# 2DTracking

UDACITY

## Contents


**1 Theory notes** **2**

**2 Implementation** **2**

2.1 MP.1 Data Buffer Optimization . . . . . 2
2.2 MP.2 Keypoint Detection . . . . . 2
2.3 MP.3 Keypoint Removal . . . . . 4
2.4 MP.4 Keypoint Descriptors . . . . . 4
2.5 MP.5 Descriptor Matching / MP.6 Descriptor Distance Ratio . . . . . 4
2.5.1 Distance ratio . . . . . 5
2.5.2 Cross check . . . . . 5

**3 Performance Evaluation** **5**

3.1 MP.7 Performance Evaluation 1 - Detectors . . . . . 5
3.2 MP.8 Performance Evaluation 2 - Matching . . . . . 7
3.3 MP.9 Performance Evaluation 3 - Timeing . . . . . 8

# 1 Theory notes

# 2 Implementation

## 2.1 MP.1 Data Buffer Optimization

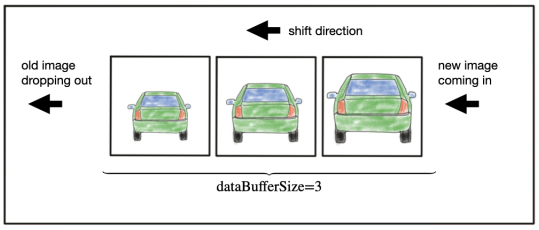



Figure 1: Keypoint count and size distribution - plotted for first image

## 2.2 MP.2 Keypoint Detection

Different keypoint detectors were implemented using the OpenCV module - see figure 2. Classic detectors depend on image gradients and calculate some function of the eigenvalues of the associated hessian. The most famous ones are the Harris or Shi-Thomaso corner detector. The Harris detector was implemented as:

```
 cv::cornerHarris(img, dst, blockSize, apertureSize, k, ...);
# some code for non max suppression
```

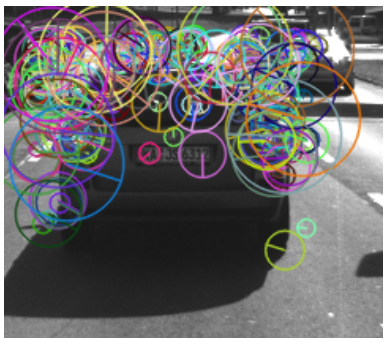
The Shi-Tomaso was implemented as:

```
cv::goodFeaturesToTrack(img, corners, maxCorners, ...);
```

More modern keypoint detectors often dont calculate gradients and rely on binary information. They have a unitfied framework in OpenCV.

```
detector = cv::SIFT::create();
detector->detect(img,keypoints);
```

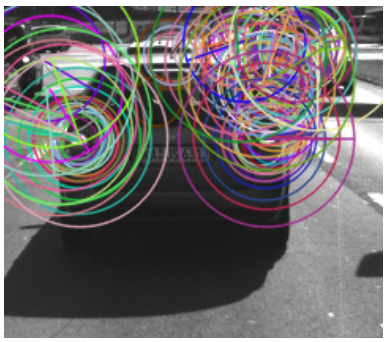
(a) AKAZE



(b) BRISK

(c) FAST

(d) HARRIS



(c) ORB

(d) SHI-TOMASI

(c) SIFT

Figure 2: Visualisation of the detected keypoints on the first frame over the different detectors

## 2.3 MP.3 Keypoint Removal

For a collision detection system, the preceding vehicle is the region-of-interest (ROI). Thus, it's better to remove keypoints outside the pre-defined ROI window.

```
 if (vehicleRect.contains(it->pt))
    continue;

it = keypoints.erase(it);
```

## 2.4 MP.4 Keypoint Descriptors

Implement a variety of keypoint descriptor methods (BRIEF, ORB, FREAK, AKAZE, SIFT). (6c0a14a)
For matching descriptors, add FLANN as alternative to the brutal-force method, and add K-Nearest-Neighbor (KNN) as an addition to the Nearest-Neighbor (NN). (a2bcfea)
Implement the descriptor distance ratio test to filter the bad keypoint matches. (62381f0)

```
extractor = cv::BRISK::create(threshold, octaves, patternScale);
extractor->compute(img, keypoints, descriptors);
```

## 2.5 MP.5 Descriptor Matching / MP.6 Descriptor Distance Ratio

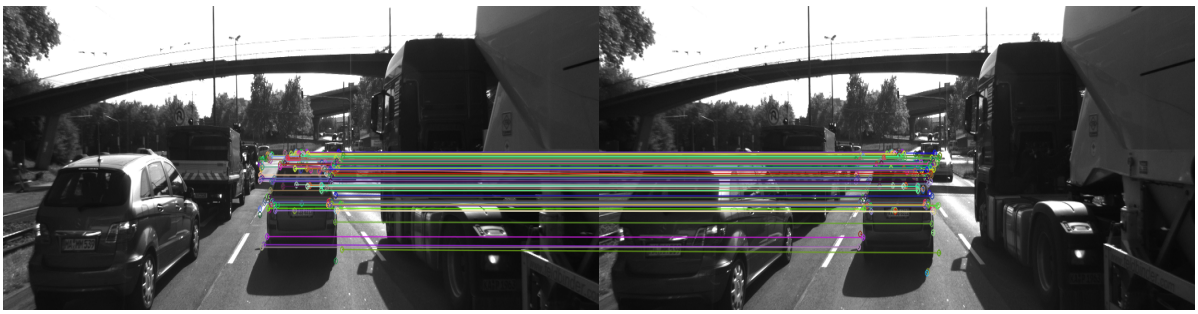


Figure 3: Keypoint count and size distribution - plotted for first image

Norm: Hamming Norms: L1, L2, Hamming,
It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned.

```
        matcher = cv::BFMatcher::create(normType, crossCheck);
```

FLANN BruteForce

```
matcher = cv::DescriptorMatcher::create(cv::DescriptorMatcher::
                                        FLANNBASED);
```

```
matcher->match(descSource, descRef, matches);
```

KNN

```
matcher->knnMatch(descSource, descRef, knn_matches, nMatches);
```

### 2.5.1 Distance ratio

Descriptor distance ratio: Use the K-Nearest-Neighbor matching to implement the descriptor distance ratio test, which looks at the ratio of best vs. second-best match to decide whether to keep an associated pair of keypoints.

### 2.5.2 Cross check

# 3 Performance Evaluation

## 3.1 MP.7 Performance Evaluation 1 - Detectors

*Task*: Count the number of keypoints on the preceding vehicle for all 10 images and take note of the distribution of their neighborhood size. Do this for all the detectors you have implemented.

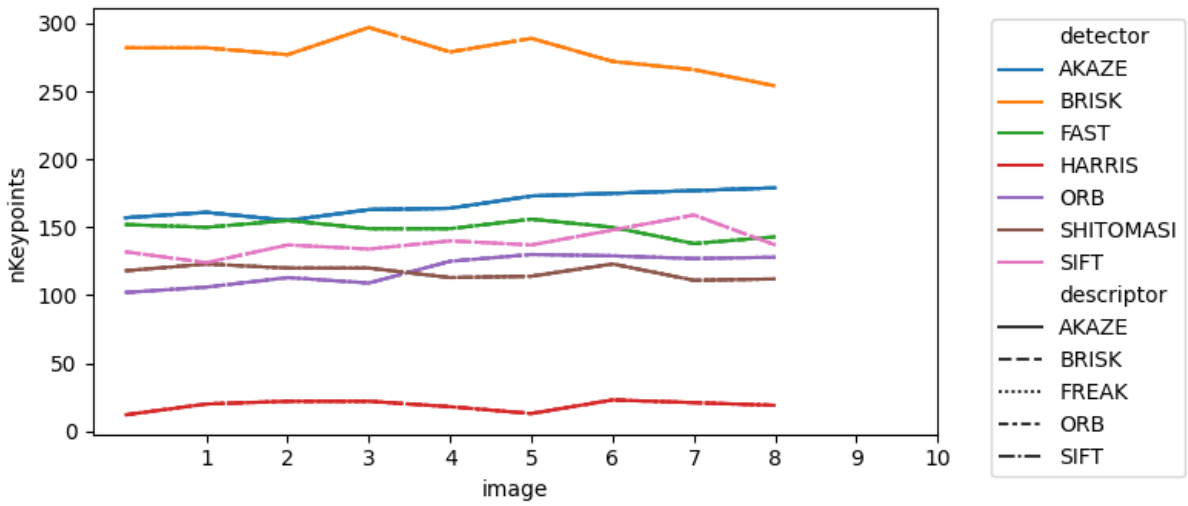



Figure 4: Keypoint count over different images

The figure 5 displays the keypoint count on the preceding and the keypoint size distribution for all implemented detectors. The following detectors do not provide any size information in OpenCV: Fast, Harris and Shi-Thomasi.

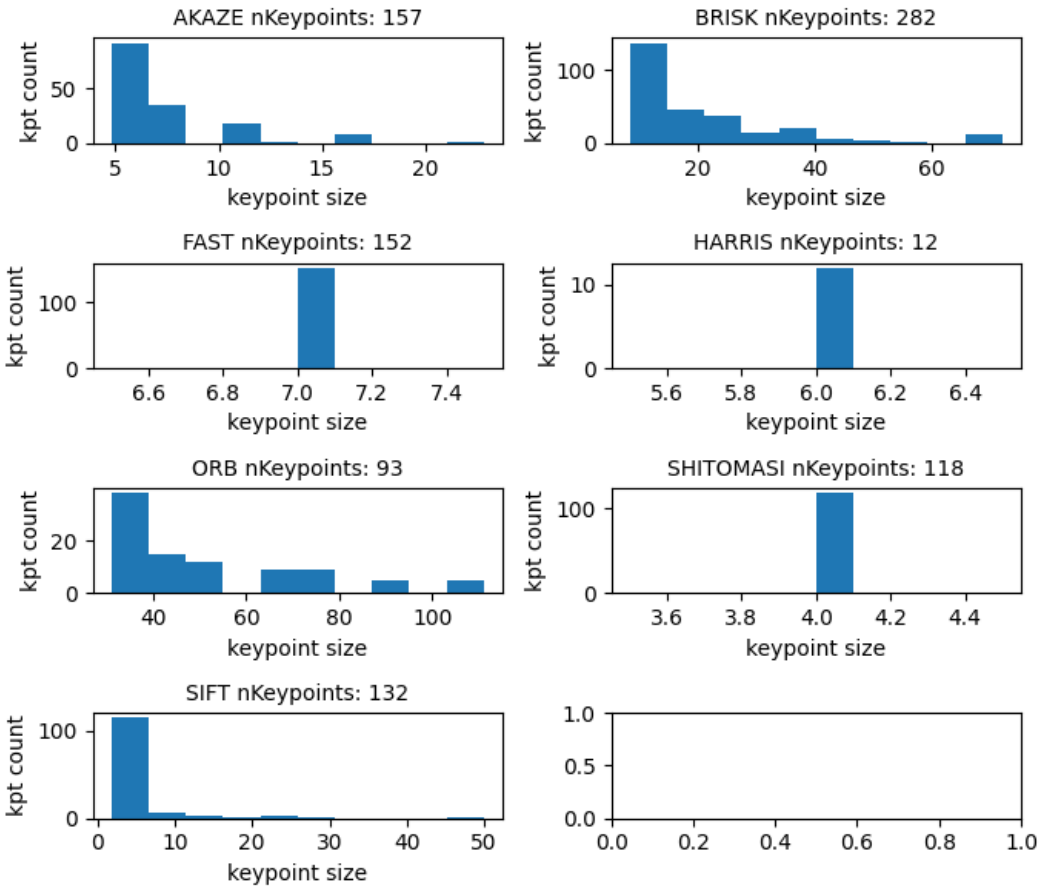



Figure 5: Keypoint count and size distribution - plotted for first image

## 3.2 MP.8 Performance Evaluation 2 - Matching

*Task*: Count the number of matched keypoints for all 10 images using all possible combinations of detectors and descriptors. In the matching step, the BF approach is used with the descriptor distance ratio set to 0.8.

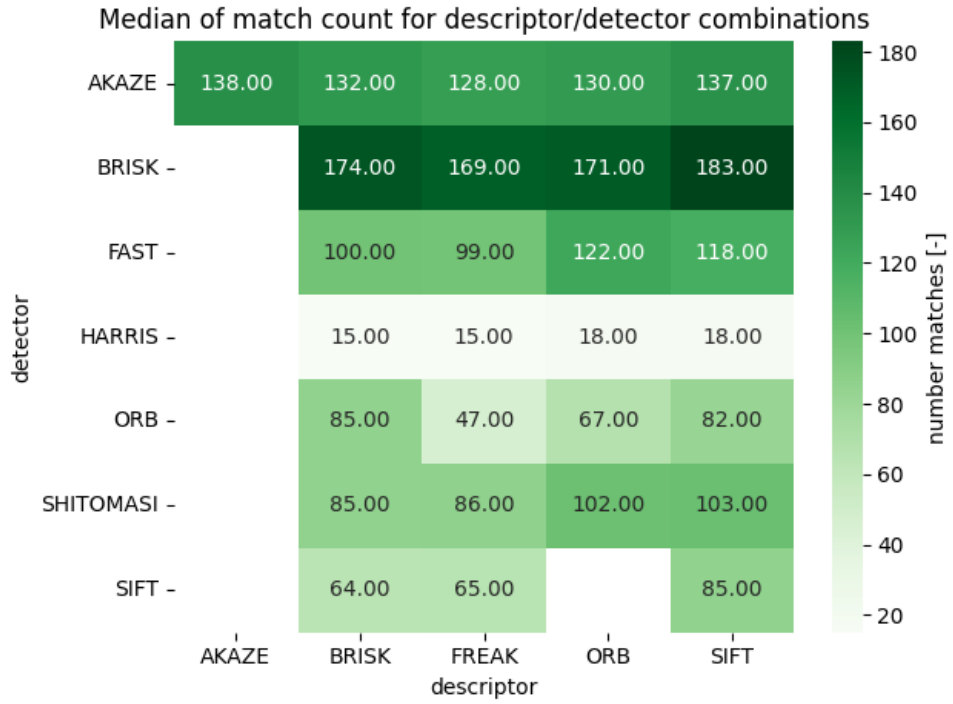



Figure 6: Keypoint count and size distribution - plotted for first image

## 3.3 MP.9 Performance Evaluation 3 - Timeing

*Task*: Log the time it takes for keypoint detection and descriptor extraction. The results must be entered into a spreadsheet and based on this data, the TOP3 detector / descriptor combinations must be recommended as the best choice for our purpose of detecting keypoints on vehicles.

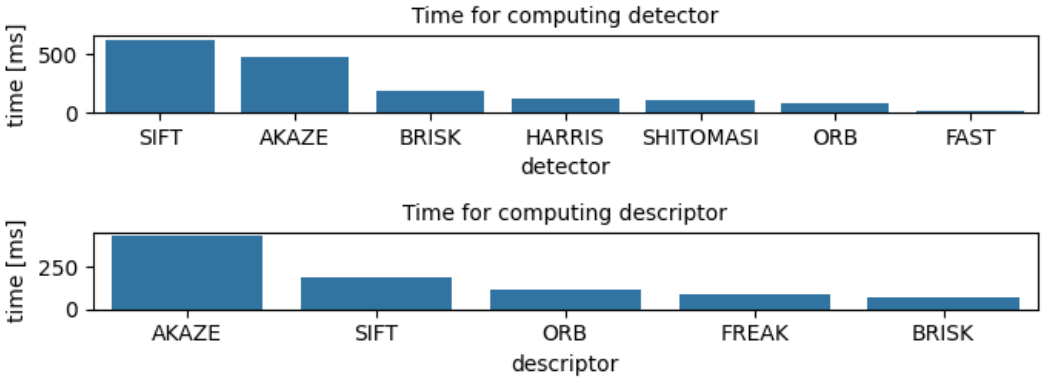



Figure 7: Keypoint count and size distribution - plotted for first image

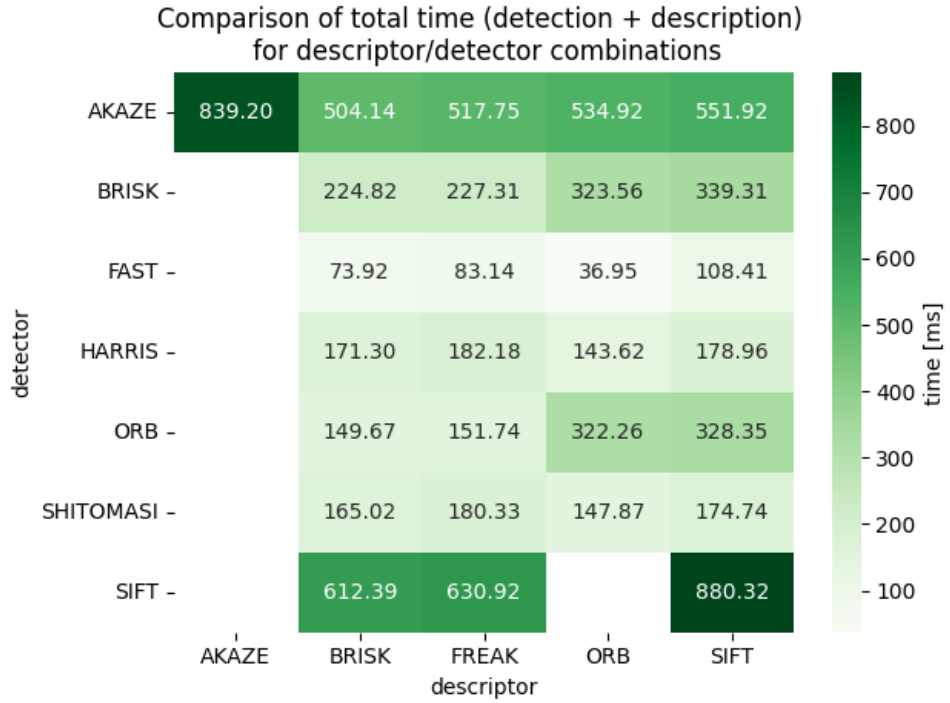



Figure 8: Keypoint count and size distribution - plotted for first image

With FAST detectors (broadly the fastest), the three fastest descriptors were: