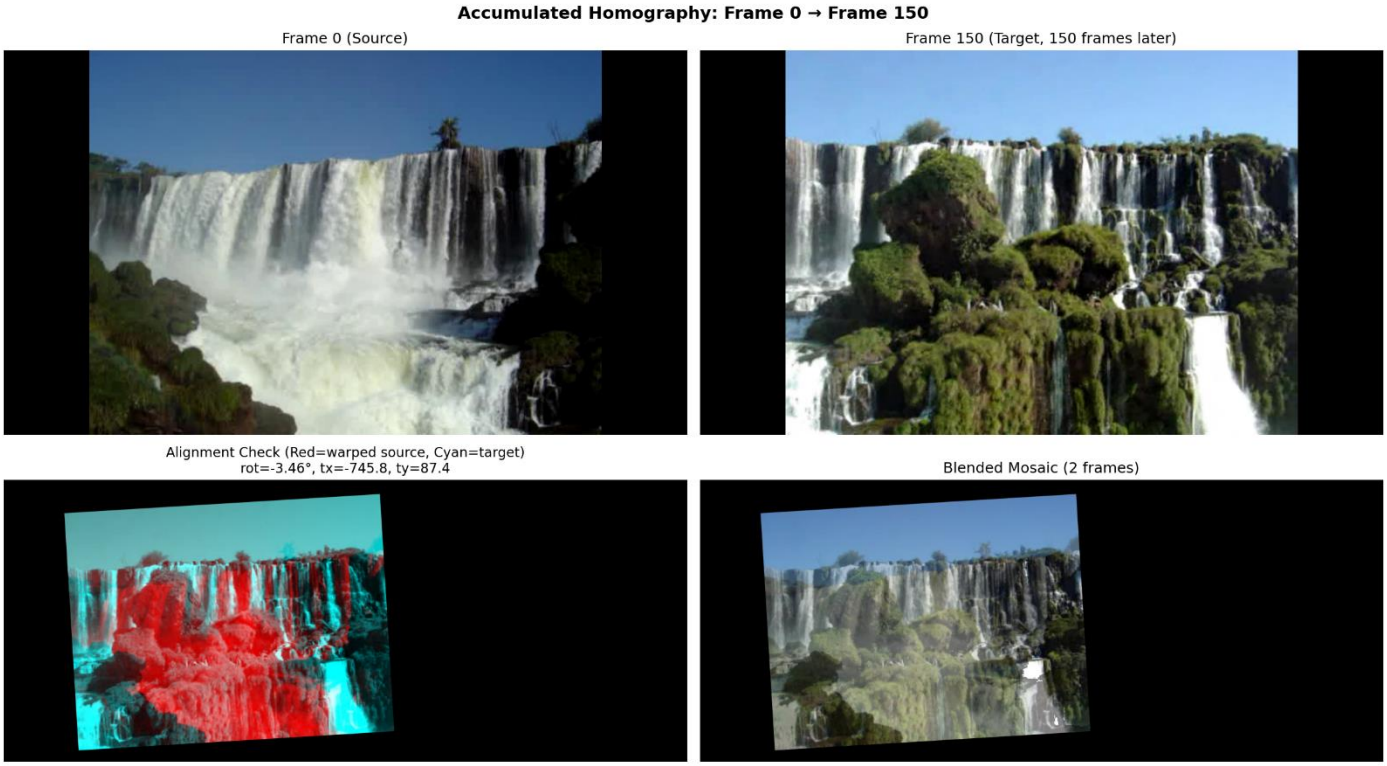
2. **Algorithm:**

2.1. **Feature Detection and Tracking**: Input: Pair of frames. Output: Set of corresponding point pairs. We start with feature detection using Shi-Tomasi, which is a refinement of the Harris algorithm we saw in class described in "Good Features to Track" (1994). For each pair of frames of the input video we detect them and track using Lucas-Kanade optical flow, here is an example on boat.mp4:

2.2. **Homography Estimation:** Input: Set of tracked points. Output: 3x3 Homography Matrix. Rather than using all tracked points to estimate homography, we filter and select only “horizon pairs” – pairs of points that are horizontally distant but close vertically, thus we ensure that the relative motion reflects true camera rotation and is not corrupted by the parallax, thus we avoid the “Banana” effect. The rotation is estimated from angle change between these pairs of points and getting their median. Translation is then computed by applying the rotation and finding the median residual displacement. For robustness we filter out vertically moving points above a threshold to negate moving objects like water in a waterfall. An example on Iguazu.mp4:

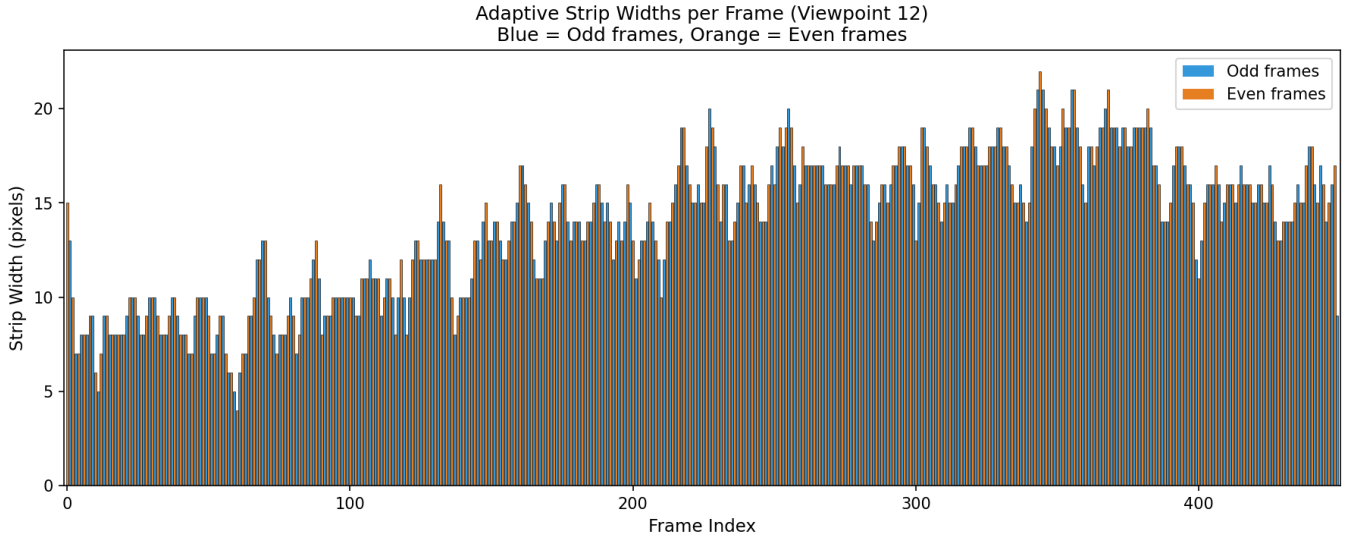


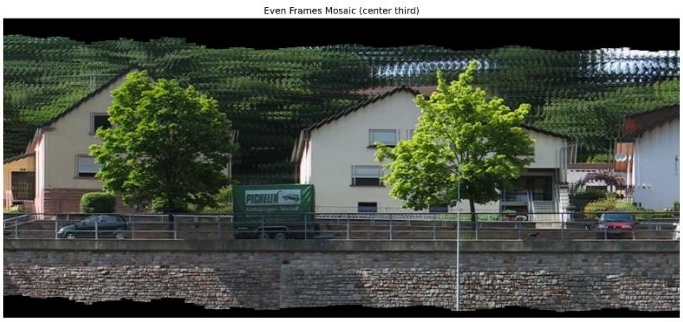
2.3. **Homography accumulation**: Input: Set of frames and a reference frame. Output: Aligned frames relative to a reference frame. Each frame-to-frame homography captures only the small motion between consecutive frames. To align all frames to a common reference (like the middle frame), we chain these transforms through matrix multiplication (In the right order). This will allow us later to slice the frames and stitch those slices to create panoramas with proper homography even without content overlap between a source frame and a target frame, which will be the center of our viewpoint.

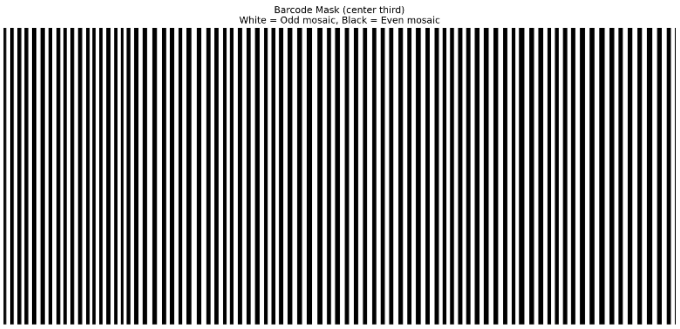


2.4. **Stitching and Barcode Blending**: Input: Aligned frames and their relative motion. Output: Two intermediate even/odd panoramas and a blending mask. To generate multiple viewpoints for the stereo effect, we build panoramas by taking vertical strips from each frame at viewpoint-dependent positions, with their width dependant on the motion between frames. However, naive strip selection causes blinking artifacts (see Section 3.4). Our solution (Inspired by "Real-Time Stereo Mosaicing using Feature Tracking", Peleg et al, 2011): build two separate mosaics -

one from odd-indexed frames, one from even-indexed frames - each using double-width strips. A "barcode" mask alternates between these mosaics, and Laplacian pyramid blending smooths the transitions. This ensures each location is covered by overlapping contributions, eliminating sudden appearance/disappearance of objects. While min-cut can give sharper results that ensure we don't cut out objects with fast flow, it is computationally more expensive, but I do explore it in the bonus section. Results on a video generated out of 24 panoramas based on boat.mp4:







3. **Implementation**:

3.1. **Description**: While an educational attempt was made by me to implement Harris, Shi-Tomasi and Lucas-Kanade, OpenCV is substantially more optimised and gave slightly better results with less computational time. All the sub-algorithms were implemented or imported in the order of description in last section, myriad of tests were performed to see which thresholds and hyper-parameters were working the best for the videos I worked with, ultimately what differs in the implementation from the plain description is the tuning of parameters and addressing the unexpected artifacts in computation, more about that in section **3.4**.

3.2. **Libraries used**:

Component	Library	Function
Corner detection	OpenCV	goodFeaturesToTrack (Shi-Tomasi)
Optical flow	OpenCV	calcOpticalFlowPyrLK (Lucas-Kanade)
Image warping	OpenCV	warpPerspective, remap
Pyramid blending	OpenCV	pyrDown, pyrUp
Matrix operations	NumPy	Homography multiplication, median statistics
From scratch	-	Horizon pairs selection, barcode mask generation, homography accumulation, strip boundary computation, overall logic

3.3. **Hyper-parameters and thresholds**:

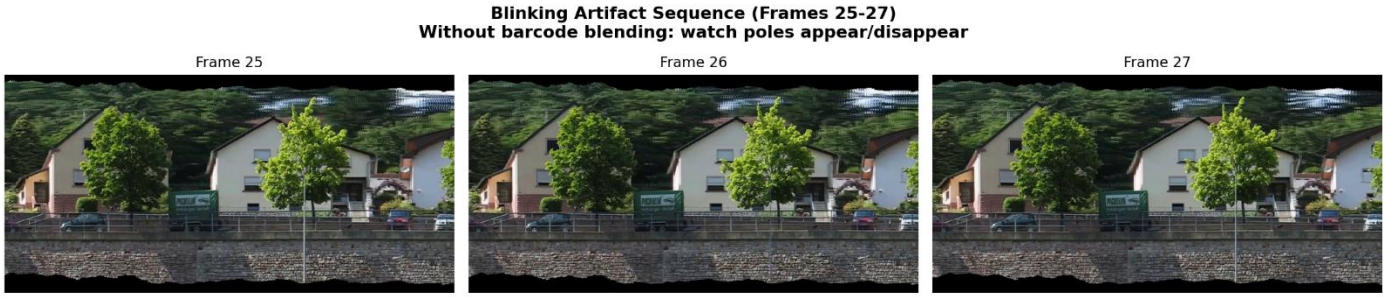
Parameter	Value	Location/Panpose	Justification
maxCorners	2000	Shi-Tomasi detection	Large pool to filter from: ensures enough points survive aggressive filtering
qualityLevel	0.01	Shi-Tomasi detection	Low threshold captures more corners, we filter later
minDistance	7 px	Shi-Tomasi detection	Prevents point clustering, ensures spatial spread
blockSize	7 px	Shi-Tomasi detection	Neighborhood size for corner response calculation
winSize	(21, 21)	Lucas-Kanade tracking	Large window for robust tracking in textured scenes
maxLevel	3	Lucas-Kanade tracking	Pyramid levels for handling larger frame-to-frame motions
criteria	30 iter, ε=0.01	Lucas-Kanade tracking	Convergence criteria for iterative refinement
max_vertical_drift	2% of height	Vertical motion filter	Rejects dynamic elements (like water); assumes horizontal pan
min_dist_x	25% of width	Horizon pairs selection	Points must be horizontally distant for stable angle estimation
max_dist_y	10% of height	Horizon pairs selection	Points must be vertically close (similar depth band) to avoid parallax
n_samples	300	Horizon pairs selection	Random subset for computational efficiency
max_rotation	3°	Rotation clamp	Safety net: >3°/frame is unrealistic; indicates outlier noise
DIRECTION_FRAME_GAP	8 frames	Direction detection	Sufficient motion between frames to reliably detect L2R vs R2L
num_samples	5	Direction detection	Average over multiple frame pairs for robustness
levels	4	Pyramid blending	Standard depth: balances detail preservation vs blend smoothness
margin	10% of width	Panorama generation	Avoids sampling black edges when computing viewpoint slice centers
num_viewpoints	24 (default)	Panorama generation	Smooth stereo animation; sufficient density for 3D effect
strip_width	2 frames	Barcode blending	Double-width strips create overlap for smooth odd/even transitions

3.4. **Challenges and Solutions**:

Banana Effect: When estimating homography from all points, parallax causes different depths to suggest different rotations. The far objects move slowly while the near move fast, making the algorithm perceive rotation that isn't there. The panorama curves like a banana.

Solution: Horizon pairs method - only use point pairs at similar depths (horizontally far, vertically close).

Dynamic Scenes: In videos with waterfalls or potentially levitating or falling objects, tracked points on dynamic elements corrupt the homography estimate with false motion.
Solution: Vertical motion filter - reject any tracked point with vertical displacement exceeding 2% of frame height. Real camera pan produces primarily horizontal motion.
Blinking Artifacts: When building panoramas by selecting one strip per frame, thin and fast objects like poles may fall inside a strip in one viewpoint but outside in the next, causing them to appear and disappear as the video plays. An example from early attempts:

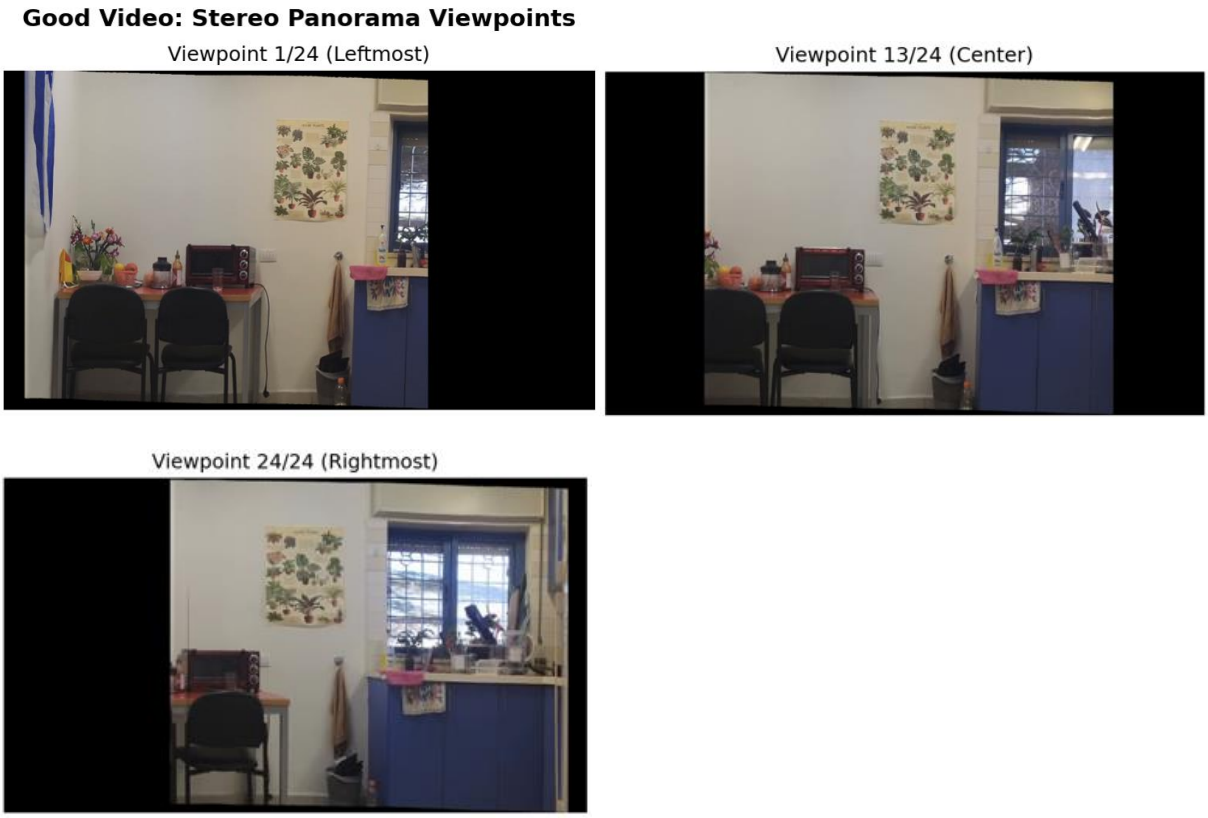


Solution: Barcode blending with double-width strips. By building odd and even mosaics separately and blending them, each location receives contributions from multiple frames, smoothing out the transitions.

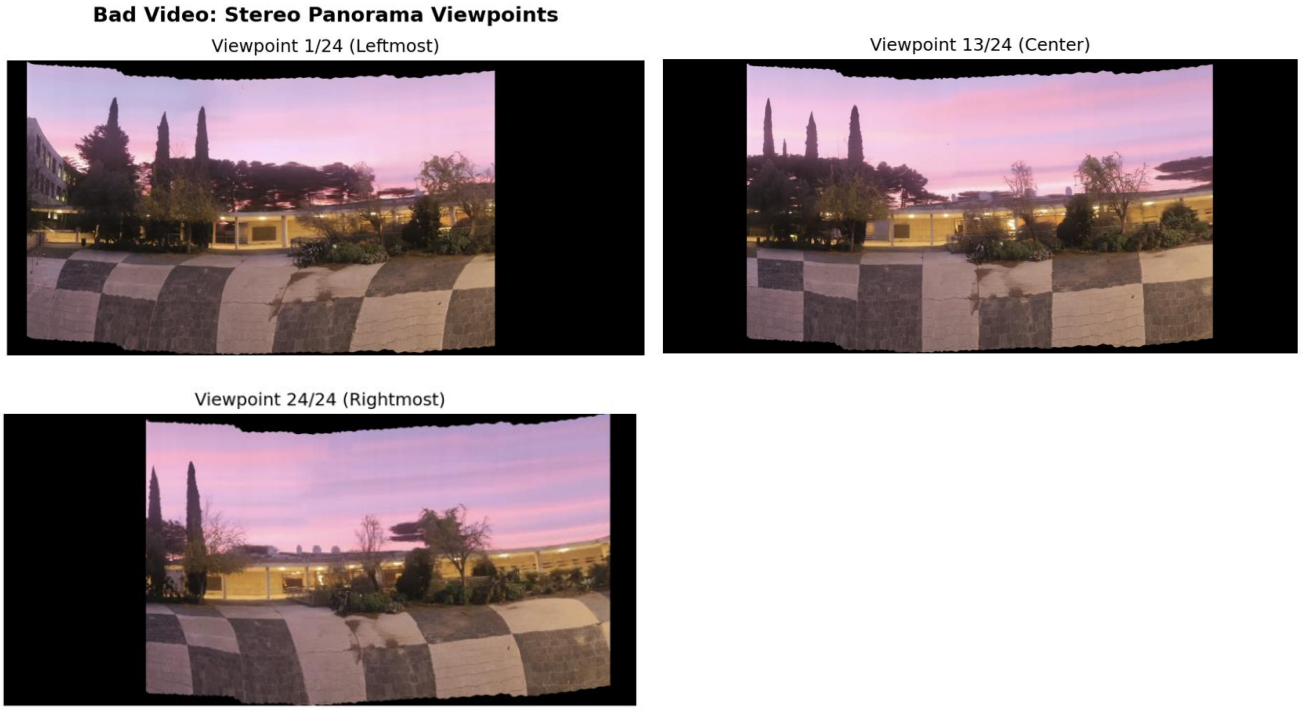
4. **Visual results**:
   Viewpoint Mosaic (Boat):

First, middle, and last viewpoints showing the stereo parallax effect - notice how object positions shift between viewpoints. While barcode blending negated the blinking effect, the blending might have smeared small details, as we will see in the bonus section, min-cut gives sharper results.

Personal Video – Good: A panorama built out of a video of my kitchen.



Personal Video – Bad: A video near the entrance to the university.



Why is this video a bad input? This video fundamentally violates the Planar Scene / Infinity Assumption. The algorithm relies on the premise that the scene acts as a single flat plane at infinity relative to the camera motion. However, this specific footage contains two distinct depth layers: the ground tiles (near plane) and the buildings (far plane). A single homography cannot

align both simultaneously because the ground moves significantly faster than the buildings due to strong parallax. As a result, the output shows severe stretching and ghosting in the lower region while the center (buildings) remains clean. Successful inputs, such as the Boat video or my Kitchen example, adhere to the assumption that objects of interest are sufficiently distant relative to their internal depth variation, allowing a single transformation to approximate the motion without these artifacts.

5. **Conclusion**:
   In conclusion, this project demonstrated the potential of stereo mosaicing to create convincing parallax effects from standard 2D video inputs. By implementing techniques such as "Horizon Pairs" for rotation stability and barcode blending for temporal consistency, we were able to mitigate significant issues like the "Banana Effect" and blinking artifacts. While the current implementation relies on specific scene assumptions to function correctly, it serves as a solid and expandable baseline. As explored in the bonus section, integrating more advanced seam-finding methods like Min-Cut offers a clear path to further refine visual fidelity, suggesting that this technology is a promising step toward more immersive 3D scene reconstruction, and other effects that go beyond sideways panoramic views.