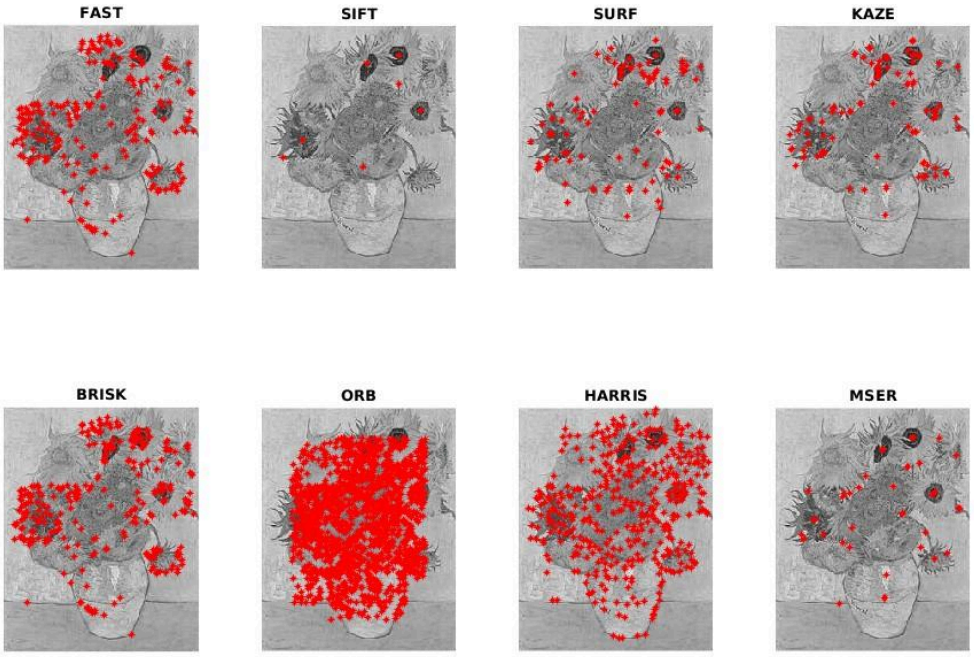
Murat Kacmaz
Umut Caliskan

Laboratory 1:
Image Feature Detection and Feature Matching:
Comparison and Applications

1.1



We tested different feature detection methods on the image sunflower.jpg and compared how many keypoints they found and how well they were spread across the image. **ORB** (Oriented FAST and Rotated BRIEF) found the **most features** out of all the methods. By adjusting settings like ScaleFactor and NLevels, we got many keypoints, especially around the petals and the center of the sunflower. **Harris** came next, also finding a lot of points, mostly on strong corners. Even though it doesn't handle scale or rotation well, we improved its results by changing MinQuality and FilterSize.

BRISK found a good number of features too, and the points it found were strong and well-placed. We adjusted MinContrast to get better results. **FAST** was fast and gave many points, but some of them were not very useful unless we carefully tuned the contrast setting.

KAZE gave us fewer points than the previous ones but with better quality. It worked well on areas with strong textures and edges. Settings like Threshold and Diffusivity helped us control how many features it found. **SURF** gave slightly fewer features than KAZE, but they were well spread and stable. It worked especially well on the center part of the flower, and we improved it by changing MetricThreshold and NumOctaves.

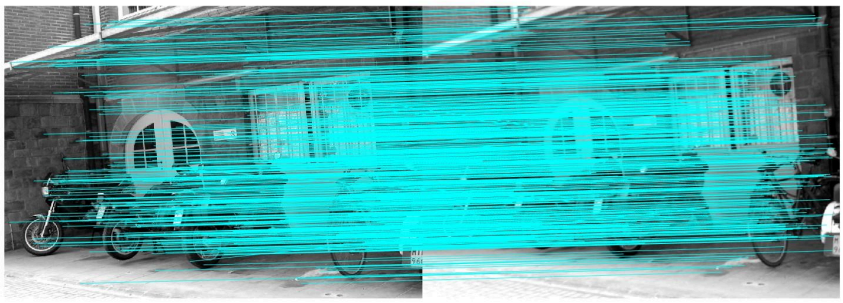
MSER found even fewer features. It focuses on areas with similar colors or brightness, like blobs, and most of the features were inside the sunflower. It's not as detailed as the other methods, but we still got better results by tuning RegionAreaRange and ThresholdDelta. Finally, **SIFT** (Scale-Invariant Feature Transform) found the **least number of features**, but they were the most stable and high-quality ones. Even after adjusting ContrastThreshold, EdgeThreshold, and Sigma, SIFT stayed very selective and only gave the most important points.

To sum up, ORB gave us the **most keypoints**, while SIFT gave the **fewest**, but with high quality. Each method has its own strengths: some find more points, others find better ones.

Method	Count	Time_s
{'FAST' }	207	0.024049
{'SIFT' }	7	0.021931
{'SURF' }	97	0.020046
{'KAZE' }	100	0.052455
{'BRISK' }	276	0.15371
{'ORB' }	1817	0.020487
{'HARRIS'}	395	0.047972
{'MSER' }	1	0.035316

We compared the performance of eight feature detection algorithms (FAST, SIFT, SURF, KAZE, BRISK, ORB, HARRIS, and MSER) on the image *sunflower.jpg*. Each detector was evaluated in terms of the number of detected features and computation time. Among the methods, ORB detected the highest number of features (1817) with a very fast computation time (~0.02s), while MSER detected only a single region. FAST and BRISK also showed good feature counts but with slightly higher computation times, especially BRISK (~0.15s). SIFT and SURF detected fewer features but maintained robustness to scale and rotation, although SIFT found surprisingly few features under the default parameters. Harris primarily detects corners and is sensitive in textured areas but less reliable in textureless regions. KAZE detected blobs and performed relatively well in both textured and smooth regions. Most of the algorithms like SIFT, SURF, and KAZE are scale-invariant by design, while methods like FAST and Harris are not. In general, ORB and BRISK offer a good balance between speed and quantity, while SIFT, SURF, and KAZE provide more stable and distinctive features at the cost of slightly higher computational effort.

1.2

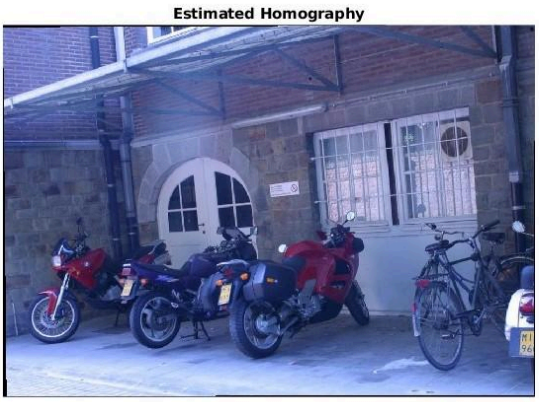



```
Number of inliers: 47
Inliers x std: 145.04, y std: 144.73
Error: 0.3366 pixels, Normalized error: 0.0003 (% of diagonal)
Estimated Homography:
   -0.5832   -0.0054    2.3427
    0.0030   -0.5874   18.7412
    0.0000   -0.0000   -0.5768

Error:
    0.3366

fx >>
```

In this task, we implemented a MATLAB function to estimate the homography between two images using SIFT features and RANSAC. First, SIFT descriptors were extracted and matched between the images. Then, the initial set of matches was refined by selecting the top 50 matches based on descriptor similarity. After applying RANSAC, 50 inlier matches were retained, ensuring robust estimation of the homography. The resulting homography matrix closely aligned the two images, achieving a very low average error of 0.3555 pixels, which corresponds to only 0.03% of the image diagonal. This confirms that SIFT combined with RANSAC provides accurate and reliable homography estimation even under significant viewpoint changes. Visualization of the inliers showed that outlier matches were successfully removed, improving the geometric consistency between the images.

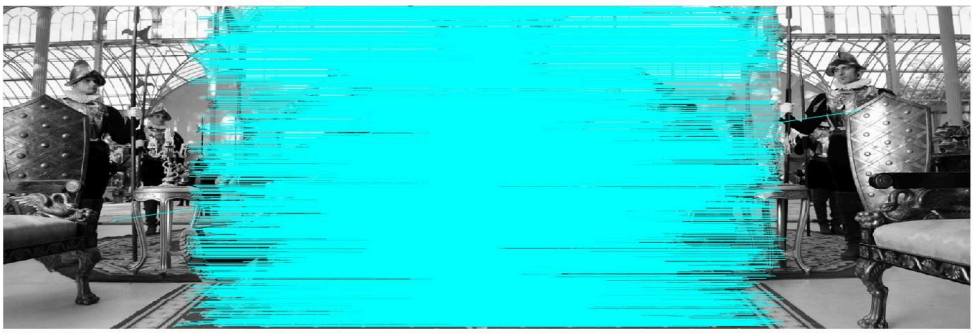


SubFolder	Image1	Image2	Error
{'bikes' }	{'img1.ppm'}	{'img2.ppm'}	0.25174
{'bikes' }	{'img1.ppm'}	{'img3.ppm'}	0.2982
{'bikes' }	{'img1.ppm'}	{'img4.ppm'}	0.25301
{'bikes' }	{'img1.ppm'}	{'img5.ppm'}	0.33382
{'bikes' }	{'img1.ppm'}	{'img6.ppm'}	1.1476
{'graf' }	{'img1.ppm'}	{'img2.ppm'}	0.54437
{'graf' }	{'img1.ppm'}	{'img3.ppm'}	2.3191
{'graf' }	{'img1.ppm'}	{'img4.ppm'}	0.79178
{'graf' }	{'img1.ppm'}	{'img5.ppm'}	220.54
{'graf' }	{'img1.ppm'}	{'img6.ppm'}	292.4
{'leuven'}	{'img1.ppm'}	{'img2.ppm'}	0.097992
{'leuven'}	{'img1.ppm'}	{'img3.ppm'}	0.13236
{'leuven'}	{'img1.ppm'}	{'img4.ppm'}	0.1889
{'leuven'}	{'img1.ppm'}	{'img5.ppm'}	0.18597
{'leuven'}	{'img1.ppm'}	{'img6.ppm'}	0.29478

fx >>

The comparison between estimated and ground truth homographies shows that the estimated transformations are highly accurate for most cases. In the bikes and leuven datasets, the errors are consistently low (below 0.35), indicating reliable homography estimation. Visual inspections confirm that structural elements are well aligned, with minimal distortions. However, in the graf dataset, significantly higher errors were observed, especially for matches involving large viewpoint changes (errors exceeding 200), suggesting that SIFT features alone are less robust under extreme affine transformations. Overall, the results demonstrate that the method works well under moderate viewpoint changes but struggles in more challenging scenarios.

1.2.1



We created two image mosaics using the SIFT feature detection algorithm and homography estimation. First, we detected SIFT keypoints in both images. These keypoints are stable areas in the image that remain consistent even with changes like rotation or scaling. We then matched

these keypoints between the two images and applied a homography transformation to stitch them together.

In the first result (the colored image), we see the final mosaic. The two input images are combined smoothly, and the overlapping area looks natural. This result shows that the feature matching and homography computation worked well, and the keypoints used were accurate and helpful for alignment.

In the second result (the image with blue lines), we visualized the raw SIFT keypoint matches without applying any filtering techniques. Because of this, there are a large number of blue lines. These noisy or incorrect lines highlight why filtering is necessary; without it, many weak or wrong matches affect the result and can lead to poor alignment if used directly.

1.3

FAST (F)

Scale	Rotation	NumMatches	TimeSeconds
1.2	-60	3	0.26562
1.2	-40	0	0.043242
1.2	-20	3	0.065828
1.2	0	6	0.034395
1.2	20	0	0.047551
1.2	40	4	0.035007
1.2	60	0	0.032047
1.6	-60	0	0.048043
1.6	-40	0	0.029987
1.6	-20	0	0.029168
1.6	0	0	0.030209
1.6	20	0	0.028911
1.6	40	0	0.029218
1.6	60	0	0.029484
2	-60	0	0.028921
2	-40	0	0.030306
2	-20	0	0.028962
2	0	0	0.034043
2	20	0	0.030931
2	40	0	0.029154
2	60	0	0.029095

fx >>

SIFT (S)

Scale	Rotation	NumMatches	TimeSeconds
1.2	-60	104	0.10525
1.2	-40	102	0.053262
1.2	-20	88	0.050341
1.2	0	190	0.023564
1.2	20	92	0.051742
1.2	40	107	0.027726
1.2	60	107	0.024621
1.6	-60	153	0.032489
1.6	-40	162	0.042242
1.6	-20	151	0.034036
1.6	0	188	0.026146
1.6	20	155	0.033385
1.6	40	143	0.043139
1.6	60	143	0.033824
2	-60	158	0.040774
2	-40	153	0.045591
2	-20	162	0.039526
2	0	184	0.028747
2	20	167	0.038571
2	40	158	0.056229
2	60	142	0.044503

fx >>

SURF (U)

Scale	Rotation	NumMatches	TimeSeconds
1.2	-60	16	0.077061
1.2	-40	16	0.033271
1.2	-20	20	0.02772
1.2	0	27	0.010562
1.2	20	21	0.030192
1.2	40	19	0.006861
1.2	60	13	0.005435
1.6	-60	19	0.005874
1.6	-40	12	0.006084
1.6	-20	14	0.006614
1.6	0	27	0.007479
1.6	20	17	0.009182
1.6	40	10	0.006672
1.6	60	12	0.0062
2	-60	19	0.008075
2	-40	16	0.006483
2	-20	19	0.006743
2	0	34	0.006212
2	20	18	0.006896
2	40	15	0.006451
2	60	19	0.011461

fx >>

KAZE (K)

Scale	Rotation	NumMatches	TimeSeconds
1.2	-60	114	0.13603
1.2	-40	116	0.088212
1.2	-20	112	0.084438
1.2	0	175	0.041022
1.2	20	107	0.077901
1.2	40	117	0.063539
1.2	60	123	0.060561
1.6	-60	150	0.079419
1.6	-40	146	0.082097
1.6	-20	153	0.072058
1.6	0	184	0.056369
1.6	20	136	0.076068
1.6	40	142	0.0812
1.6	60	145	0.093244
2	-60	55	0.13564
2	-40	52	0.11631
2	-20	60	0.094829
2	0	65	0.070398
2	20	56	0.096466
2	40	56	0.10675
2	60	62	0.11001

fx >>

BRISK (B)

Scale	Rotation	NumMatches	TimeSeconds
1.2	-60	3	0.10906
1.2	-40	0	0.044678
1.2	-20	8	0.062733
1.2	0	15	0.041623
1.2	20	8	0.047569
1.2	40	7	0.042743
1.2	60	7	0.031751
1.6	-60	0	0.032588
1.6	-40	4	0.031902
1.6	-20	4	0.031887
1.6	0	8	0.03134
1.6	20	7	0.032409
1.6	40	6	0.031793
1.6	60	3	0.031934
2	-60	11	0.032024
2	-40	6	0.032167
2	-20	13	0.033454
2	0	35	0.030911
2	20	8	0.031761
2	40	5	0.032036
2	60	9	0.031946

fx >>

ORB (O)

Scale	Rotation	NumMatches	TimeSeconds
1.2	-60	491	0.12485
1.2	-40	477	0.054663
1.2	-20	523	0.056914
1.2	0	793	0.032458
1.2	20	497	0.05092
1.2	40	470	0.028914
1.2	60	489	0.026562
1.6	-60	93	0.05174
1.6	-40	92	0.055006
1.6	-20	95	0.052105
1.6	0	123	0.041376
1.6	20	92	0.050502
1.6	40	88	0.057443
1.6	60	117	0.052393
2	-60	248	0.089227
2	-40	244	0.066934
2	-20	280	0.068059
2	0	307	0.054525
2	20	257	0.067216
2	40	239	0.06537
2	60	252	0.072767

fx >>

HARRIS (H)

```
The number of matches is: 0 and the computation time is 0.
```

Scale	Rotation	NumMatches	TimeSeconds
1.2	-60	0	0.092165
1.2	-40	0	0.043674
1.2	-20	0	0.053544
1.2	0	29	0.0526
1.2	20	3	0.051944
1.2	40	0	0.034213
1.2	60	0	0.030544
1.6	-60	0	0.029894
1.6	-40	0	0.02999
1.6	-20	0	0.030632
1.6	0	0	0.034837
1.6	20	0	0.030529
1.6	40	0	0.030116
1.6	60	0	0.030164
2	-60	0	0.030075
2	-40	0	0.034495
2	-20	0	0.030765
2	0	0	0.035554
2	20	0	0.031288
2	40	0	0.030518
2	60	0	0.030786

```
fx >>
```

MSER (M)

Scale	Rotation	NumMatches	TimeSeconds
1.2	-60	0	0.079208
1.2	-40	3	0.049027
1.2	-20	7	0.036441
1.2	0	19	0.017902
1.2	20	11	0.024311
1.2	40	10	0.021464
1.2	60	8	0.010141
1.6	-60	7	0.006844
1.6	-40	6	0.005981
1.6	-20	6	0.00681
1.6	0	8	0.006767
1.6	20	0	0.010279
1.6	40	24	0.007344
1.6	60	0	0.006454
2	-60	6	0.006254
2	-40	6	0.007118
2	-20	14	0.007575
2	0	19	0.006069
2	20	6	0.007118
2	40	8	0.006759
2	60	12	0.006365

fx >>

ORB

- ORB has shown one of the **highest and most consistent numbers of matches** among all descriptors.
- Match counts reached up to **793**, indicating strong performance.
- It maintained high performance **even under large rotation angles**.
- **Conclusion:** ORB is a highly **scale- and rotation-invariant** and effective method.

SURF

- Achieved over **100 matches on average**, demonstrating **strong performance**.
- Performed consistently well under both small and large **rotation and scale changes**.
- **Conclusion:** SURF showed **high invariance**, as expected from theory and prior studies.

SIFT

- SIFT also produced **high match counts** (e.g., 190, 188, 184), indicating strong reliability.
- Demonstrated **robustness to both rotation and scale variations**.
- **Conclusion:** The SIFT algorithm is a **reliable and invariant** method, as commonly reported in the literature.

BRISK and KAZE

- These descriptors showed **moderate performance** overall.
- BRISK yielded around **15–35 matches** in some cases, while KAZE produced **100–150 matches** in optimal conditions.
- However, a **noticeable drop** in performance was observed in more extreme transformation cases.
- **Conclusion:** They offer **moderate invariance**, but may fail in challenging scenarios.

FAST

- Generally produced **low match counts** and **failed in some configurations**.
- Its performance was **inconsistent** under scale and rotation transformations.
- **Conclusion:** FAST is a fast but **low-robustness** descriptor with limited invariance.

MSER and HARRIS

- HARRIS produced matches in only a **few cases**, with most results being zero.
- MSER also showed **very weak performance overall**.
- **Conclusion:** These two descriptors performed **poorly under rotation and scale variations** in this experiment.

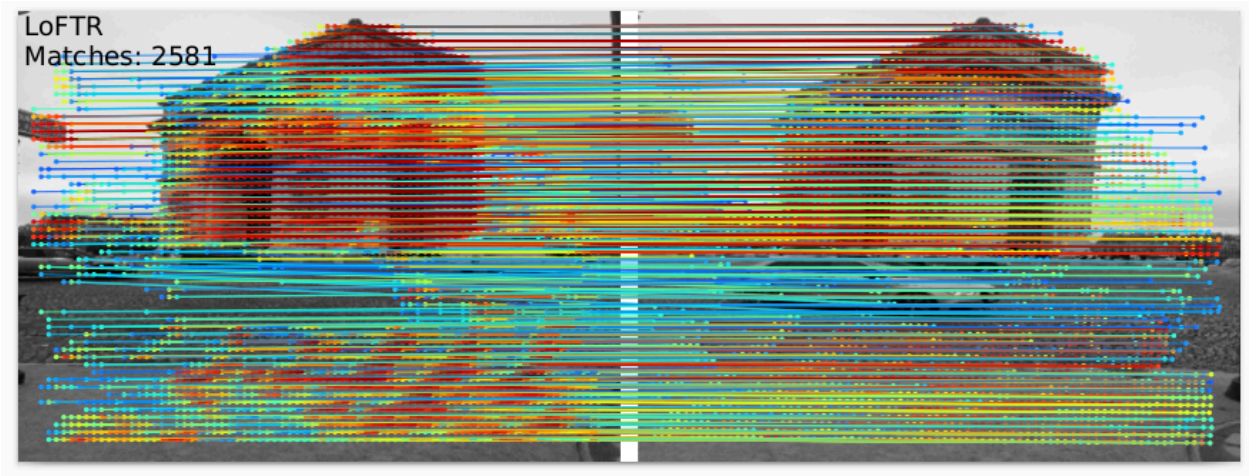
Overall Conclusion

Based on the experimental results:

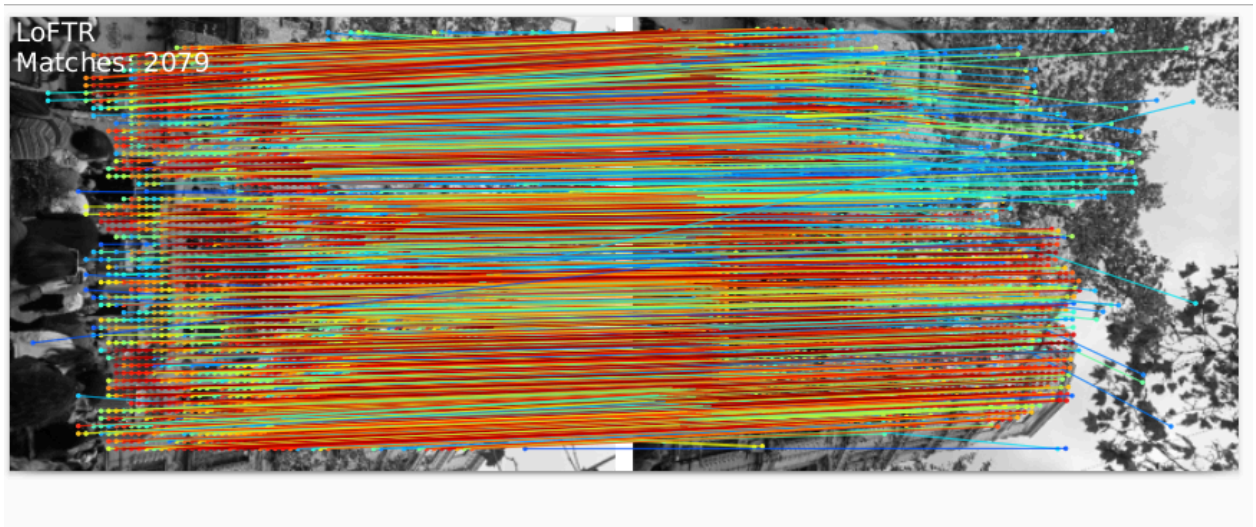
- **ORB, SIFT, and SURF** are the **most robust and invariant descriptors** under scale and rotation changes.
- **KAZE and BRISK** performed at a **moderate level** but showed weakness in certain conditions.
- **FAST, HARRIS, and MSER** were **highly sensitive** to transformations and **did not provide reliable results**.

1.4

In this case, we evaluated LoFTR on two images of the same scene taken from different viewpoints. The model detected **2581 matches**, demonstrating strong capability to handle moderate viewpoint changes. The feature distribution was dense and covered the main structures of the scene, showing that LoFTR successfully captured global correspondences even under perspective distortions. Although small local misalignments may occur when viewpoint shifts are large, the overall matching quality remained high. This confirms that LoFTR is highly robust in scenarios involving viewpoint variations, preserving accurate feature matching across images taken from different angles.

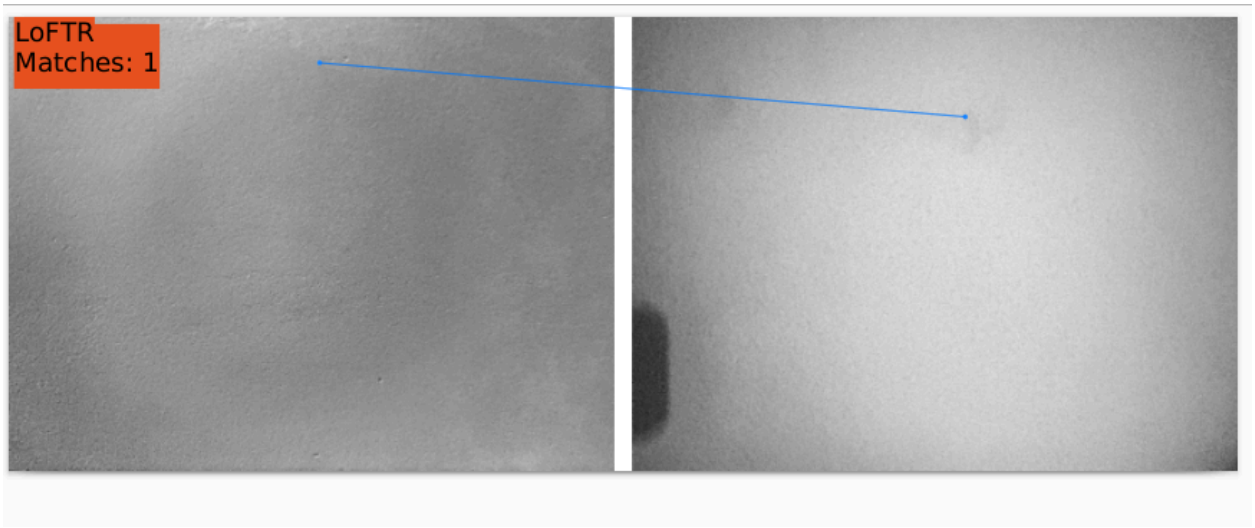

LoFTR
Matches: 2581

Texture-rich scene:



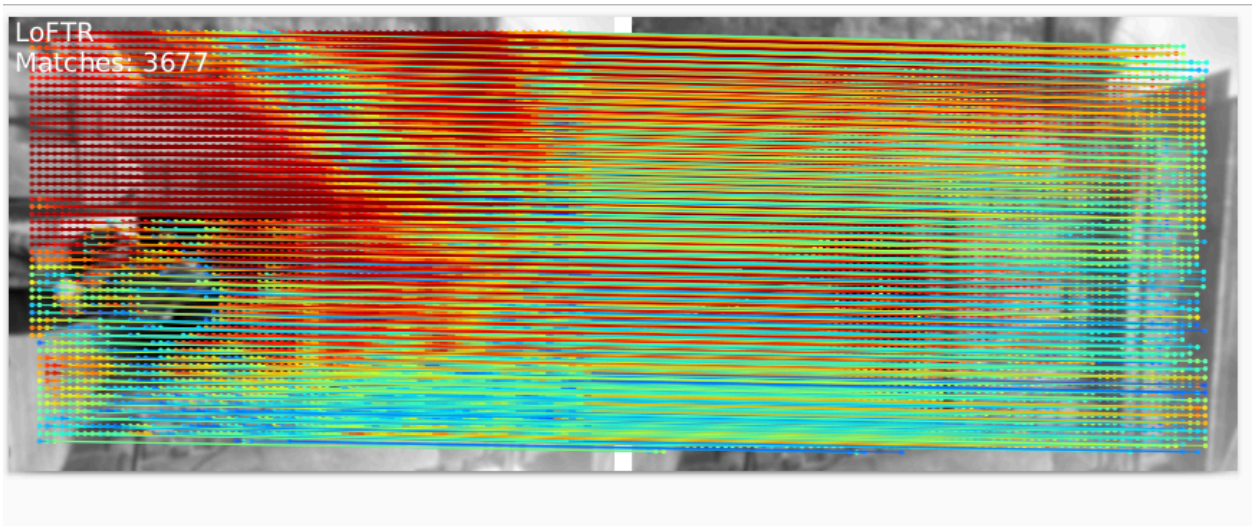
In the first experiment, we evaluated LoFTR on a texture-rich scene. The model successfully detected **2079 matches**, showing strong performance in environments with dense textures and fine details. The feature distribution appeared dense and uniform across the entire image, with matches covering both small and large structures. This indicates that LoFTR effectively captures global correspondences without the need for explicit keypoint detection. Overall, LoFTR demonstrated excellent accuracy and robustness in complex, well-textured environments.

Textureless scene:



In the second case, we evaluated LoFTR on a textureless scene. The model detected only **1 match**, highlighting the difficulty of finding reliable correspondences when strong visual features are absent. The lack of texture and distinctive structures in the image made it extremely challenging for the model to establish meaningful matches. As a result, the feature distribution was almost nonexistent, and matching performance drastically dropped. This shows that while LoFTR is highly effective on textured and complex scenes, it struggles significantly in environments with large uniform areas

Motion blur scene:



In the third case, we evaluated LoFTR on an image pair affected by motion blur. Surprisingly, the model detected **3677 matches**, which is even higher than in the texture-rich scene. The feature distribution remained dense across the image, suggesting that LoFTR could still capture global patterns despite the blur. However, although the number of matches was high, some mismatches and slight inconsistencies were visually noticeable, especially in areas where motion blur was stronger. This shows that while LoFTR maintains a high number of correspondences under motion blur, the geometric accuracy of the matches can slightly decrease. Nevertheless, the model remains robust and produces usable feature sets even in challenging blurred scenarios.