

# Feature Selector and Tracker

## Assignment 2

Gabriele Padovani  
*Artificial Intelligence Systems*  
Trento, Italy  
gabriele.padovani@studenti.unitn.it

### I. INTRODUCTION

To offer a quantitative estimate of the performance of the extractors used, as well as of the tracking, the average frames per second achieved during execution is calculated. This metric, though not perfect gives an idea of how performant an extractor is compared to the others. Since the amount of features found by different algorithms may change, a secondary metric is calculated, keeping track of how many points are returned for each frame.

It is important to stress that a higher number of points found does not imply that the feature extractor is better, as, for example with FAST, tens of thousands of points returned render real-time tracking infeasible.

### II. FEATURE SELECTOR APPROACHES USED

The feature extractors evaluated for this second Computer Vision Assignment are:

- **SIFT**: applies increasingly intense Gaussian filter to the reference image, and looks for maximum and minimum value points, after performing pixel-wise difference.
- **FAST**: corner detector, compares pixel values in a specified radius, and is designed to offer really high performance;
- **ORB**: similar to FAST, but also takes into account the rotation of features;
- **Good Features to Track**: compares the diversity matrices of neighbouring pixels, looking for points with strong eigenvalue, both in the x and y components.

### III. TRACKERS USED

To offer tracking capability, Lukas Kanade Optical Flow was used in conjunction with Good features to track, while SIFT and ORB were both tested with OpenCV's Brute Force matcher and Flann-based matcher.

The former works by refining an initial estimate of a displacement vector, taken from the image at very low resolution, and corrects it by checking it against the same image at increasingly higher definition.

On the other hand, the brute force matcher works by comparing, for each feature in the first image, all the points in the second one, and returning the best match. The Flann-based matcher promises instead higher performance on large sets of features, by learning on a set of descriptors, and utilizing a

more efficient type of distance metric, to quickly exclude false matches.

### IV. PERFORMANCE COMPARISON

As explained in the introductory section, two metrics were calculated, to offer some kind of quantitative evaluation of these algorithms:

#### Feature extractors (Alone)

	Features per Frame	Average FPS
FAST	23323.17	5.43
GFTT	25.0	4.26
ORB	1000.0	5.72
SIFT	3339.3	0.65

#### Feature Tracking

	Features per Frame	Average FPS
GFTT + LKOF	31.08	7.41
ORB + BFMatcher	1000.0	3.53
ORB + Flann Matcher	1000.0	3.42
SIFT + BFMatcher	3471.45	0.56
SIFT + Flann Matcher	3557.33	0.49

Something to note is that the performance of the optical filter is higher than just Good Features to Track alone, because in this test the extractor is called less often, so the tracking algorithm does most of the work. In the test with just point retrieval, the extractor function is called every frame.

### V. TEMPLATE MATCHING EXPERIMENTS

To conclude, some experiments have been done with OpenCV's template matcher, although not very effectively. The matcher seems to find some counters, but returns an offset position. My guess is that this is likely due to the large amount of similar objects in the scene, or to the uniformity in color of the video.

### VI. LINKS

- Github
- Short Output Clips
- Video Presentation