FAST: Features from Accelerated Segment Test

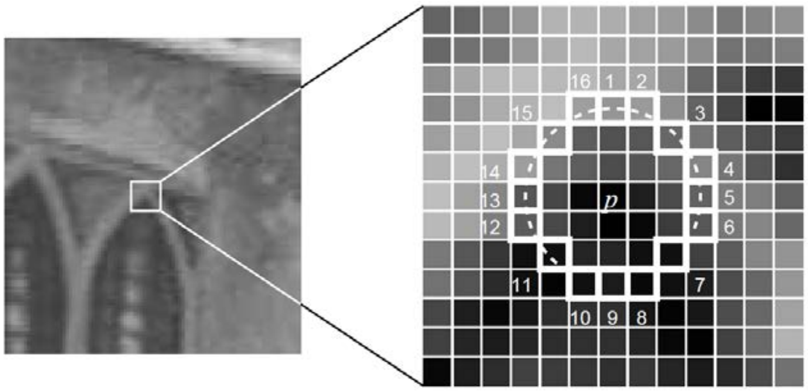
Edward Rosten and Tom Drummond. Machine Learning for High-speed Corner Detection, 2006.

It is several times faster than other existing corner detectors.

But it is not robust to high levels of noise. It is dependent on a threshold

Detect corners using the FAST algorithm

The segment test criterion operates by considering a circle of sixteen pixels around the corner candidate ***p***.



The original detector classifies ***p*** as a corner if there exists a set of ***n*** contiguous pixels in the circle which are all brighter than the intensity of the candidate pixel $\boldsymbol{I_p}$ plus a threshold ***t***, or all darker than $\boldsymbol{I_p - t}$, as illustrated in Figure.

n was chosen to be twelve because it admits a high-speed test which can be used to exclude a very large number of non-corners: the test examines only the four pixels at 1, 5, 9 and 13 (the four compass directions). If ***p*** is a corner then at least three of these must all be brighter than $\boldsymbol{I_p + t}$ or darker than $\boldsymbol{I_p - t}$.

If neither of these is the case, then ***p*** cannot be a corner. The full segment test criterion can then be applied to the remaining candidates by examining all pixels in the circle. This detector in itself exhibits high performance, but there are several weaknesses:

1. The high-speed test does not generalize well for $\boldsymbol{n < 12}$.
2. The choice and ordering of the fast test pixels contains implicit assumptions about the distribution of feature appearance.
3. Knowledge from the first 4 tests is discarded.
4. Multiple features are detected adjacent to one another.

Implementation

Authors present an approach which uses machine learning to address the first three points. The process operates in two stages. In order to build a corner detector for a given ***n***, first, corners are detected from a set of images (preferably from the target application domain) using the segment test criterion for ***n*** and a convenient threshold.

This uses a slow algorithm which for each pixel simply tests all 16 locations on the circle around it. For each location on the circle x ∈ {1..16}, the pixel at that position relative to ***p*** (denoted by p → x) can have one of three states:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \leq I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t \quad \text{(similar)} \\ b, & I_p + t \leq I_{p\to x} \quad \text{(brighter)} \end{cases}$$

Choosing an x and computing $S_{p\to x}$ for all p ∈ P (the set of all pixels in all training images) partitions P into three subsets, P_d, P_s, P_b, where each p is assigned to $P_{S_{p\to x}}$. Let K_p be a boolean variable which is true if p is a corner and false otherwise.

Define a new boolean variable, Kp, which is true if p is a corner and false otherwise.

Use the ID3 algorithm (decision tree classifier) to query each subset using the variable Kp for the knowledge about the true class. It selects the x which yields the most information about whether the candidate pixel is a corner, measured by the entropy of Kp. This is recursively applied to all the subsets until its entropy is zero. The decision tree so created is used for fast detection in other images.

Non-maximal Suppression

Detecting multiple interest points in adjacent locations is another problem. It is solved by using Non-maximum Suppression.

Compute a score function, V for all the detected feature points. V is the sum of absolute difference between p and 16 surrounding pixels values. Consider two adjacent keypoints and compute their V values. Discard the one with lower V value.

OpenCV implementation

C++

```
void FAST(InputArray image, vector<KeyPoint>& keypoints, int threshold, bool nonmaxSuppression=true )
```

Parameters:

- image – grayscale image where keypoints (corners) are detected.
- keypoints – keypoints detected on the image.

- threshold – threshold on difference between intensity of the central pixel and pixels of a circle around this pixel.
- nonmaxSuppression – if true, non-maximum suppression is applied to detected corners (keypoints).