Optimized corner detection and object detection

Edward Rosten

Damian Eads, David Helmbold, Reid Porter, Tom Drummond
Optimizing the right thing

Two examples:

1. Corner detection
2. Object detection

What are they and how do you optimize them?
What is corner detection?

Useful for:

- 2D tracking, 3D tracking, SLAM, object recognition, etc.
- Visually ‘salient’ features.
- Localized in 2D.
- Sparse.
- High ‘information’ content.
- Repeatable between images.

Edward Rosten, Reid Porter, Tom Drummond
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 $p - t$).
FAST feature detection

$p$
FAST feature detection

- Pixels are either:
  - Much brighter.
FAST feature detection

- Pixels are either:
  - Much brighter.
  - Much darker.
FAST feature detection

- Pixels are either:
  - Much brighter.
  - Much darker.
  - Similar.
FAST feature detection

- 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 $N$. 
Output C++ code

A long string of nested if-else statements:

```cpp
for(y = 3; y < i.size().y - 3; y++)
    for(x=0; x < i.size().x;x++)
    {
        centre = image[y][x];
        if(image[y-3][x] > centre + threshold)
            if(image[y+3][x+1] > centre + threshold)
                if(...
                    else
                    ...
```
## How FAST? (very)

<table>
<thead>
<tr>
<th>Detector</th>
<th>Set 1</th>
<th></th>
<th>Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pixel rate (MPix/s)</td>
<td>%</td>
<td>MPix/s</td>
</tr>
<tr>
<td>FAST $n = 9$</td>
<td>188</td>
<td>4.90</td>
<td>179</td>
</tr>
<tr>
<td>FAST $n = 12$</td>
<td>158</td>
<td>5.88</td>
<td>154</td>
</tr>
<tr>
<td>Original FAST ($n = 12$)</td>
<td>79.0</td>
<td>11.7</td>
<td>82.2</td>
</tr>
<tr>
<td>SUSAN</td>
<td>12.3</td>
<td>74.7</td>
<td>13.6</td>
</tr>
<tr>
<td>Harris</td>
<td>8.05</td>
<td>115</td>
<td>7.90</td>
</tr>
<tr>
<td>Shi-Tomasi</td>
<td>6.50</td>
<td>142</td>
<td>6.50</td>
</tr>
<tr>
<td>DoG</td>
<td>4.72</td>
<td>195</td>
<td>5.10</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

Detect features in frame 2

Warp frame 1 to match frame 2

Compare warped feature positions to detected features in frame 2

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

- Define feature detector as:
  \textit{A decision tree which detects points with a high repeatability.}

- To evaluate repeatability:
  1. Detect features in all frames.
  2. Compute repeatability.

- That is hard to optimize!
  Optimize tree using simulated-annealing.

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

- Use more pixels 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:

\[
\text{cost} = (1 + w_r R^{-2})(1 + w_n N^2)(1 + w_s S^2)
\]

\(R\) = 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:

$$cost = (1 + w_r R^{-2})(1 + w_n N^2)(1 + w_s S^2)$$

$R$ = 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:

\[
\text{cost} = (1 + w_r R^{-2})(1 + w_n N^2)(1 + w_s S^2)
\]

- \(R\) = 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:

$$cost = (1 + w_r R^{-2})(1 + w_n N^2)(1 + w_s S^2)$$

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

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

Brighter  Darker
Operations

‘Similar’ leaf nodes are constrained.
Operations

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

Brighter

Darker
Operations

Select a random node. If node is a leaf:

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

flip the class (if possible), ...
Operations

... or ...

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

grow a random subtree.
Operations

If node is a non-leaf:

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

randomize the offset, ...
Operations

... or ...

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

Brighter

Darker
Operations
replace node with a leaf, ...
Operations

... or ...

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

delete one subtree

![Tree diagram with nodes and leaves labeled as Brighter or Darker, and node with offset.]

1. Node (with offset)
2. Leaf (non corner)
3. Leaf (corner)
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. Compile directly to machine code.
3. Detect features.
4. Compute repeatability.
5. Evaluate cost.
6. Keep the modification if:
   \[ e^{\frac{\text{oldcost} - \text{cost}}{\text{temp}}} > \text{rand}(0,1) \]
7. Reduce the temperature.

Now repeat that 200 times.
Training data for repeatability

- Change in scale.
- Mostly affine warping.
- Varied texture.
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?
Results: repeatability curves
## Aggregate results

<table>
<thead>
<tr>
<th>Detector</th>
<th>AUR</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>

![Graph showing corners per frame vs repeatability](image)

**AUR**
## 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>FAST-ER</td>
<td>75.4</td>
<td>12.2</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: $992 \times 668$ pixels.
- Set 2: $352 \times 288$ (quarter-PAL) video.
- Percentage budget for PAL, NTSC, DV, 30Hz VGA.
Conclusions on FAST

- FAST is very fast
  - And very repeatable.
- FAST-ER is even more repeatable.
- Source code is available:
  
  http://mi.eng.cam.ac.uk/~er258/work/fast.html
Object Detection
Object detection

Target detection

Traffic analysis

Damian Eads, Edward Rosten, David Helmbold
Object detection: difficulties

Which ones are cars?

- Problem is unstructured
  \[ \text{Image} \rightarrow \{(x_1, y_1), (x_2, y_2), \cdots\} \]
- Number of objects unknown \textit{a priori}
- Not a fixed set of labels
What is a detection anyway?

1. Not pixels! 50% of pixels on all of the objects is not the same as all of the pixels on 50% of the objects.
2. It depends...
Measures of performance

- Identification:
  - Within boundary
- Tracking
  - Nearby, but with unique assignment
- Counting
  - Unique assignment
  - Within radius of sliding window
Measures of performance

- Identification:
  - Within boundary

- Tracking
  - Nearby, but with unique assignment

- Counting
  - Unique assignment
  - Within radius of sliding window
Measures of performance

• Identification:
  ○ Within boundary

• Tracking
  ○ Nearby, but with unique assignment

• Counting
  ○ Unique assignment
  ○ Within radius of sliding window
Grammar guided features

Feature → Threshold > 0.29 → Regions 50 pix

Post-process

Weak classification

System layout

Regions 50 pix
System layout

Grammar guided features

Feature  →  Threshold > 0.29  →  Regions 50 pix

Feature  →  Threshold > 1.03  →  Regions 17 pix

Feature  →  Threshold <= 2.9  →  Median 11 pix

Post-process

Weak classification

∑
System layout

Grammar guided features

- Feature
  - Threshold > 0.29
  - Regions 50 pix
- Feature
  - Threshold > 1.03
  - Regions 17 pix
- Feature
  - Threshold <= 2.9
  - Median 11 pix

Post-process

Weak classification

- Regions
- Threshold
- Median

Detection

Scoring

- True positive
- False positive
- True positive

A ROC
Feature extraction

- Features are small image processing programs.
- Stochastic generative grammar for making programs
- Composed of basic operators: morphology, percentiles, Gabor filters, Haar-like features, edges, ...
- Combined using: addition, subtraction, multiplication, sigmoiding, ...

$I \times \text{Erode} \left( \text{“Ellipse”}, \frac{2\pi}{5}, 5.1, 1.2 \right)$
Feature grammars

- A grammar consists of *productions*
  \[ P \rightarrow A | B \]
- Productions are expanded stochastically:
  - \( P \) can be turned into \( A \) or \( B \)
  - \( P \) is *non-terminal*
  - \( A \) and \( B \) are *terminal*
  - Non-terminals expanded until only terminals remain
  - Expansion rules have domain expertise built in
  - Intelligent sampling of feature space
Example

\[
\begin{align*}
\text{Feature}(x) &\rightarrow \text{Binary}(\text{Unary}(x), \text{Unary}(x)) \mid \text{Unary}(x) \\
\text{Unary}(x) &\rightarrow x \mid \text{Erode}(x, \text{RandomSE}()) \\
\text{Binary}(x, y) &\rightarrow \text{Add}(x, y) \mid \text{Multiply}(x, y) \\
\text{RandomSE}() &\rightarrow \text{Ellipse}(\mathcal{U}(0, \pi), \mathcal{U}(1, 10), \mathcal{U}(1, 10))
\end{align*}
\]

\[f(x) = \text{Feature}(x)\]
Example

\[
\text{Feature}(x) \rightarrow \text{Binary} (\text{Unary}(x), \text{Unary}(x)) \mid \text{Unary}(x)
\]

\[
\text{Unary}(x) \rightarrow x \mid \text{Erode}(x, \text{RandomSE}(\))
\]

\[
\text{Binary}(x, y) \rightarrow \text{Add}(x, y) \mid \text{Multiply}(x, y)
\]

\[
\text{RandomSE()} \rightarrow \text{Ellipse} (\mathcal{U}(0, \pi), \mathcal{U}(1, 10), \mathcal{U}(1, 10))
\]

\[
f(x) = \text{Binary} (\text{Unary}(x), \text{Unary}(x))
\]
Example

Feature\( (x) \) $\rightarrow$ Binary(Unary\( (x) \), Unary\( (x) \)) $|$ Unary\( (x) \)

Unary\( (x) \) $\rightarrow$ \( x \) $|$ Erode\( (x, \text{RandomSE}()) \)

Binary\( (x, y) \) $\rightarrow$ Add\( (x, y) \) $|$ Multiply\( (x, y) \)

RandomSE() $\rightarrow$ Ellipse\( (\mathcal{U}(0, \pi), \mathcal{U}(1, 10), \mathcal{U}(1, 10)) \)

\[ f(x) = \text{Binary(Unary}(x), \text{Erode}(x, \text{RandomSE})()) \]
Example

Feature$(x) \rightarrow \text{Binary}(\text{Unary}(x), \text{Unary}(x)) \mid \text{Unary}(x)$

Unary$(x) \rightarrow x \mid \text{Erode}(x, \text{RandomSE}())$

Binary$(x, y) \rightarrow \text{Add}(x, y) \mid \text{Multiply}(x, y)$

RandomSE() $\rightarrow \text{Ellipse}(\mathcal{U}(0, \pi), \mathcal{U}(1, 10), \mathcal{U}(1, 10))$

\[
f(x) = \text{Binary}(\text{Unary}(x), \text{Erode}(x, \text{Ellipse}(\frac{2\pi}{5}, 5.1, 1.2))))
\]
Example

\[
\begin{align*}
\text{Feature}(x) & \rightarrow \text{Binary}(\text{Unary}(x), \text{Unary}(x)) \mid \text{Unary}(x) \\
\text{Unary}(x) & \rightarrow x \mid \text{Erode}(x, \text{RandomSE}()) \\
\text{Binary}(x, y) & \rightarrow \text{Add}(x, y) \mid \text{Multiply}(x, y) \\
\text{RandomSE}() & \rightarrow \text{Ellipse}(\mathcal{U}(0, \pi), \mathcal{U}(1, 10), \mathcal{U}(1, 10))
\end{align*}
\]

\[
f(x) = \text{Binary}(x, \text{Erode}(x, \text{Ellipse}(\frac{2\pi}{5}, 5.1, 1.2)))
\]
**Example**

\[
\begin{align*}
\text{Feature}(x) & \rightarrow \text{Binary}(\text{Unary}(x), \text{Unary}(x)) \mid \text{Unary}(x) \\
\text{Unary}(x) & \rightarrow x \mid \text{Erode}(x, \text{RandomSE}()) \\
\text{Binary}(x, y) & \rightarrow \text{Add}(x, y) \mid \text{Multiply}(x, y) \\
\text{RandomSE}() & \rightarrow \text{Ellipse}(\mathcal{U}(0, \pi), \mathcal{U}(1, 10), \mathcal{U}(1, 10))
\end{align*}
\]

\[
f(x) = \text{Multiply}(x, \text{Erode}(x, \text{Ellipse}(\frac{2\pi}{5}, 5.1, 1.2)))
\]
Example

Feature$(x) \rightarrow$ Binary(Unary$(x)$, Unary$(x)$) $|$ Unary$(x)$
Unary$(x) \rightarrow$ $x$ $|$ Erode$(x, \text{RandomSE}())$
Binary$(x, y) \rightarrow$ Add$(x, y)$ $|$ Multiply$(x, y)$
RandomSE() $\rightarrow$ Ellipse$(\mathcal{U}(0, \pi), \mathcal{U}(1, 10), \mathcal{U}(1, 10))$

\[
f(x) = \text{Multiply}(x, \text{Erode}(x, \text{Ellipse}(\frac{2\pi}{5}, 5.1, 1.2)))
\]
Some random features
Turning pixels into objects

- Large local maxima
  Choice of pre-smoothing radius
- KDE on large local maxima
  Also kernel size
- Connected components
  Choice of threshold

Optimize over data not used for boosting.
Results
Results: Target detection

- Grammar w/ PP
- Grammar w/o PP
- Haar w/ PP
- Random

Detection rate vs Mean false positive rate for different methods.
Results: Tracking

- Grammar w/ PP
- Grammar w/o PP
- Haar w/ PP
- Random

Detection rate vs. Mean false positive rate graphs for different methods.

- LLM
- CC
- KDE
- Random
Conclusions

- New features: Grammar-guided features
- Training against scoring measures

http://users.soe.ucsc.edu/~eads/software.shtml
More results
Sensitivity to $w_i$

![Graph showing probability density distribution for different categories: All, Best, Worst, Median, against Mean AUR score.](graph.png)
Results: Perspective (box) dataset

Box dataset

Repeatability vs. Corners per frame for various corner detection methods:
- DoG
- FAST-12
- FAST-9
- FAST-ER
- Harris
- Harris-Laplace
- Random
- Shi-Tomasi
- SUSAN
Results: Bas-relief dataset

![Graph showing repeatability vs. corners per frame for various corner detectors in the Bas-relief dataset. The x-axis represents corners per frame, ranging from 0 to 2000, and the y-axis represents repeatability, ranging from 0 to 1. The corners per frame are: DoG, FAST-12, FAST-9, FAST-ER, Harris, Harris-Laplace, Random, Shi-Tomasi, and SUSAN. Each detector has a distinct marker and color for easy identification.](image-url)
Results: Scale and rotation (bark) dataset

Bark dataset

Repeatability

Corners per frame

- 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

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

![Graph showing repeatability vs. corners per frame for different corner detectors on the boat dataset. The x-axis represents corners per frame, and the y-axis represents repeatability. Different corner detectors are represented by distinct line styles and markers. From top to bottom, the lines represent DoG, FAST-12, FAST-9, FAST-ER, Harris, Harris-Laplace, Shi-Tomasi, Random, and SUSAN.]
Results: Perspective (graffiti) dataset

![Graph showing results of different corner detection methods on the Graffiti dataset. The x-axis represents the number of corners per frame, ranging from 0 to 2000. The y-axis represents repeatability, ranging from 0 to 1. Different methods are represented by distinct markers and colors.]
Results: Lighting dataset

Leuven dataset

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

![Graph showing repeatability vs corners per frame for different feature detection algorithms on the UBC dataset. The graph compares algorithms like DoG, FAST-12, FAST-9, FAST-ER, Harris, Harris-Laplace, Random, Shi-Tomasi, and SUSAN.](image-url)
Results: Perspective (wall) dataset
Evaluation: Datasets (3D Models)

14 images:

15 images:

8 images:
Evaluation: Homographies

6 images per set: