Machine Learning for High Speed Corner Detection

Ed Rosten
Los Alamos National Laboratory*
edrosten@lanl.gov

Tom Drummond
University of Cambridge
twd20@cam.ac.uk

Department of Engineering

Abstract

Goal

A very fast, high quality corner detector.

Contributions

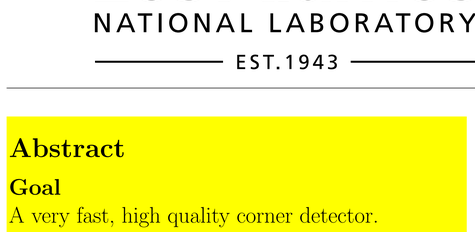
1. The segment-test algorithm for detecting features.
2. Machine learning used to create very efficient implementation: **the FAST feature detector.**
3. Extensive testing of the new detector against existing ones.

Results

1. Extremely fast feature detector: uses **less than 7%** of the available CPU time in live video feeds.
2. Very high quality: FAST outperformed the other detectors in repeatability.
3. **Verifiable results**: the corner detector and testing datasets are available for download.

The Segment-Test Algorithm

If $\geq N$ contiguous pixels in a Bresenham circle of radius r around a centre pixel p are all brighter than p by some threshold or all darker than p by some threshold, then there is a feature at p.



$r = 3$ and $N = 12$. Pixels 11–16 and 1–6 are all brighter than p, so p is a feature.

FAST: Features from Accelerated Segment Test

The segment test algorithm is useful because it can be made to be very computationally efficient.

Consider the case where $N = 12$ and $r = 3$: Pixels 1 and 9 are tested first. If these are similar in intensity to p, then p cannot be a feature. Then 5 followed by 13 are tested, with the same logic applied at each step.

For $N = 12$, at least 12 pixels need to be tested to tell if p is a feature, **but only 2 tests may be required to tell that p is not a feature.**

Problems:

1. $N = 12$ is not the best choice
2. The ordering of questions is not optimal
3. Multiple features are detected adjacent to one another (see the accompanying paper for the solution to this).

Even FASTer

The segment-test algorithm can be implemnted efficiently by learning a decision tree using ID3 [1].

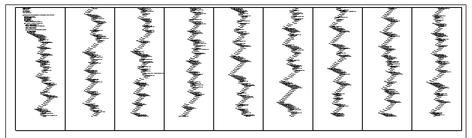
- Every pixel can be classified as a *feature* or *non-feature* (according to the segment-test algorithm).
- Every pixel has 16 attributes, corresponding to the 16 pixels in the ring (for $r = 3$). Attributes are:

  1 if intensity > centre + threshold (i.e. brighter)
  -1 if intensity < centre − threshold (i.e. darker)
  0 otherwise.
- A decision tree can classify a pixel as a feature or non-feature, given the attributes.

The FAST corner detector is then learned as follows:

1. Classify and extract attributes for **all** pixels in a video sequence.
2. For each pixel, ask: **is attribute 'α' equal to 0, 1 or -1?**
   This splits the list of pixels in to 3 sublists:
   list 1: attribute α is 1,
   list 2: attribute α is 0,
   list 3: attribute α is -1.
   The question is chosen to give the greatest entropy reduction.
3. For each sublist, ask a new question which splits each sublist in to a further 3 sublists.
4. Repeat the process for all sublists until the entropy is reduced to zero, i.e. the sublist contains either all features or all non-features.

For speed, the resulting tree of questions is output directly as C code and compiled. This appears as a long list of nested if-else statements:

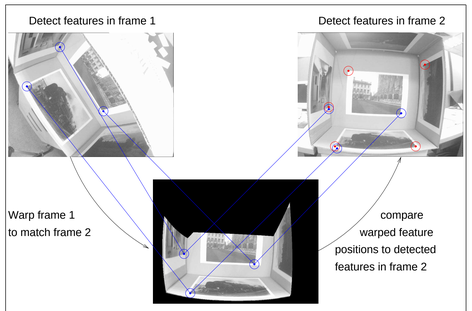


Approximately one third of the C-code generated for the 9 point FAST detector, shown in a very small font. A total of 4235 lines of code are generated.

For $N = 9$ and $r = 3$ only 2.26 questions are required on average to classify a pixel.

Repeatability Evaluation

The same scene viewed from two different positions should yield features which correspond to the same real-world 3D locations[2]. Repeatability is measured as the percentage of features detected from view 1 which are also detected in view 2.



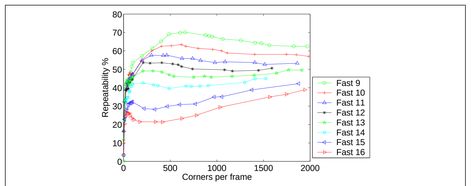
A geometric model is used to compute how to warp frame 1 to frame 2. Lucas-Kanade (with simulated annealing) is used to align all pairs of images smultaneously.

To test the detector thoroughly, several data sets where used with different properties:

Two examplars from each test dataset: (left) planar texture, (centre) geometric corners, (right) bas-relief texture. Note the large changes in viewpoint.

These test repeatability for:
Planar: mostly affine warping.
Geometric: background varying seperately from foreground.
Bas-relief: non-affine warping.

Which FAST is best?



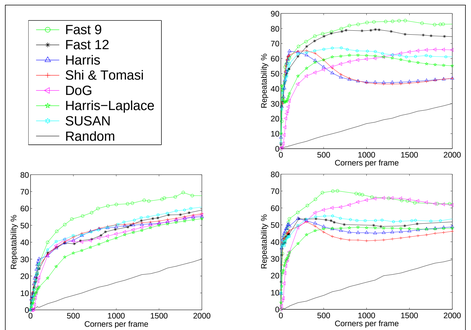
Repeatability for FAST on the bas-relief dataset.

FAST with $N = 9$ is best FAST detector.

Note that FAST with $N < 9$ responds strongly to edges. For further tests, only results for the best FAST ($N = 9$) and the original FAST ($N = 12$) will be shown.

How good is FAST?

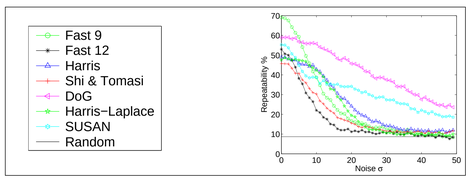
To test FAST's quality, it is compared to a variety of detectors: [3, 4, 5, 6, 7].



Repeatability results for the three datasets.

FAST with $N = 9$ significantly outperforms the other detectors.

Noise Performance



FAST performs quite poorly when the amount of added noise gets large. The small number of pixels accessed leave little opportunity to smooth out noise. Note that the noise is added in addition to considerable amounts of imaging noise.

Speed Evaluation

The following detectors have been compared:

- The tree based FAST-9 and FAST-12
- The original FAST-12
- Efficient DoG and Harris implementations
- The heavily optimized SUSAN reference implementation.

Detector	Opteron 2.6GHz		Pentium III 850MHz	
	ms	%	ms	%
FAST 9	**1.33**	**6.65**	**5.29**	**26.5**
FAST 12	1.34	6.70	4.60	23.0
Old FAST 12	1.59	7.95	9.60	48.0
SUSAN	7.58	37.9	27.5	137.5
Harris	24.0	120	166	830
DoG	60.1	301	345	1280

The best FAST detector (in terms of both repeatability and speed) is over **5 times** faster than the the quickest non-FAST detector.

Conclusions

1. FAST 9 produces very high quality features, as measured by repeatability.
2. Machine learning is used to produce a very efficient implemtation of FAST.

Skeptical?

Get the source code and the test datasets from: http://mi.eng.cam.ac.uk/~er258/work/fast.html

References

[1] Quinlan. *Machine Learning* **1** 81–106, 1986.
[2] Schmid, Mohr, Bauckhage. *IJCV* **37**(2) 151–172, 2000.
[3] Harris, Stephens. *Alvey Vis. Conf.*, 147–151. 1988.
[4] Shi, Tomasi. *CVPR*. 1994.
[5] Smith, Brady. *IJCV* **23**(1) 45–78, 1997.
[6] Mikolajczyk, Schmid. *ECCV*, 128–142. 2002.
[7] Lowe. *IJCV* **60**(2) 91–110, 2004.

*All work was performed at Cambridge University Engineering Department