Faster and better: a machine learning approach to corner detection.

Ed Rosten
What is interest point detection?

- Visually ‘salient’ features.
- Localized in 2D.
- Sparse.
- High ‘information’ content.
- Repeatable between images.

Useful for:
- 2D tracking, 3D tracking, SLAM, object recognition, ...

...
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration
Example: registration

\[
\begin{pmatrix}
  x_1 & y_1 \\
  x_2 & y_2 \\
  x_3 & y_3 \\
  x_4 & y_4 \\
  x_5 & y_5 \\
  \vdots & \vdots
\end{pmatrix}
\quad \quad
\begin{pmatrix}
  u_1 & v_1 \\
  u_2 & v_2 \\
  u_3 & v_3 \\
  u_4 & v_4 \\
  u_5 & v_5 \\
  \vdots & \vdots
\end{pmatrix}
\]
Example: registration

\[
\begin{bmatrix}
    x_1 & y_1 \\
    x_2 & y_2 \\
    x_3 & y_3 \\
    x_4 & y_4 \\
    x_5 & y_5 \\
    \vdots & \vdots \\
\end{bmatrix}
\approx
\begin{bmatrix}
    u_1 & v_1 \\
    u_2 & v_2 \\
    u_3 & v_3 \\
    u_4 & v_4 \\
    u_5 & v_5 \\
    \vdots & \vdots \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
u \\
v
\end{bmatrix}
\approx
W
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]
Typical processing pipeline

Saliency function
Typical processing pipeline

Saliency function

Threshold ↓
Typical processing pipeline

![Image of the processing pipeline]

1. Saliency function
2. Local maxima
3. Threshold
Typical processing pipeline
The segment-test detector
The segment-test detector
The segment-test detector

Contiguous arc of $N$ or more pixels:

- All much brighter than $p$  (brighter than $p + t$).
  
or

- All much darker than $p$  (darker than $p - t$).
The FAST detector (version 1)

FAST—Features from Accelerated Segment Test
The FAST detector (version 1)

- Test pixels 1 and 9
The FAST detector (version 1)

- Test pixels 1 and 9
- Test pixel 4
The FAST detector (version 1)

- Test pixels 1 and 9
- Test pixel 4
- Test pixel 12
The FAST detector (version 1)

- Test pixels 1 and 9
- Test pixel 4
- Test pixel 12
- Perform complete segment test
FAST saliency

- Highest $t$ for which point is a corner.
- Find using bisection over $t$.
  - 8 iterations required.
  - Very small subset of points.
FAST feature detection (version 2)
FAST feature detection (version 2)

- Pixels are either:
  - Much brighter.
FAST feature detection (version 2)

- Pixels are either:
  - Much brighter.
  - Much darker.
FAST feature detection (version 2)

- Pixels are either:
  - Much brighter.
  - Much darker.
  - Similar.
FAST feature detection (version 2)

- Pixels are either:
  - Much brighter.
  - Much darker.
  - Similar.
- Represent ring as a ternary vector.
- Classify vectors using segment test.
Train a classifier

• Decision tree classifiers are very efficient.
• Ask: “What is the state of pixel $x$?”
• Question splits list into 3 sublists.
• Query each sublist.
• Recurse until list contains all features or all non features.
• Choose questions to minimize entropy (ID3).

• Use questions on new feature.
• Works for any $N$. 
Example tree

Node (with offset)  Leaf (non corner)  Leaf (corner)
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Example tree
Output C++ code

A long string of nested if-else statements:

... which continues for 2 more pages.
### How FAST? (very)

<table>
<thead>
<tr>
<th>Detector</th>
<th>Set 1</th>
<th>Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pixel rate (MPix/s)</td>
<td>%</td>
</tr>
<tr>
<td>FAST $n = 9$</td>
<td>188</td>
<td>4.90</td>
</tr>
<tr>
<td>FAST $n = 12$</td>
<td>158</td>
<td>5.88</td>
</tr>
<tr>
<td>Original FAST ($n = 12$)</td>
<td>79.0</td>
<td>11.7</td>
</tr>
<tr>
<td>SUSAN</td>
<td>12.3</td>
<td>74.7</td>
</tr>
<tr>
<td>Harris</td>
<td>8.05</td>
<td>115</td>
</tr>
<tr>
<td>Shi-Tomasi</td>
<td>6.50</td>
<td>142</td>
</tr>
<tr>
<td>DoG</td>
<td>4.72</td>
<td>195</td>
</tr>
</tbody>
</table>

- 3.0GHz Pentium 4
- Set 1: 992 × 668 pixels.
- set 2: 352 × 288 (quarter-PAL) video.
- Percentage budget for PAL, NTSC, DV, 30Hz VGA.
Is it any good?
Repeatability

Is the same real-world 3D point detected from multiple views?

Detect features in frame 1

Warp frame 1 to match frame 2

compare warped feature positions to detected features in frame 2

Detect features in frame 2

Repeat for all pairs in a sequence
FAST-ER: Enhanced Repeatability

- Define feature detector as:
  
  *A decision tree which detects points with a high repeatability.*

- To evaluate repeatability:
  1. Detect features in all frames.
  2. Perform non-maximal suppression.
  3. Compute repeatability.

- Repeatability is a non-convex function of the tree configuration.

- Optimize tree using simulated-annealing.

- Use more offsets than FAST.
FAST-ER: Enhanced Repeatability

- Use more offsets than FAST.
Cost function

1. Higher repeatability is better.
2. Every pixel is a feature $\Rightarrow$ repeatability is 100%.
3. A single detected feature can have 100% repeatability.

Multi-objective optimization needed:

$$\text{cost} = (k_R + R^{-2})(k_N + N^2)(k_S + S^{-2})$$

$R = \text{Repeatability.}$
$N = \text{Number of detected features.}$
$S = \text{Size of tree.}$
Cost function

1. Higher repeatability is better.
2. Every pixel is a feature ⇒ repeatability is 100%.
3. A single detected feature can have 100% repeatability.

Multi-objective optimization needed:

\[
\text{cost} = (k_R + R^{-2})(k_N + N^2)(k_S + S^{-2})
\]

\(R = \text{Repeatability.}\)
\(N = \text{Number of detected features.}\)
\(S = \text{Size of tree.}\)
Cost function

1. Higher repeatability is better.
2. Every pixel is a feature $\Rightarrow$ repeatability is 100%.
3. A single detected feature can have 100% repeatability.

Multi-objective optimization needed:

$$cost = (k_R + R^{-2})(k_N + N^2)(k_S + S^{-2})$$

$R$ = Repeatability.
$N$ = Number of detected features.
$S$ = Size of tree.
Cost function

1. Higher repeatability is better.
2. Every pixel is a feature $\Rightarrow$ repeatability is 100%.
3. A single detected feature can have 100% repeatability.

Multi-objective optimization needed:

$$cost = (k_R + R^{-2})(k_N + N^2)(k_S + S^{-2})$$

$R = $ Repeatability.
$N = $ Number of detected features.
$S = $ Size of tree.
Operations

- Node (with offset): 1
- Leaf (non corner): 2, 3, 7
- Leaf (corner): 9, 5

Brighter

Darker
Operations

‘Similar’ lead nodes are constrained.
Operations

- Brighter
  - Node (with offset): 7
  - Leaf (non corner): 9
- Darker
  - Leaf (corner): 2
  - Leaf (non corner): 3
  - Leaf (corner): 5
Operations

Select a random node. If node is a leaf:
Operations

flip the class (if possible), …
Operations

... or ...

1 Node (with offset)  Leaf (non corner)  Leaf (corner)
Operations

grow a random subtree.
Operations

If node is a non-leaf:

- Node (with offset)
- Leaf (non corner)
- Leaf (corner)
Operations

randomize the offset, ...
Operations

... or ...

1. Node (with offset)
2. Leaf (non corner)
3. Leaf (corner)

9

Leaf (non corner)

Leaf (corner)

Darker

Brighter

Node (with offset)
Operations

replace node with a leaf, ...
Operations

... or ...

1. Node (with offset)
2. Leaf (non corner)
3. Leaf (corner)

Leaf (non corner) Leaf (corner)
Darker Brighter
Node (with offset)
Operations

delete one subtree

1. Node (with offset)
2. Leaf (non corner)
3. Leaf (corner)

Brighter: 9 -> 7
Darker: 2 -> 3
Operations
and replace it with a copy of another subtree.
Reducing the burden on the optimizer

Corners should be invariant to:

- Rotation.
- Reflection.
- Intensity inversion.

There are 16 combinations:

- 4 simple rotations (multiples of 90°).
- 2 reflections.
- 2 intensity inversions.

Run the detector in all combinations.
Iteration scheme

For 100,000 iterations:
1. Randomly modify tree.
2. Output as code.
3. Detect features and perform nonmax suppression.
4. Compute repeatability.
5. Evaluate cost.
6. Keep the modification if:
   \[ e^{\frac{\text{oldcost} - \text{cost}}{\text{temp}}} > \text{rand} \]
7. Reduce the temperature.

Now repeat that 100 times (200 Hours required).
Training data for repeatability

- Change in scale.
- Mostly affine warping.
- Varied texture.
Optimizing FAST-ER for speed

- Tree is applied 16 times at each pixel
- Use repeatability optimized FAST-ER to gather training data:
  1. Detect points in images.
  2. Extract ternary vector of surrounding pixels available to FAST-ER.
- Train single decision tree using ID3.
- Output tree as C code.
Results
Comparisons

- FAST detectors
  - Which $N$ is best?
  - Which of the 200 FAST-ER detectors is best?
- Other detectors
  - Harris.
  - Shi-Tomasi
  - DoG (Difference of Gaussians)
  - Harris-Laplace
  - SUSAN
- What parameters should these detectors use?
Comparisons

- FAST detectors
  - Which $N$ is best?
  - Which of the 200 FAST-ER detectors is best?
- Other detectors
  - Harris.
  - Shi-Tomasi
  - DoG (Difference of Gaussians)
  - Harris-Laplace
  - SUSAN
- What parameters should these detectors use?
Evaluation: Datasets (3D Models)

14 images:

15 images:

8 images:
Evaluation: Homographies

6 images per set:
Results: repeatability curves
## Aggregate results

<table>
<thead>
<tr>
<th>Detector</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST-ER</td>
<td>1313.6</td>
</tr>
<tr>
<td>FAST-9</td>
<td>1304.57</td>
</tr>
<tr>
<td>DoG</td>
<td>1275.59</td>
</tr>
<tr>
<td>Shi &amp; Tomasi</td>
<td>1219.08</td>
</tr>
<tr>
<td>Harris</td>
<td>1195.2</td>
</tr>
<tr>
<td>Harris-Laplace</td>
<td>1153.13</td>
</tr>
<tr>
<td>FAST-12</td>
<td>1121.53</td>
</tr>
<tr>
<td>SUSAN</td>
<td>1116.79</td>
</tr>
<tr>
<td>Random</td>
<td>271.73</td>
</tr>
</tbody>
</table>
Conclusions
What do the results say?

- FAST is surprisingly good.
- FAST-ER is better but slower.
More results
Results: Perspective (box) dataset

Box dataset

Repeatability %

Corners per frame

- DoG
- FAST-12
- FAST-9
- FAST-ER
- Harris
- Harris-Laplace
- Random
- Shi-Tomasi
- SUSAN
Results: Geometric dataset

Maze dataset

- DoG
- FAST-12
- FAST-9
- FAST-ER
- Harris
- Harris-Laplace
- Random
- Shi-Tomasi
- SUSAN
Results: Bas-relief dataset

The graph shows the repeatability percentage of different feature detectors over the number of corners per frame. The detectors include DoG, FAST-12, FAST-9, FAST-ER, Harris, Harris-Laplace, Shi-Tomasi, and SUSAN. The graph indicates that the repeatability increases with the number of corners per frame for all detectors.
Results: Scale and rotation (bark) dataset

Bark dataset

Corners per frame

Repeatability %

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7

0 500 1000 1500 2000

DoG
FAST-12
FAST-9
FAST-ER
Harris
Harris-Laplace
Random
Shi-Tomasi
SUSAN
Results: Blur (bikes) dataset

Bikes dataset

Repeatability %

Corners per frame

DoG
FAST-12
FAST-9
FAST-ER
Harris
Harris-Laplace
Random
Shi-Tomasi
SUSAN
Results: Scale and rotation (boat) dataset
Results: Perspective (graffiti) dataset

![Graph showing repeatability percentage vs. corners per frame for different corner detection methods in the Graffiti dataset.](image-url)
Results: Lighting dataset

Leuven dataset

- DoG
- FAST-12
- FAST-9
- FAST-ER
- Harris
- Harris-Laplace
- Random
- Shi-Tomasi
- SUSAN

Repeatability % vs. Corners per frame
Results: Blur (trees) dataset

Trees dataset

- Repeatability %
- Corners per frame

- DoG
- FAST-12
- FAST-9
- FAST-ER
- Harris
- Harris-Laplace
- Random
- Shi-Tomasi
- SUSAN
Results: JPEG compression dataset

UBC dataset

Repeatability %

Corners per frame

- DoG
- FAST-12
- FAST-9
- FAST-ER
- Harris
- Harris-Laplace
- Random
- Shi-Tomasi
- SUSAN
Results: Perspective (wall) dataset

![Graph showing the repeatability of corner detection methods over corners per frame for the Wall dataset.](chart.png)