Optimized corner detection and object detection

Edward Rosten

UNIVERSITY OF CAMBRIDGE

Los Alamos NATIONAL LABORATORY EST. 1943

THE UNIVERSITY OF CALIFORNIA 1868 LET THERE BE LIGHT

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?

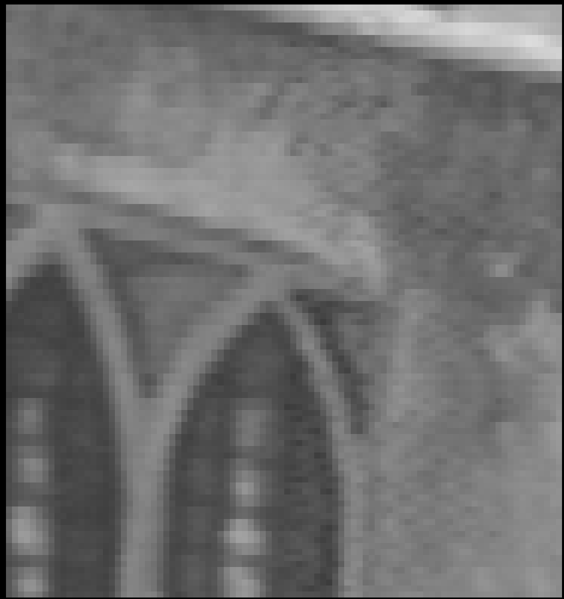
Useful for:

- 2D tracking, 3D tracking, SLAM, object recognition, etc.

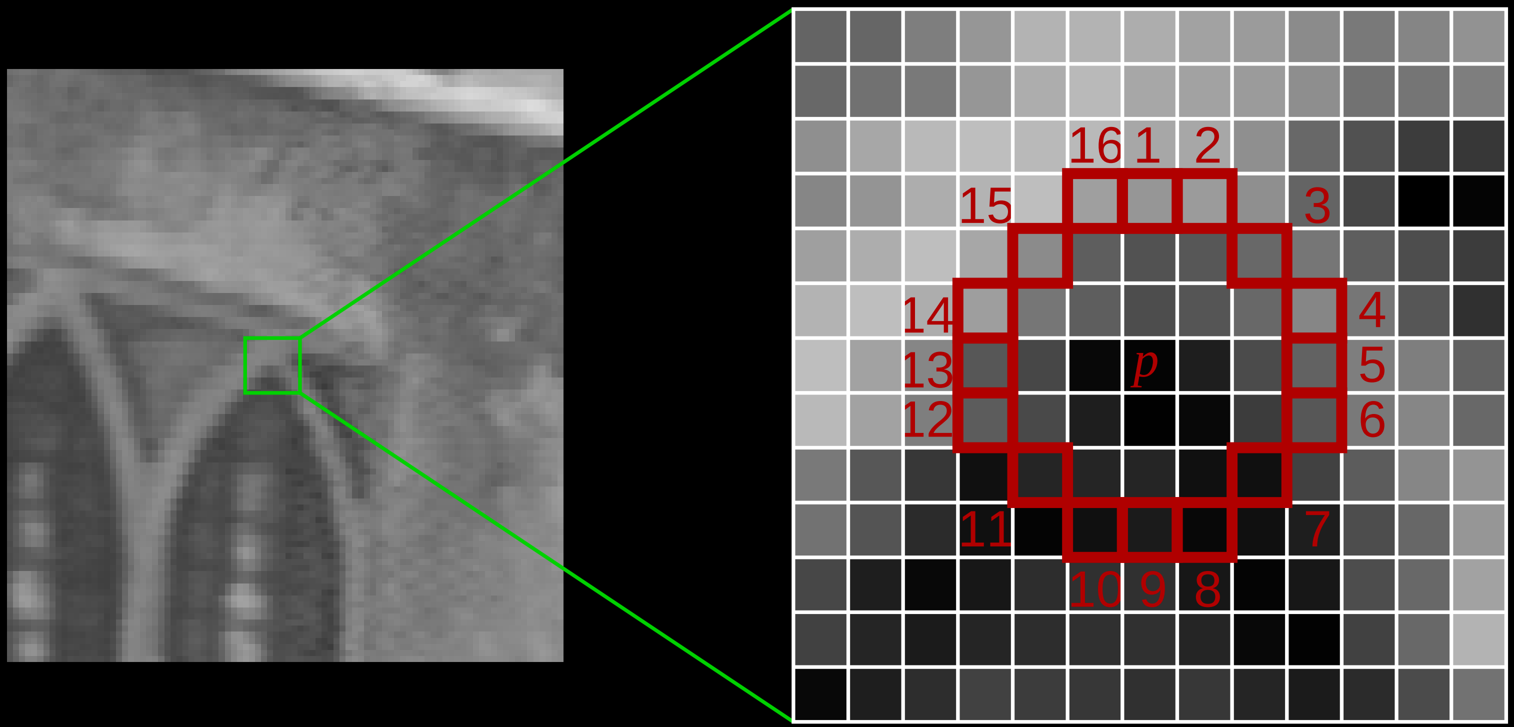
- Visually ‘salient’ features.
- Localized in 2D.
- Sparse.
- High ‘information’ content.
- Repeatable between images.

Edward Rosten, Reid Porter, Tom Drummond

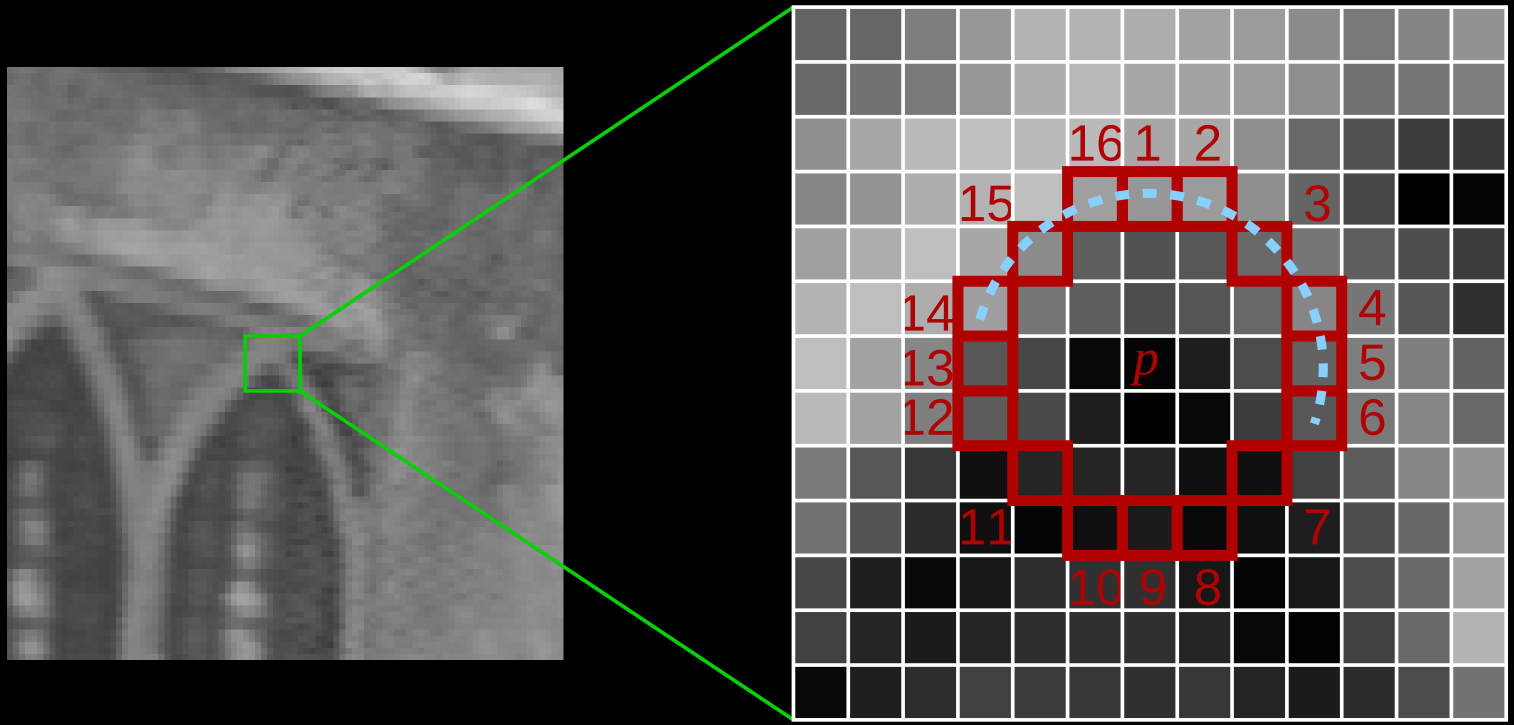
The segment-test detector



The segment-test detector



The segment-test detector



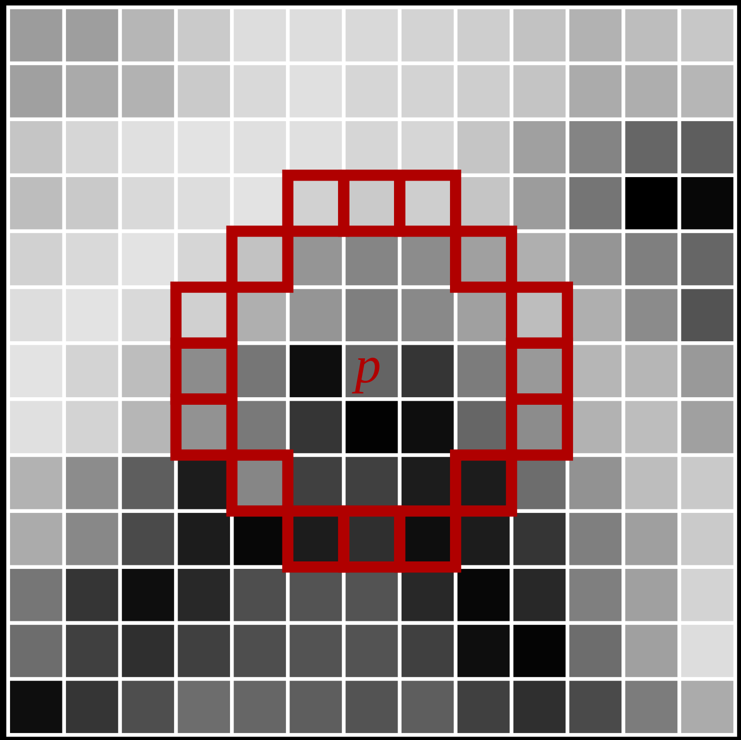
Contiguous arc of N or more pixels:

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

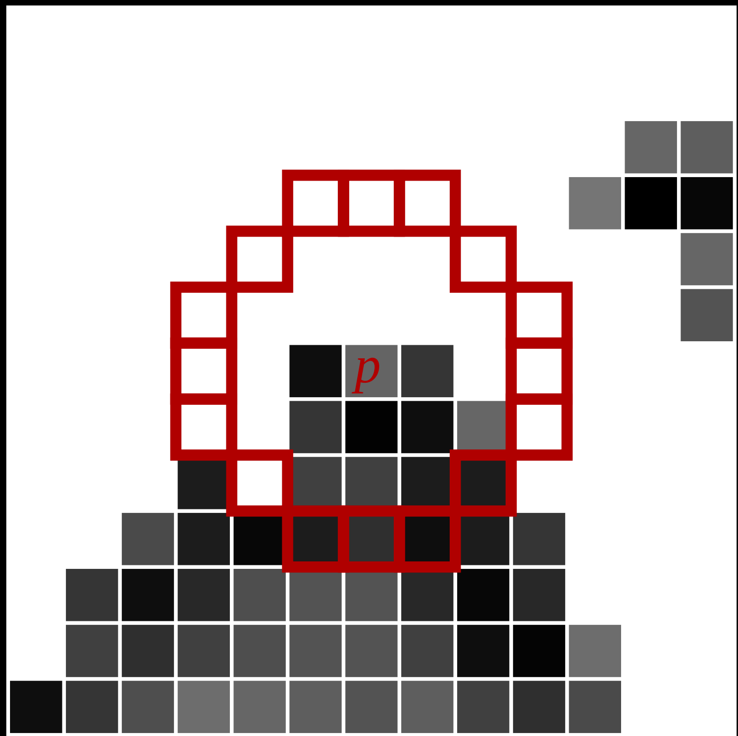
or

- All much darker than p (darker than $p - t$).

FAST feature detection

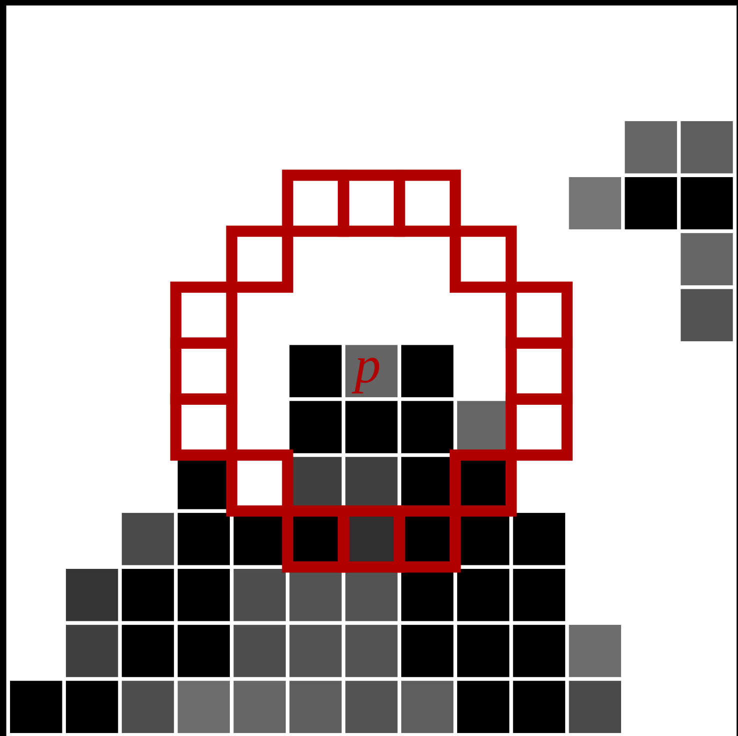


FAST feature detection



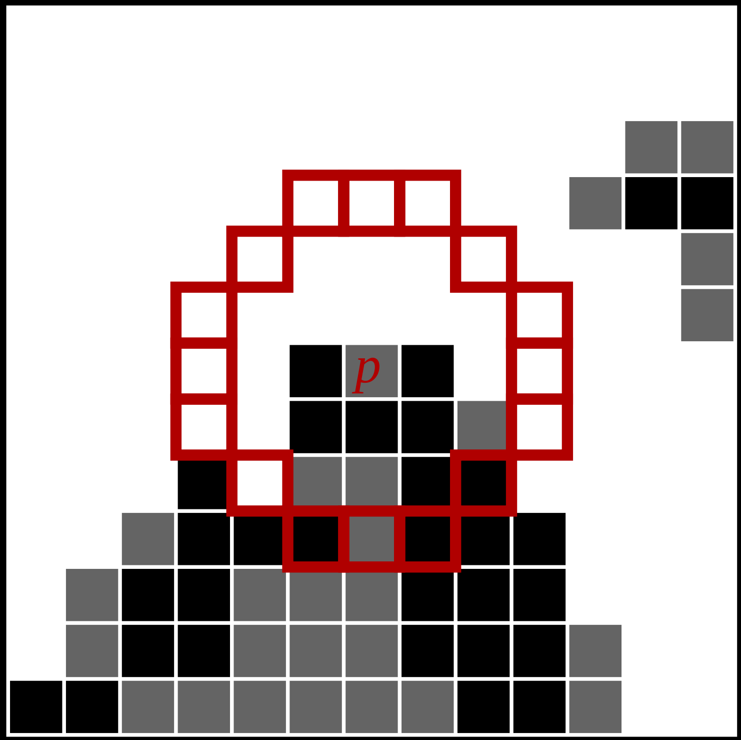
- Pixels are either:
  - Much brighter.

FAST feature detection



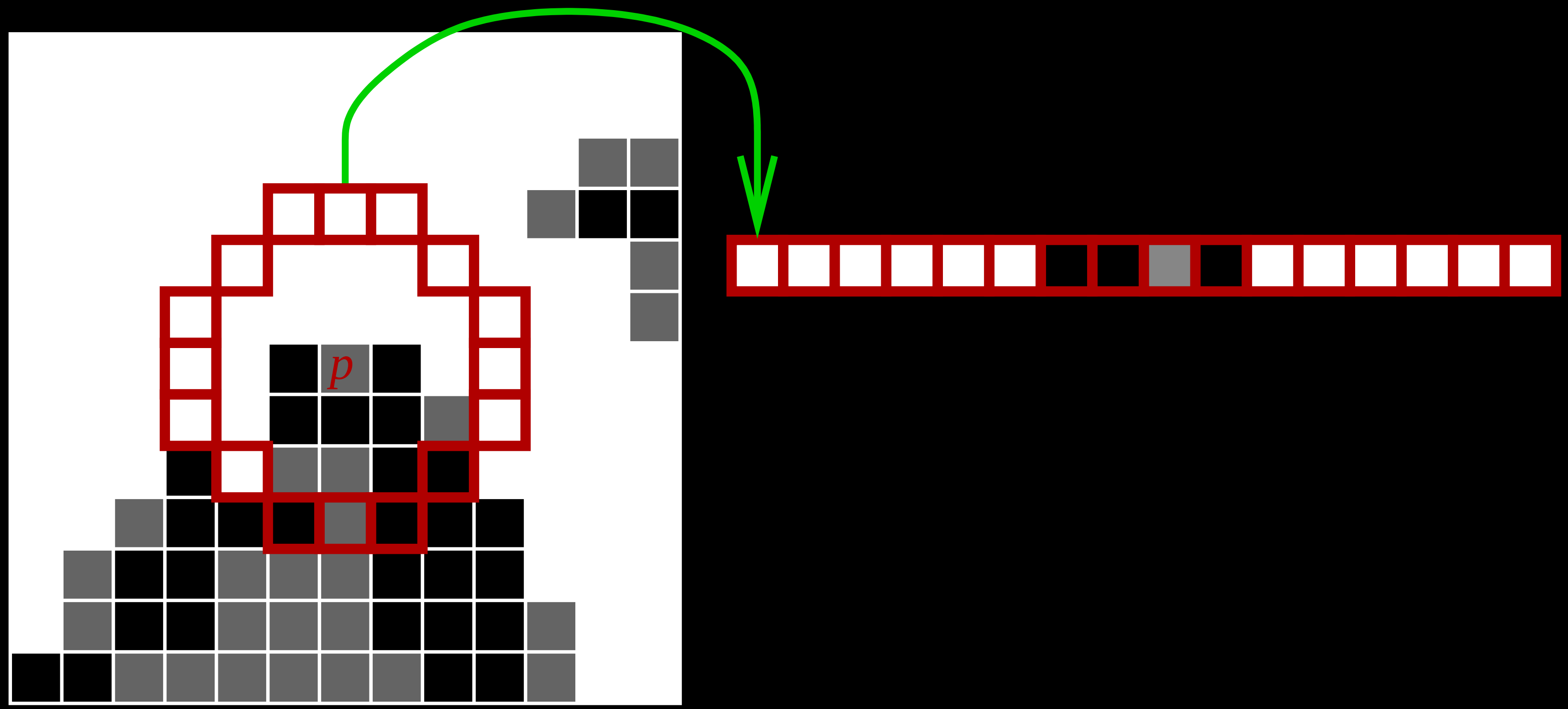
- Pixels are either:
  - Much brighter.
  - Much darker.

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.

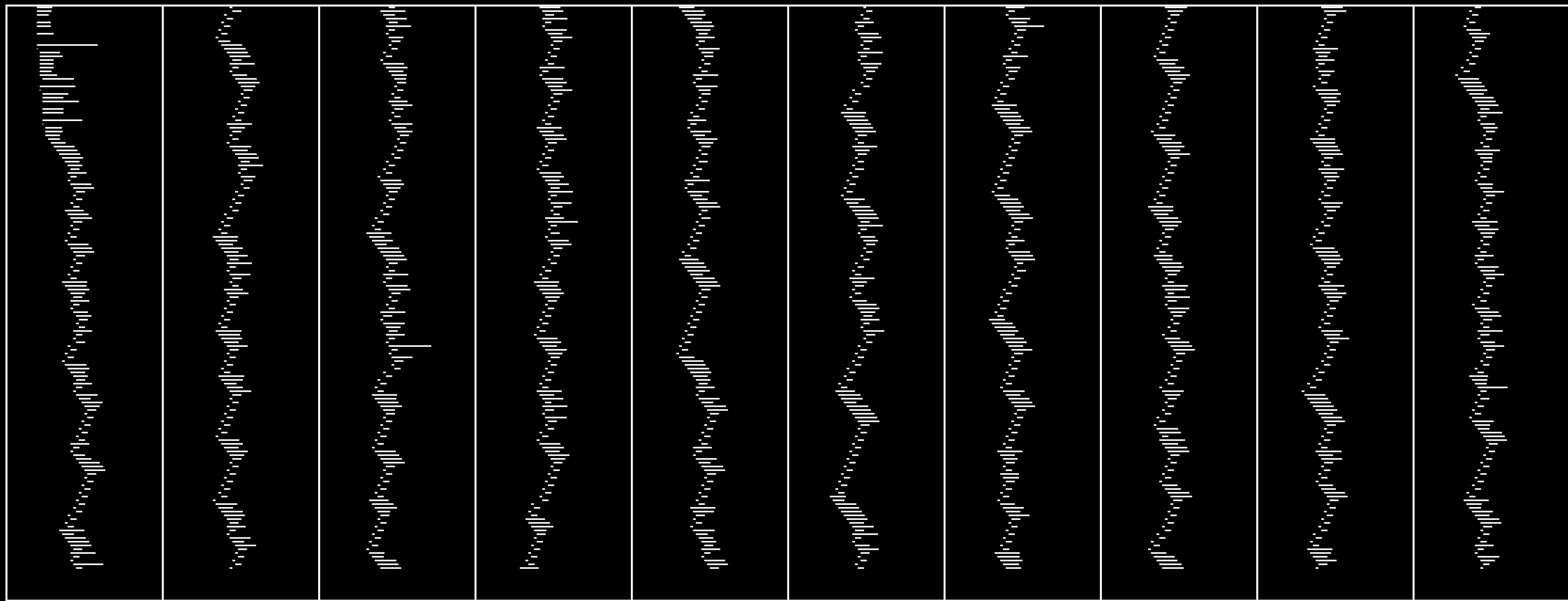
Train a classifier

- Decision tree classifiers are very efficient.
- Ask: “What is the state of pixel x?”
- Question splits list in to 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:



. . . which continues for 2 more pages.

```
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)

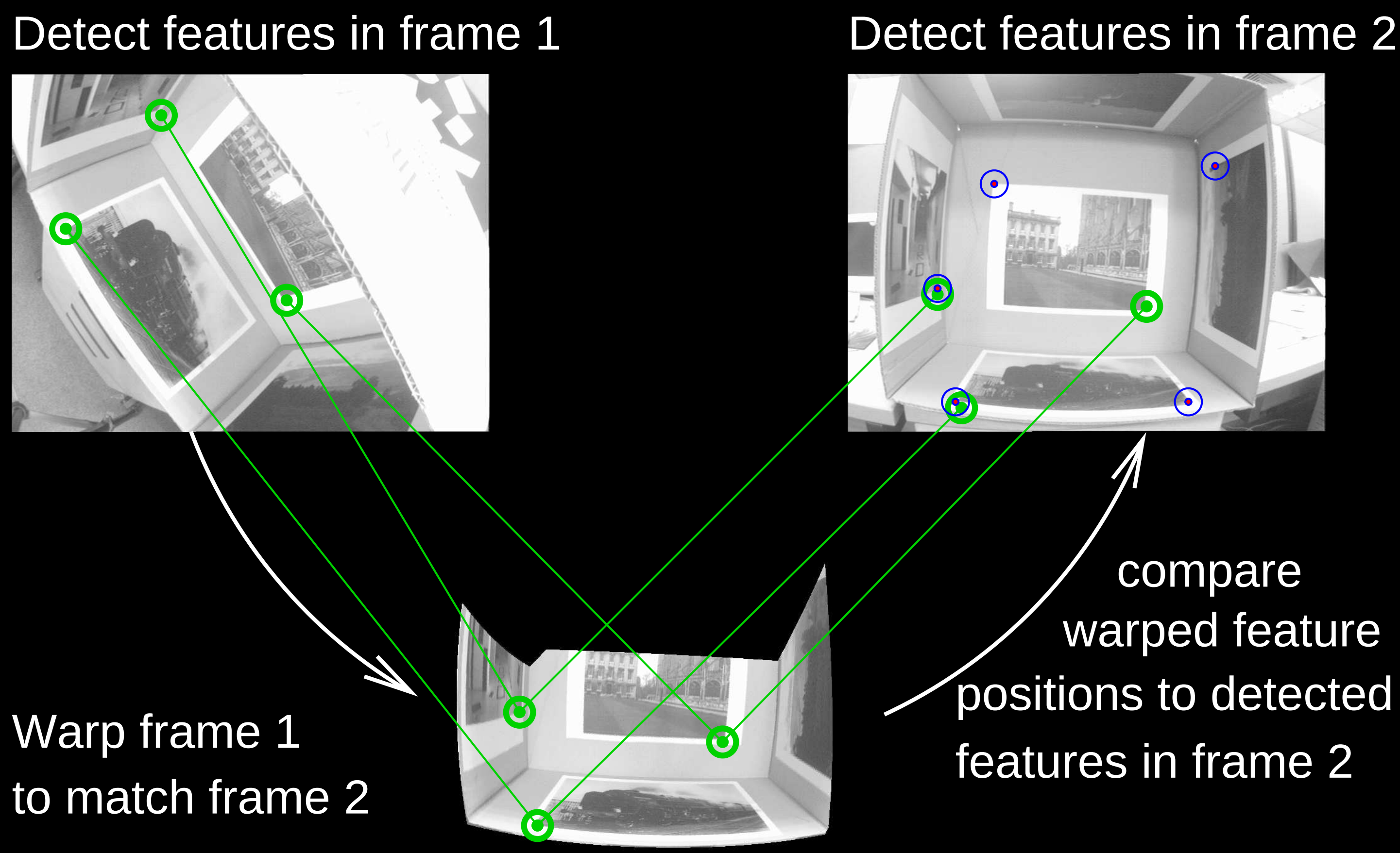
Detector	Set 1		Set 2	
	Pixel rate (MPix/s)	%	MPix/s	%
FAST $n = 9$	188	4.90	179	5.15
FAST $n = 12$	158	5.88	154	5.98
Original FAST ($n = 12$)	79.0	11.7	82.2	11.2
SUSAN	12.3	74.7	13.6	67.9
Harris	8.05	115	7.90	117
Shi-Tomasi	6.50	142	6.50	142
DoG	4.72	195	5.10	179

- 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?



Repeat for all pairs in a sequence

FAST-ER: Enhanced Repeatability

- Define feature detector as:

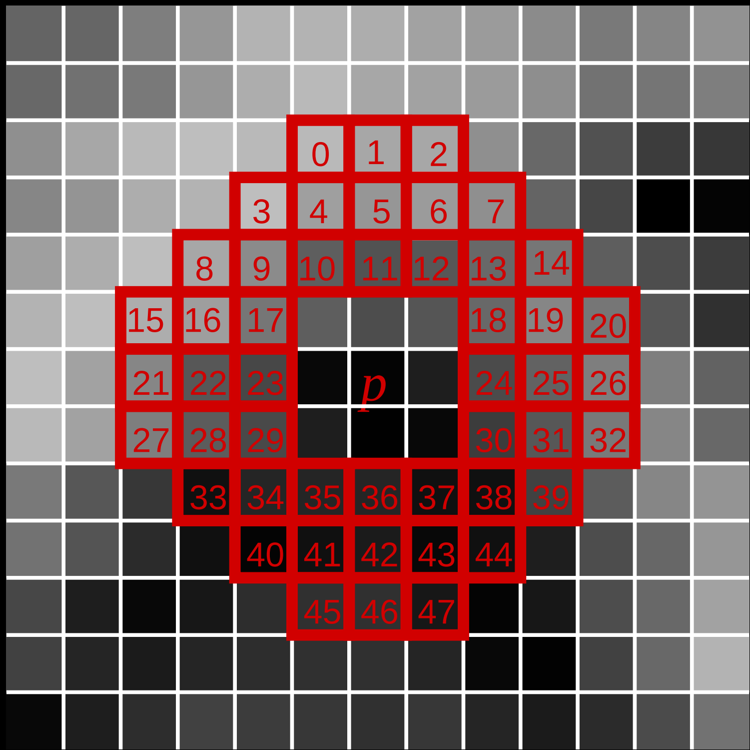
  A decision tree which detects points with a high repeatability.

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

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

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:

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

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

Cost function

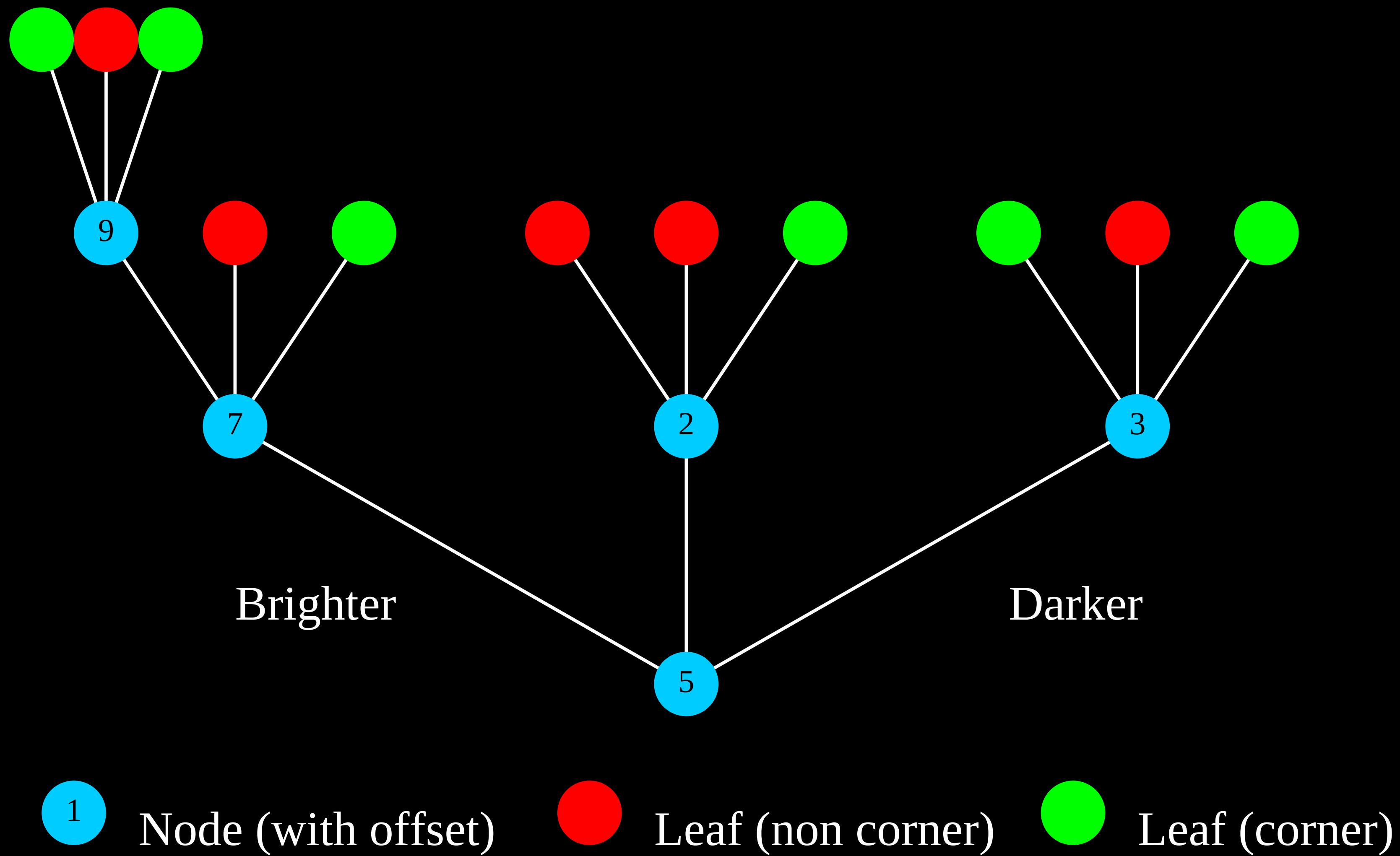
1. Higher repeatability is better.
2. Every pixel is a feature $\Rightarrow$ repeatability is 100%.
3. A single detected feature can have 100% repeatability.

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.

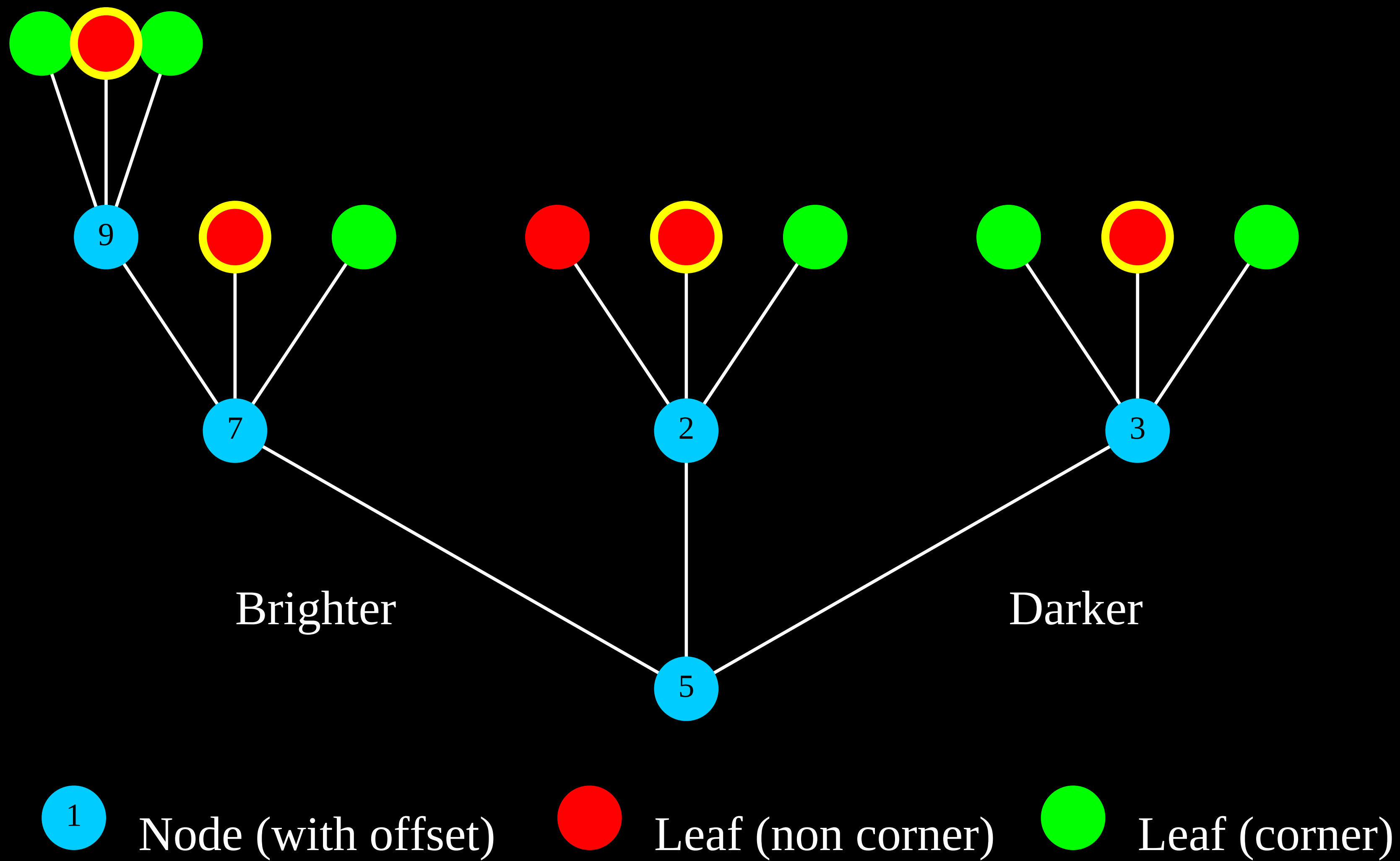
Operations



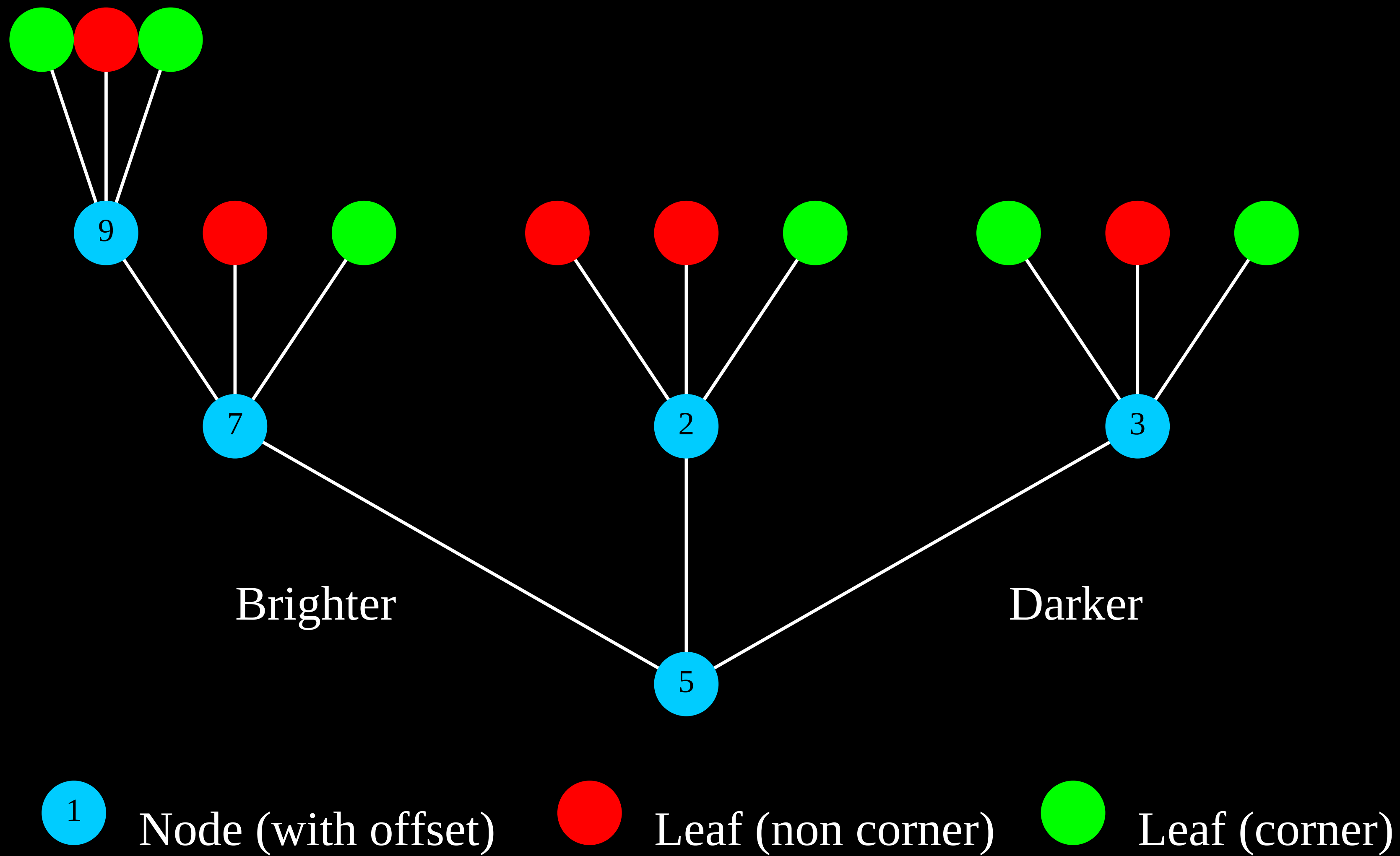
9
7
2
3
5
Brighter
Darker
1
Node (with offset)
Leaf (non corner)
Leaf (corner)

Operations

‘Similar’ leaf nodes are constrained.

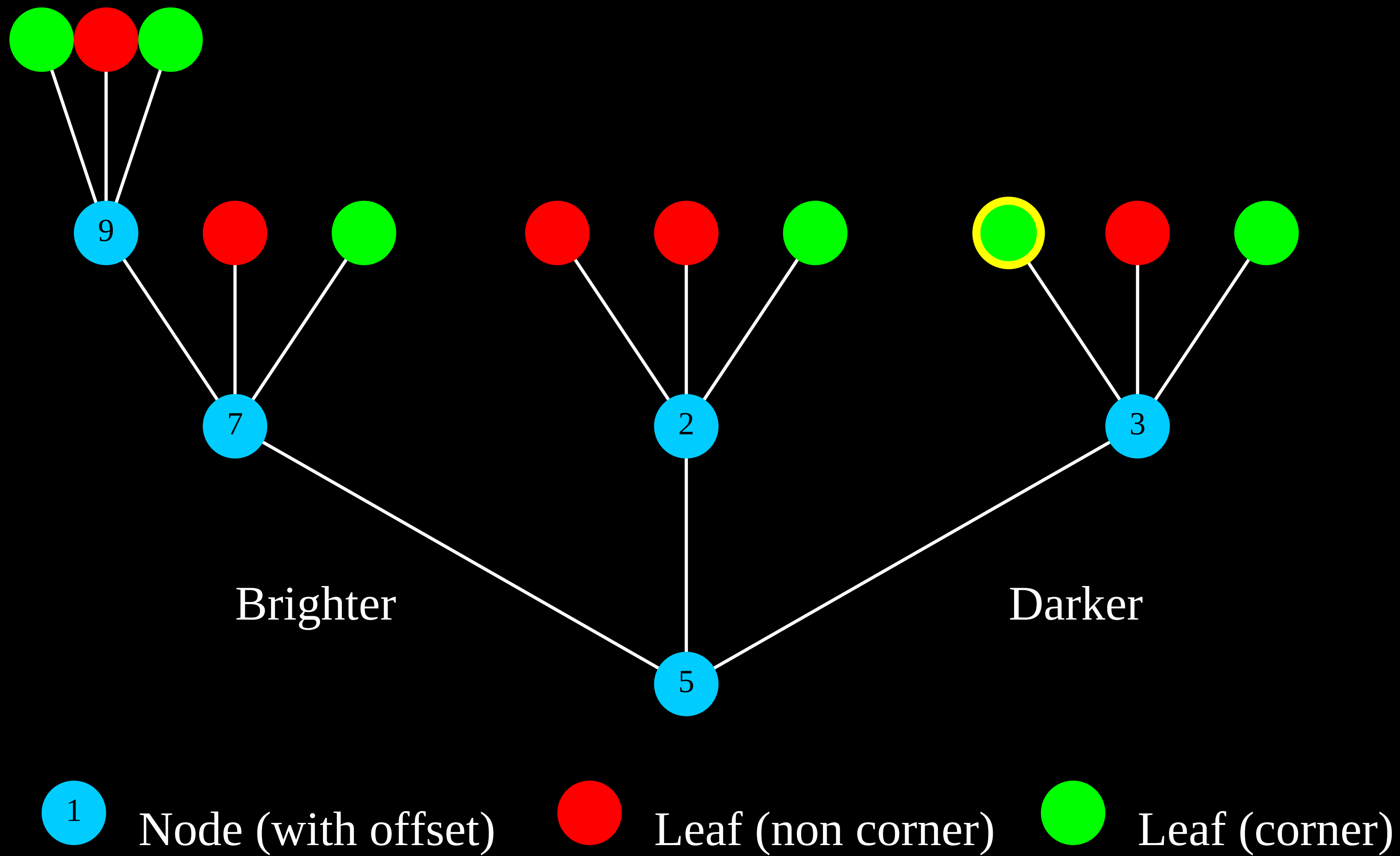


Operations


9
7
2
3
5
Brighter
Darker
1
Node (with offset)
Leaf (non corner)
Leaf (corner)

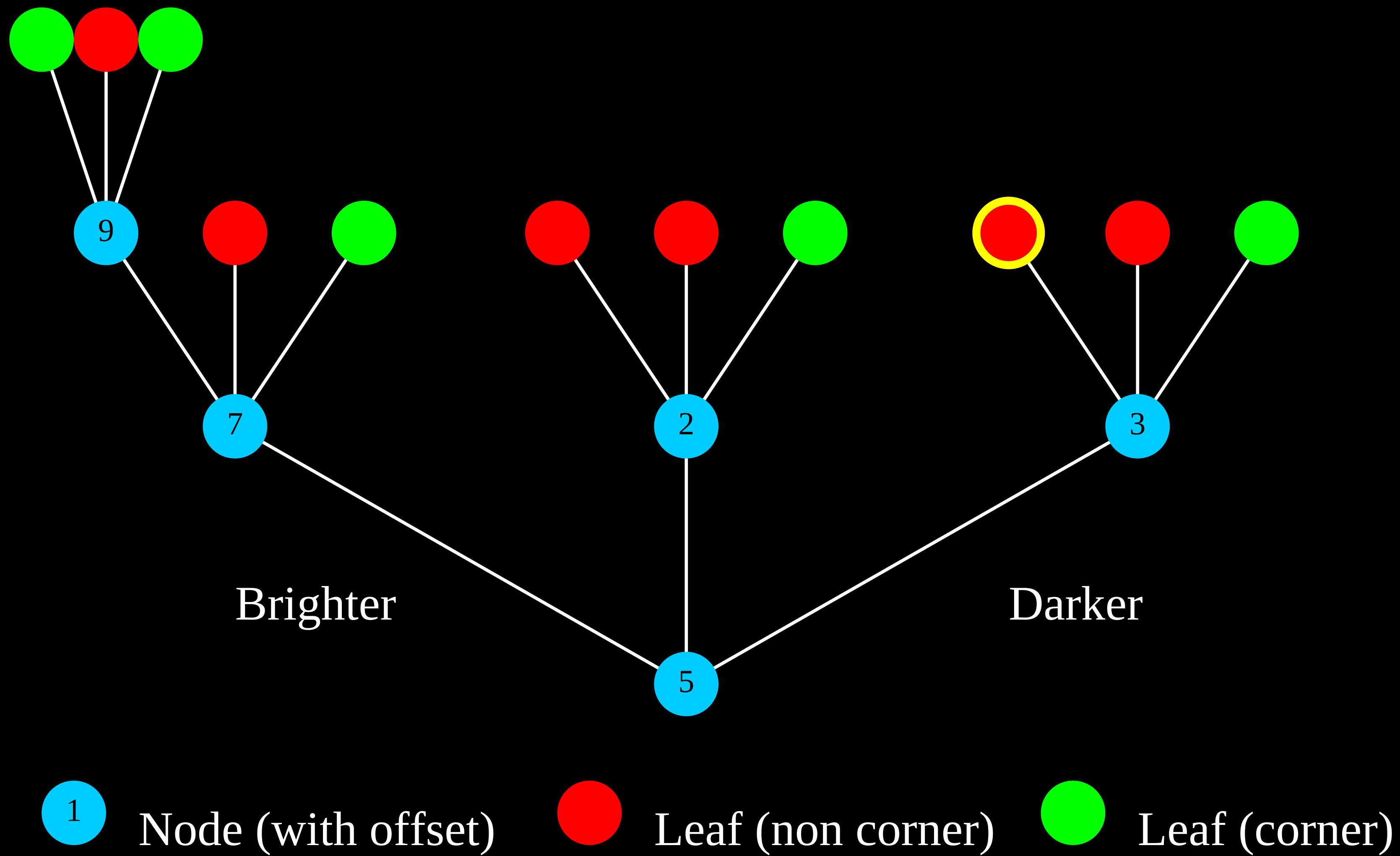
Operations

Select a random node. If node is a leaf:



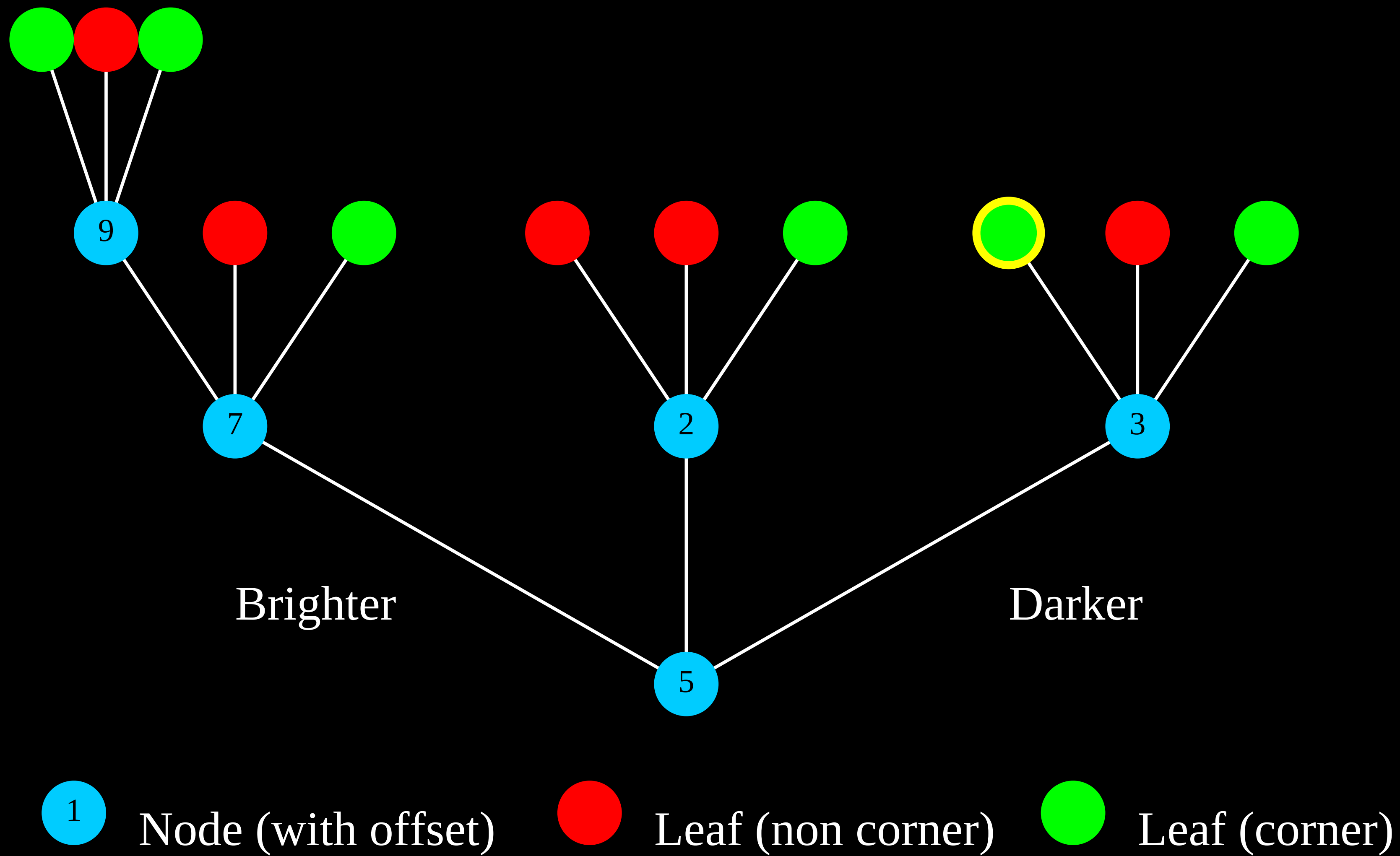
Operations

flip the class (if possible), . . .



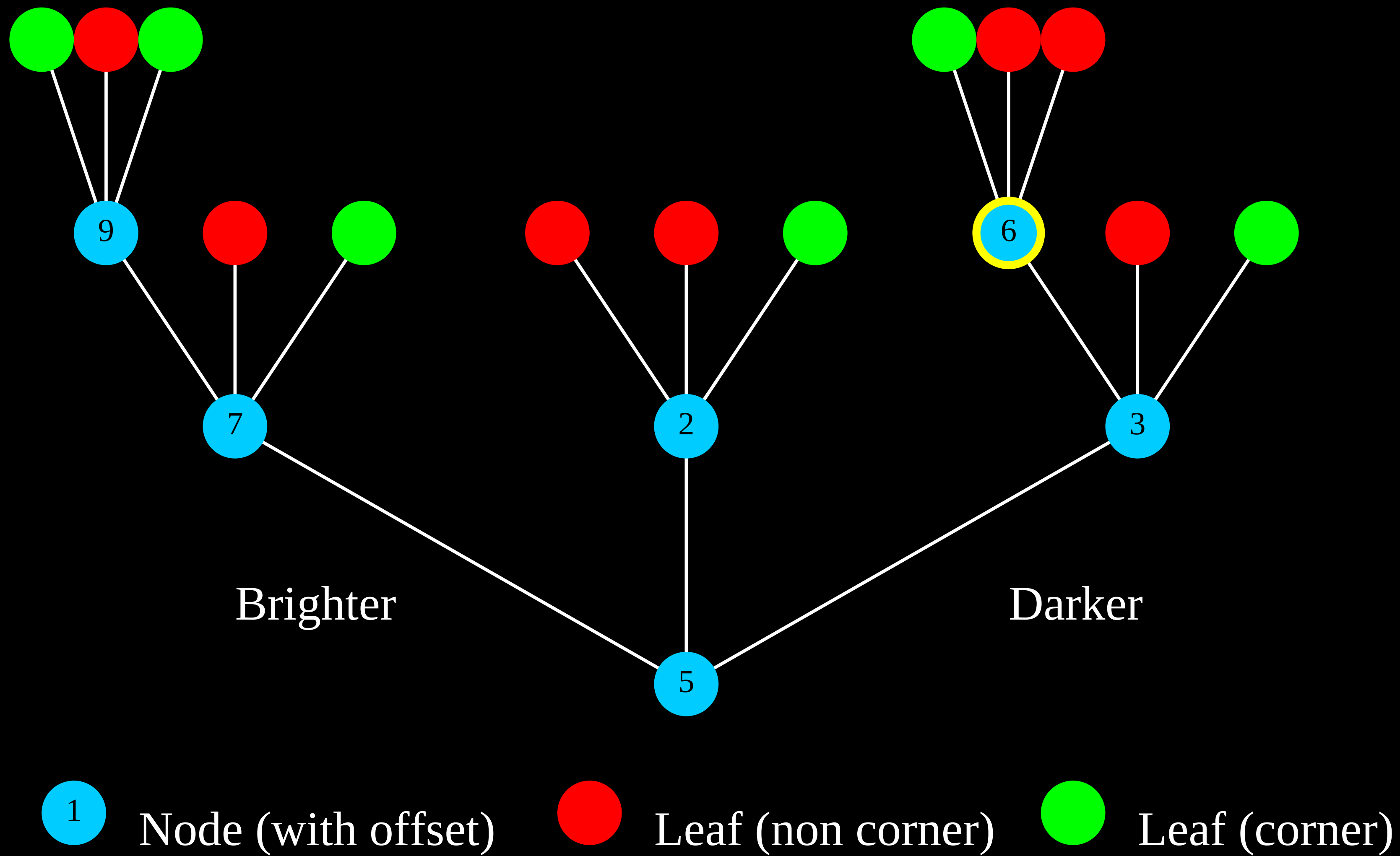
Operations

. . . or . . .



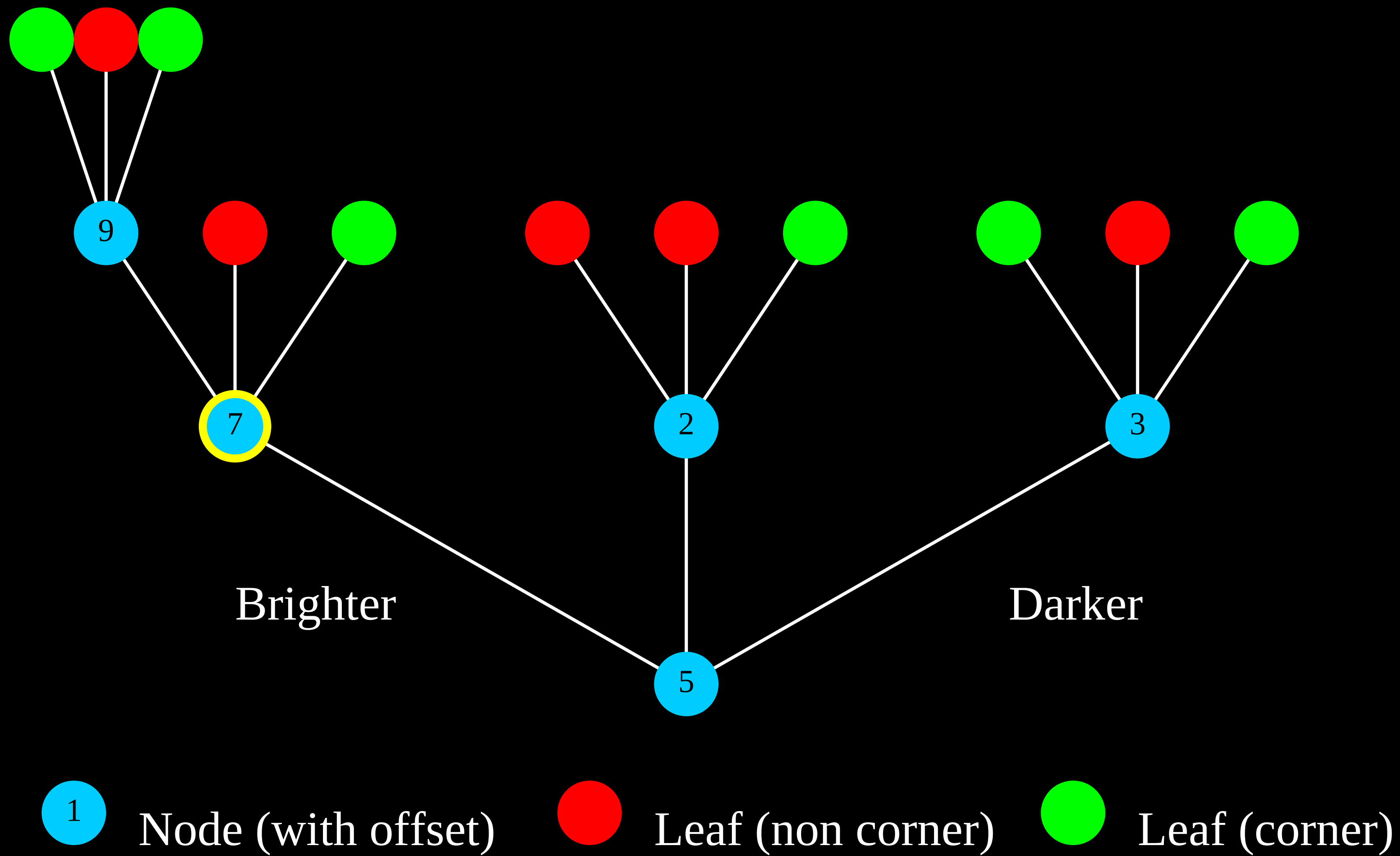
Operations

grow a random subtree.



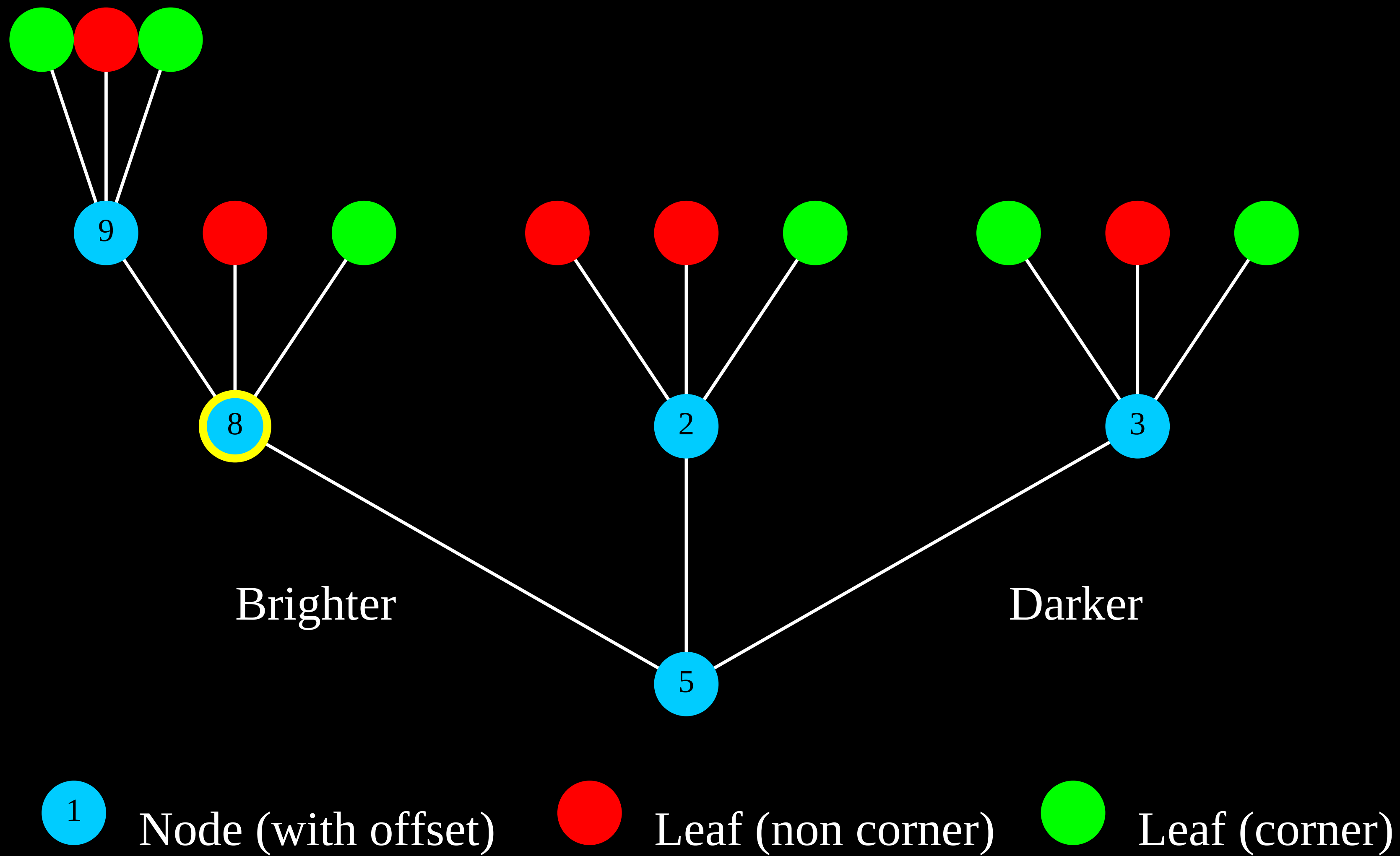
Operations

If node is a non-leaf:



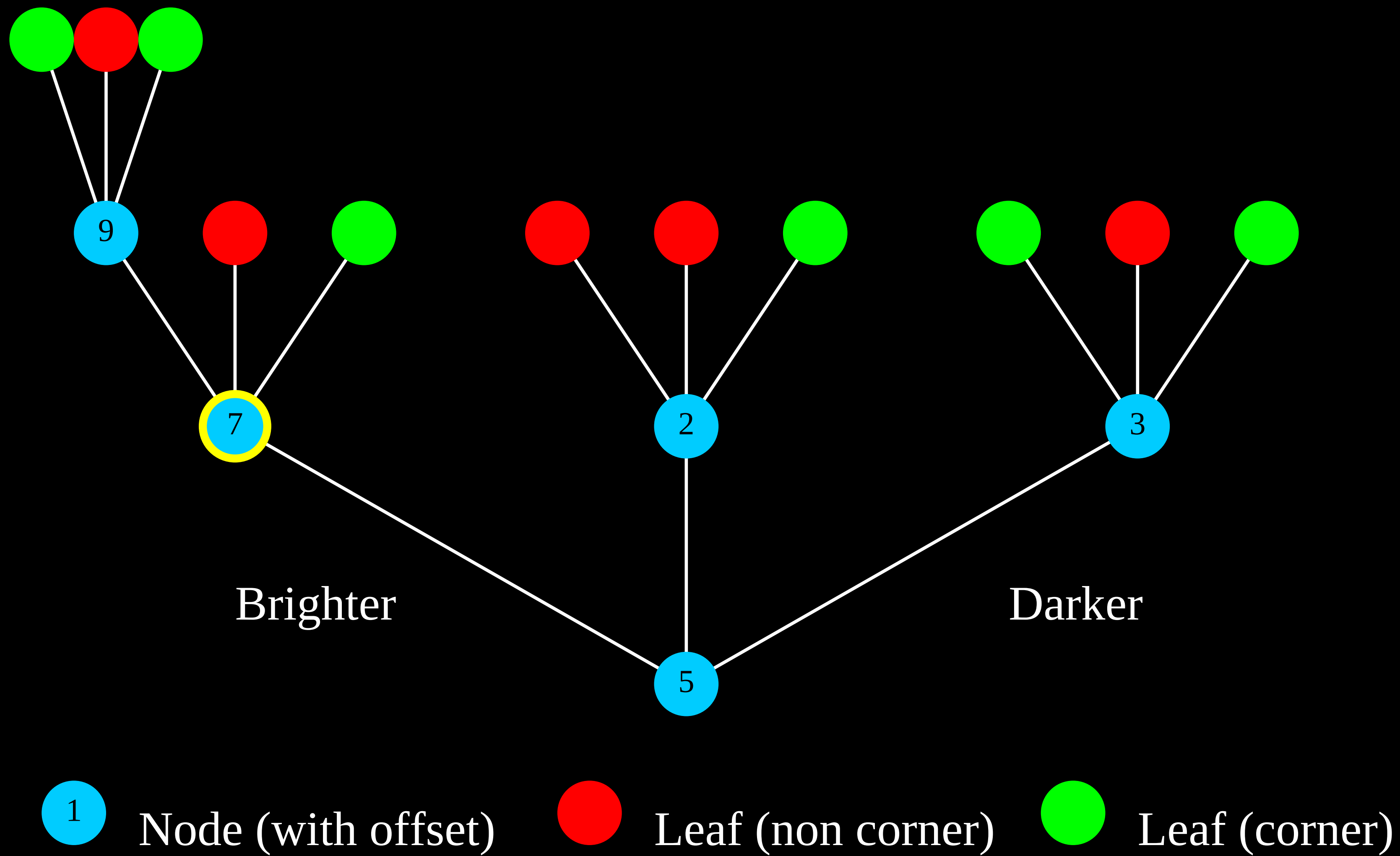
Operations

randomize the offset, . . .



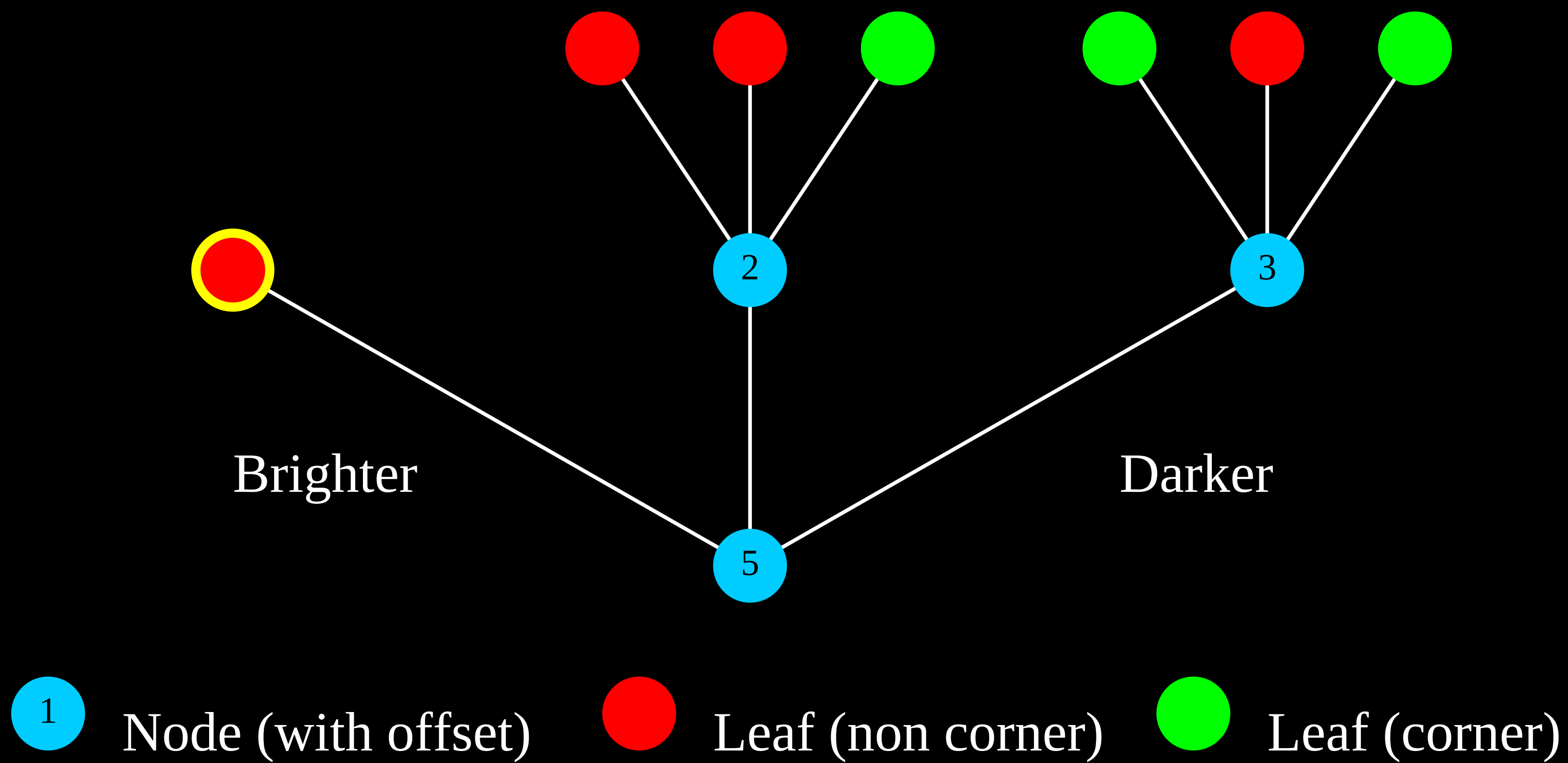
Operations

. . . or . . .



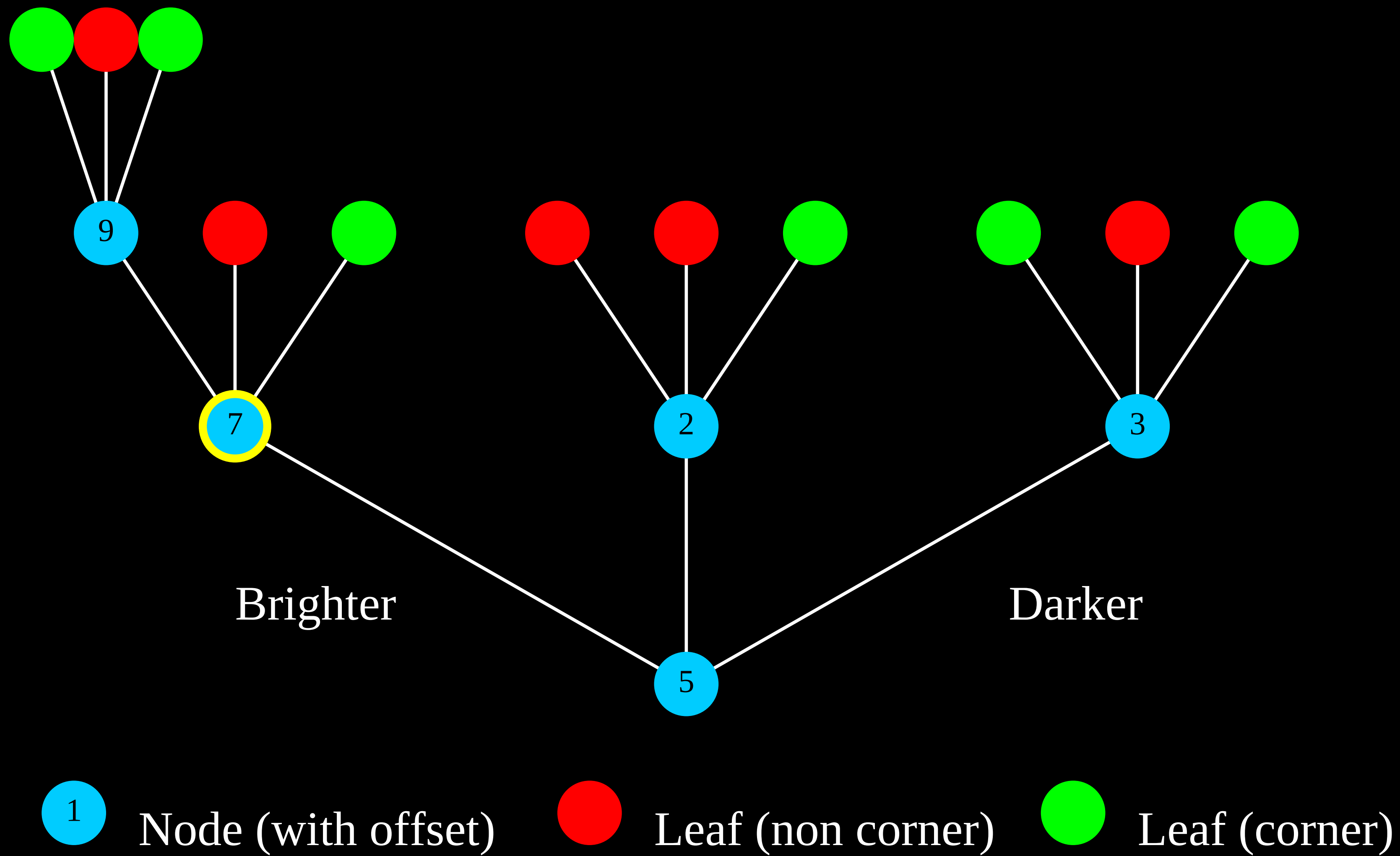
Operations

replace node with a leaf, . . .



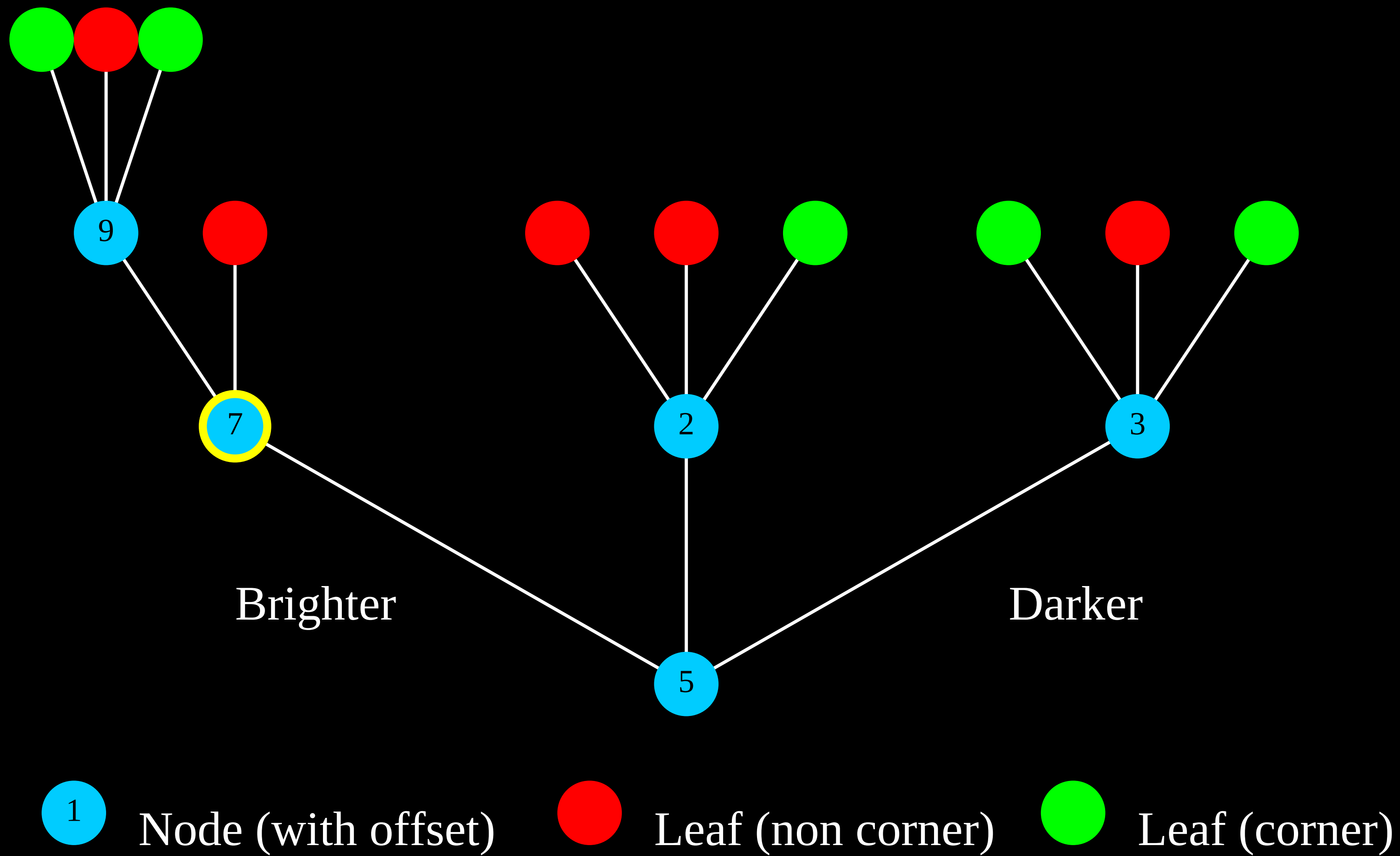
Operations

. . . or . . .



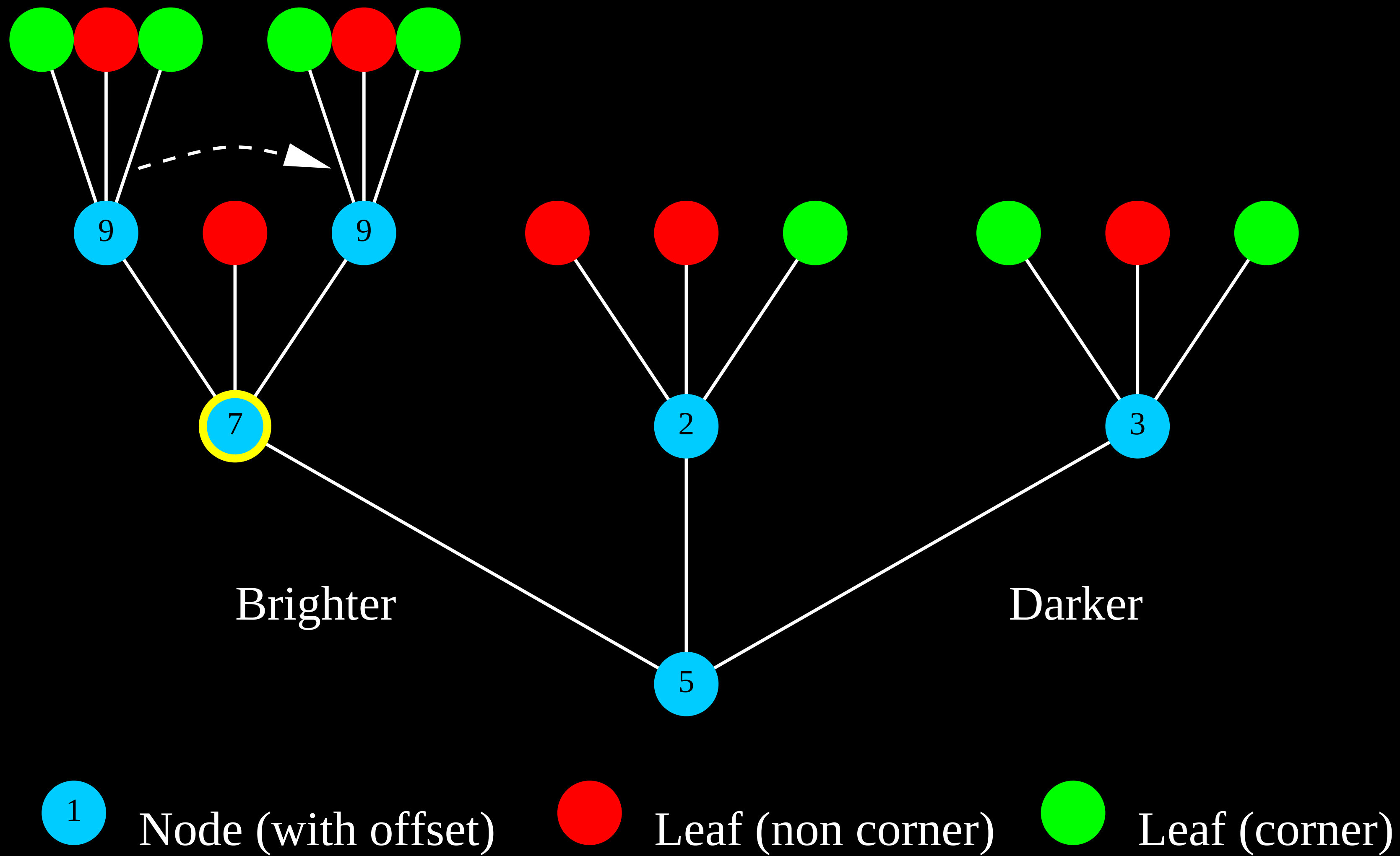
Operations

delete one subtree



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:

$$e^{\frac{\mathrm{oldcost}-\mathrm{cost}}{\mathrm{temp}}} > \mathrm{rand}(0,1)$$

7. Reduce the temperature.

Now repeat that 200 times.

Training data for repeatability

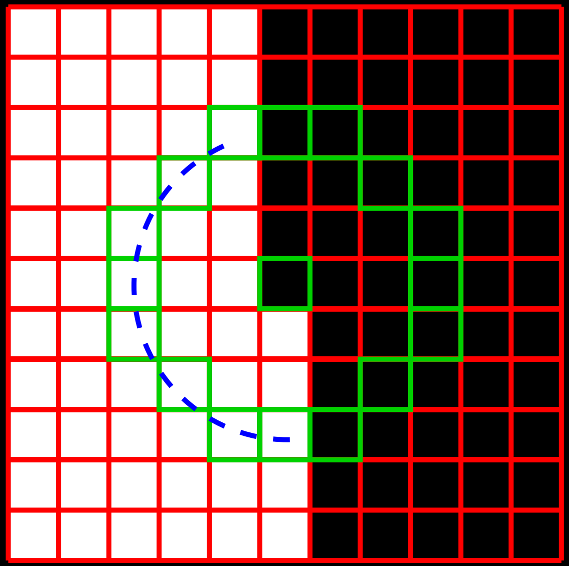
- Change in scale.
- Mostly affine warping.
- Varied texture.

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

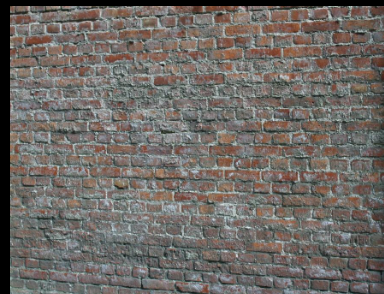






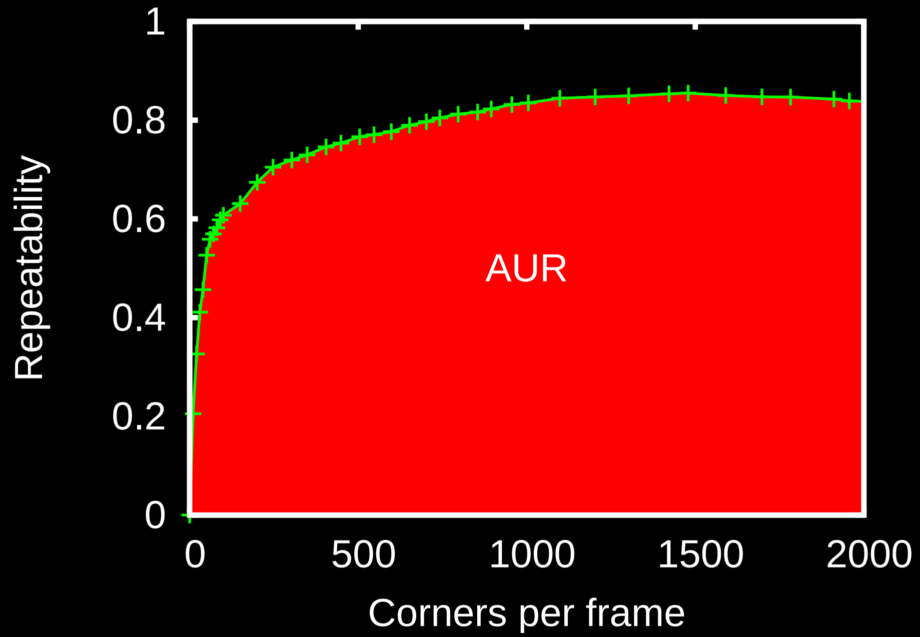






Aggregate results

Detector	AUR
FAST-ER	1313.6
FAST-9	1304.57
DoG	1275.59
Shi & Tomasi	1219.08
Harris	1195.2
Harris-Laplace	1153.13
FAST-12	1121.53
SUSAN	1116.79
Random	271.73



How FAST? (very)

Detector	Set 1		Set 2	
	Pixel rate (MPix/s)	%	MPix/s	%
FAST $n = 9$	188	4.90	179	5.15
FAST $n = 12$	158	5.88	154	5.98
Original FAST ($n = 12$)	79.0	11.7	82.2	11.2
FAST-ER	75.4	12.2	67.5	13.7
SUSAN	12.3	74.7	13.6	67.9
Harris	8.05	115	7.90	117
Shi-Tomasi	6.50	142	6.50	142
DoG	4.72	195	5.10	179

- 3.0GHz Pentium 4
- Set 1: 992×668 pixels.
- set 2: 352×288 (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

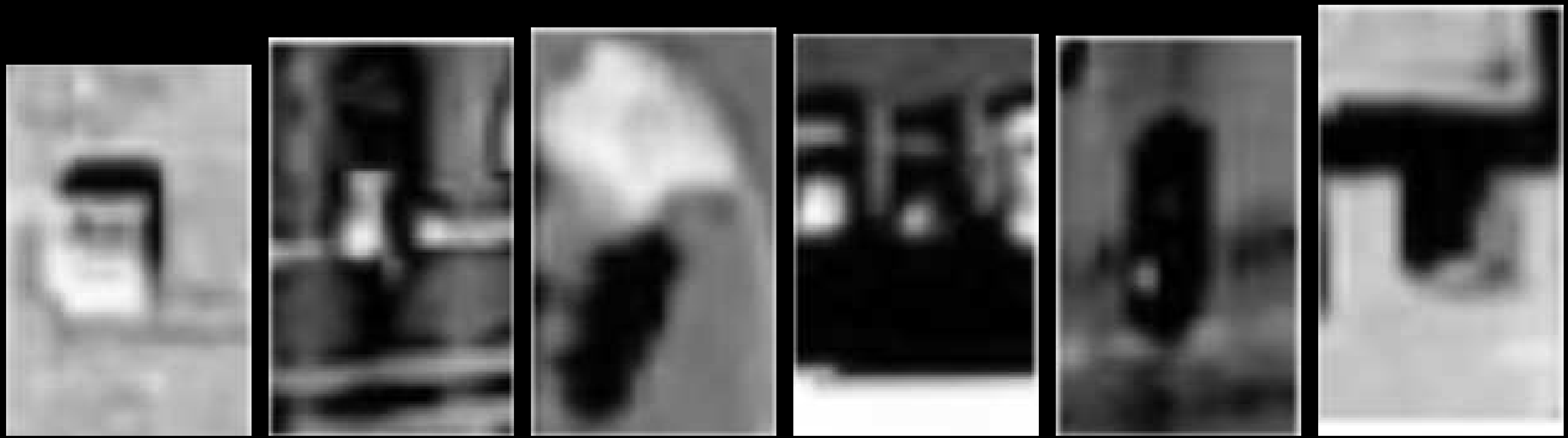
Target detection

Traffic analysis

Damian Eads, Edward Rosten, David Helmbold

Object detection: difficulties

Which ones are cars?

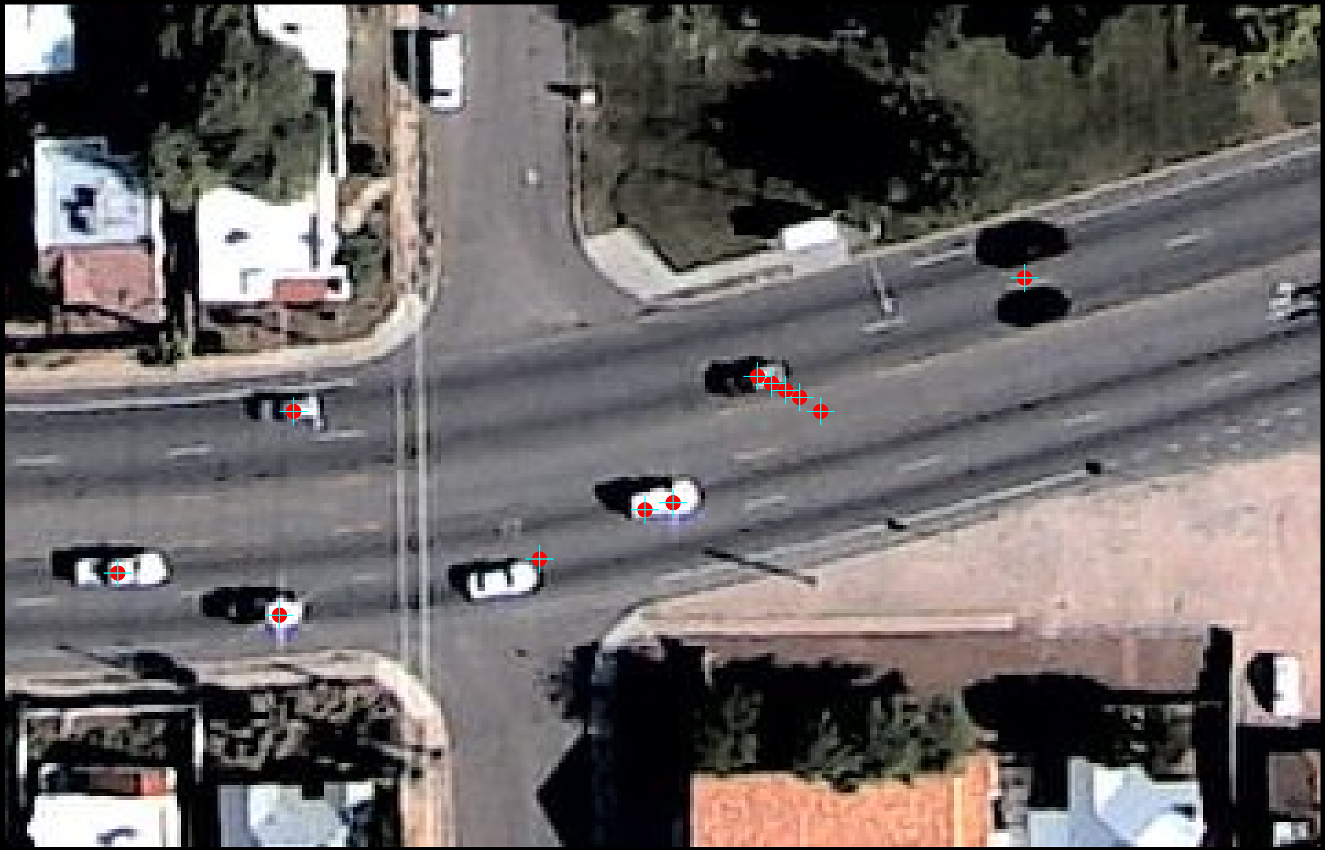


- Problem is unstructured

  Image $\longrightarrow \{(x_1, y_1), (x_2, y_2), \cdots\}$
- Number of objects unknown *a priori*
- Not a fixed set of labels

What is a detection anyway?

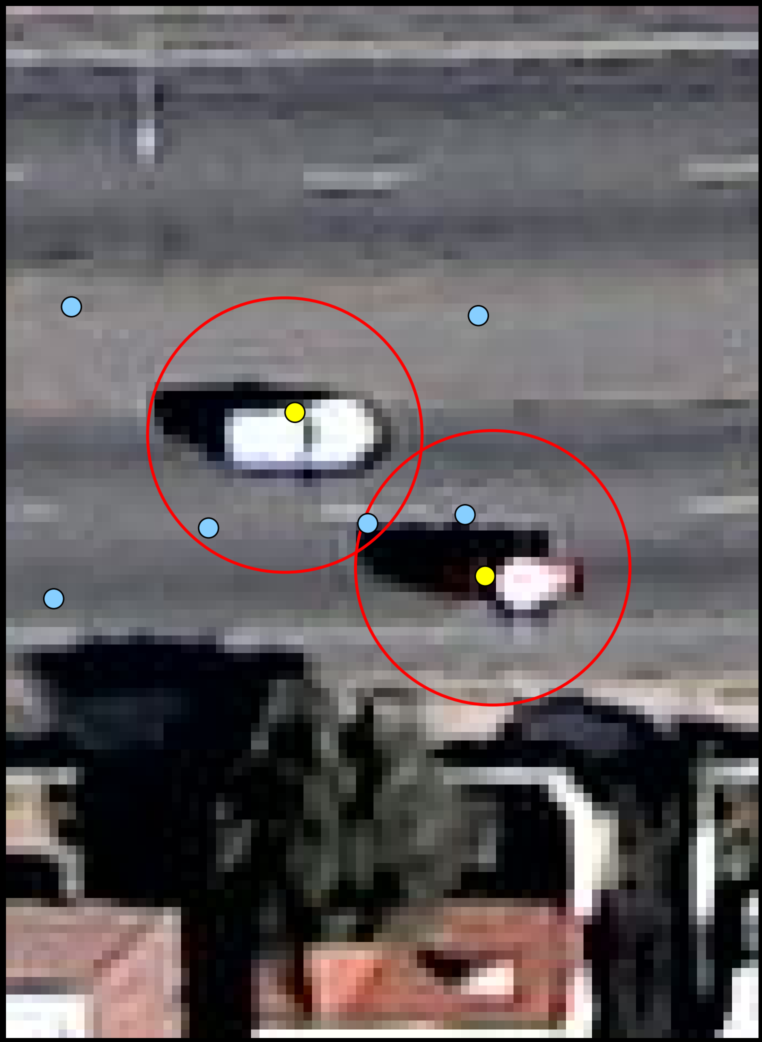
1. Not pixels! 50% of pixels on all of the objects is not the same as all of the pixels on 50% of the objects.
2. It depends...



Measures of performance

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

Measures of performance

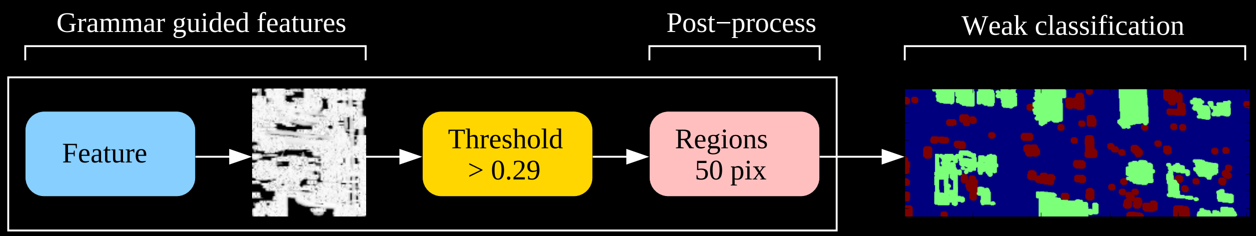


- Identification:
  - Within boundary
- Tracking
  - Nearby, but with unique assignment
- Counting
  - Unique assignment
  - Within radius of sliding window

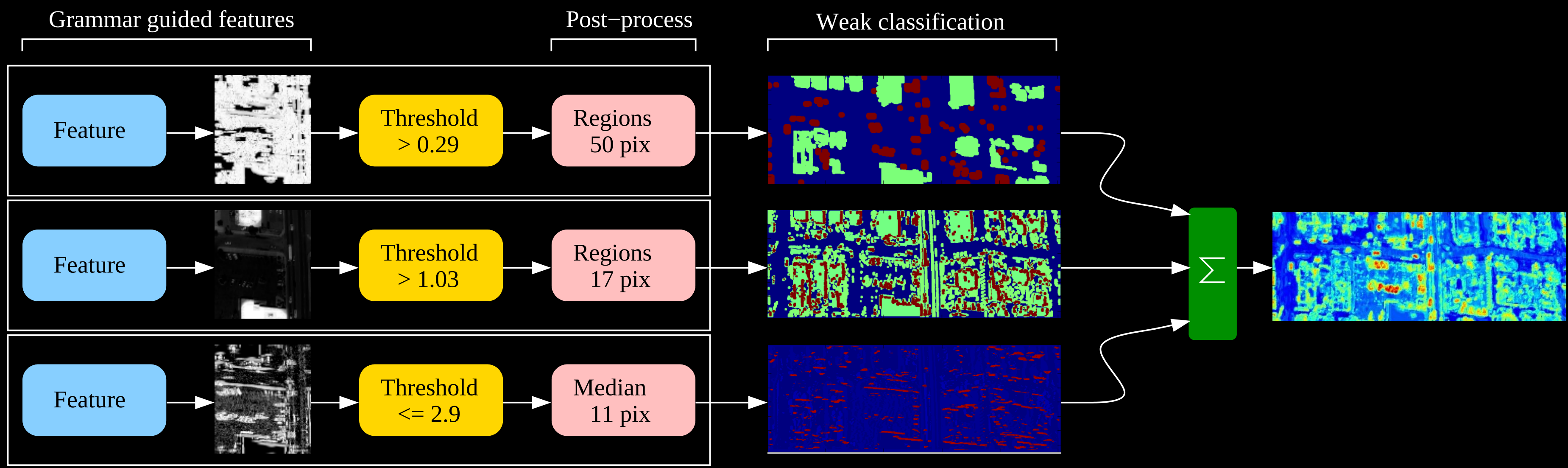
Measures of performance

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

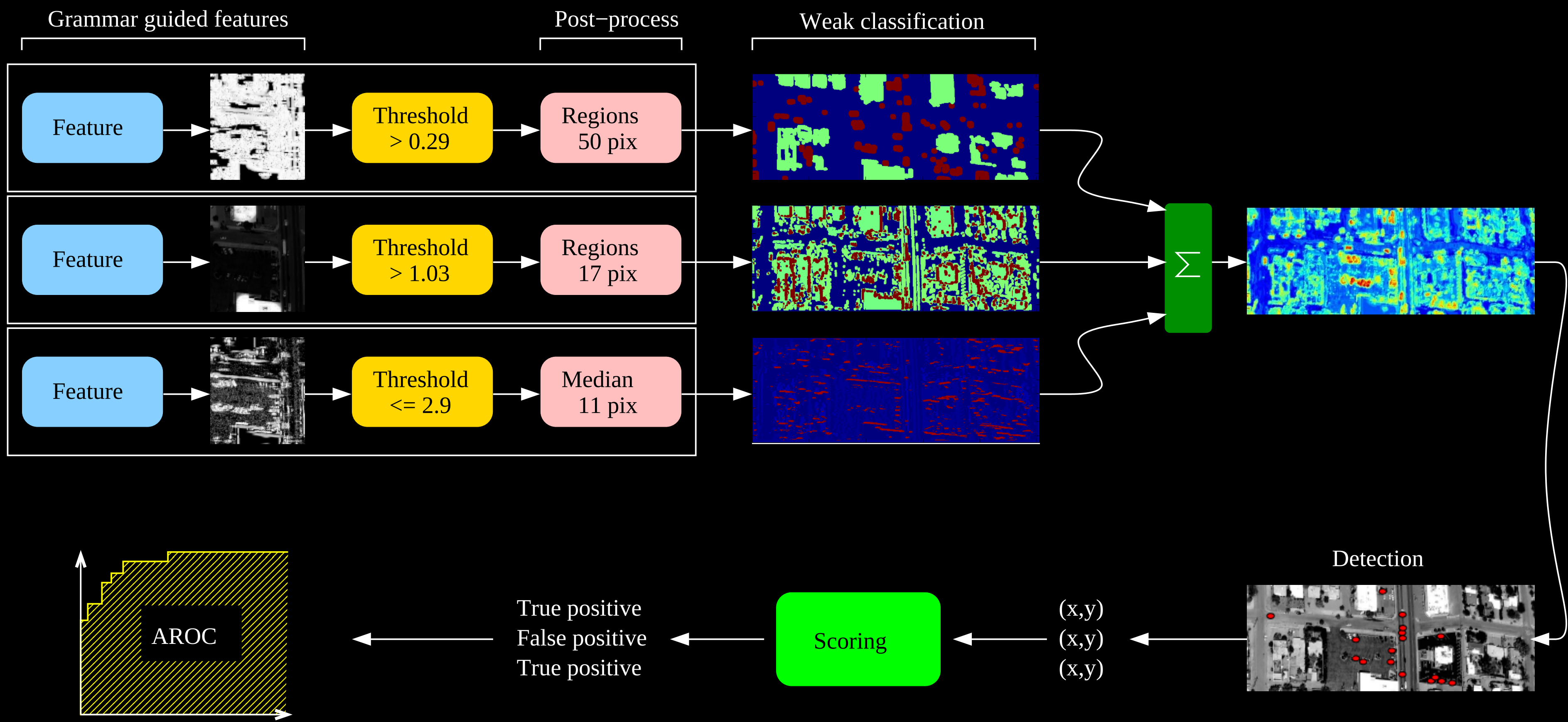
System layout



System layout

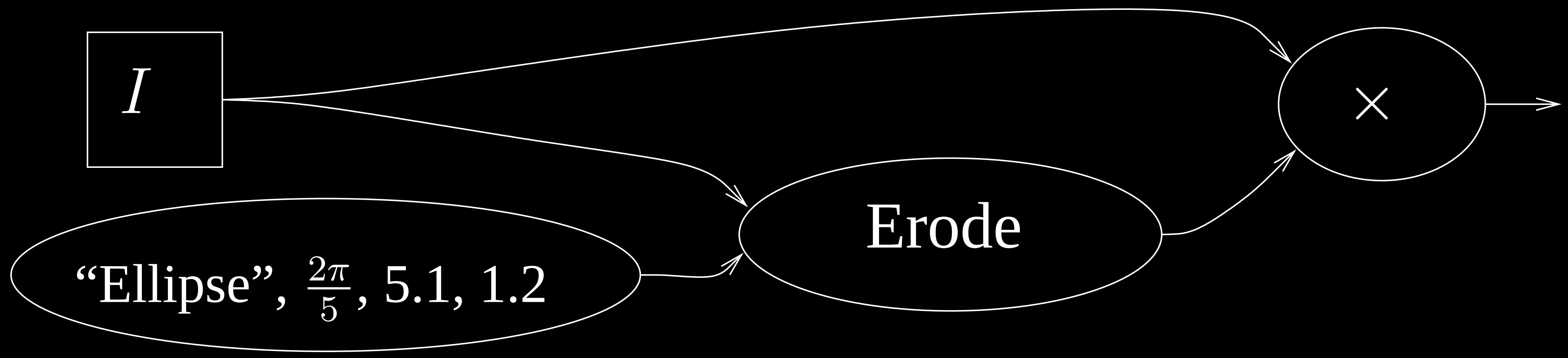


System layout



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, …



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

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

$f(x) = \text{Feature}(x)$

Example

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

$f(x) = \text{Binary}(\text{Unary}(x), \text{Unary}(x))$

Example

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

$$f(x) = \text{Binary}(\text{Unary}(x), \text{Erode}(x, \text{RandomSE}()))$$

Example

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

$$f(x) = \text{Binary}(\text{Unary}(x), \text{Erode}(x, \text{Ellipse}(\tfrac{2\pi}{5}, 5.1, 1.2)))$$

Example

$$
\begin{aligned}
\mathrm{Feature}(x) &\rightarrow \mathrm{Binary}(\mathrm{Unary}(x), \mathrm{Unary}(x)) \mid \mathrm{Unary}(x) \\
\mathrm{Unary}(x) &\rightarrow x \mid \mathrm{Erode}(x, \mathrm{RandomSE}()) \\
\mathrm{Binary}(x, y) &\rightarrow \mathrm{Add}(x, y) \mid \mathrm{Multiply}(x, y) \\
\mathrm{RandomSE}() &\rightarrow \mathrm{Ellipse}(\mathcal{U}(0, \pi),\ \mathcal{U}(1, 10),\ \mathcal{U}(1, 10))
\end{aligned}
$$

$$
f(x) = \mathrm{Binary}(x, \mathrm{Erode}(x, \mathrm{Ellipse}(\tfrac{2\pi}{5}, 5.1, 1.2)))
$$

Example

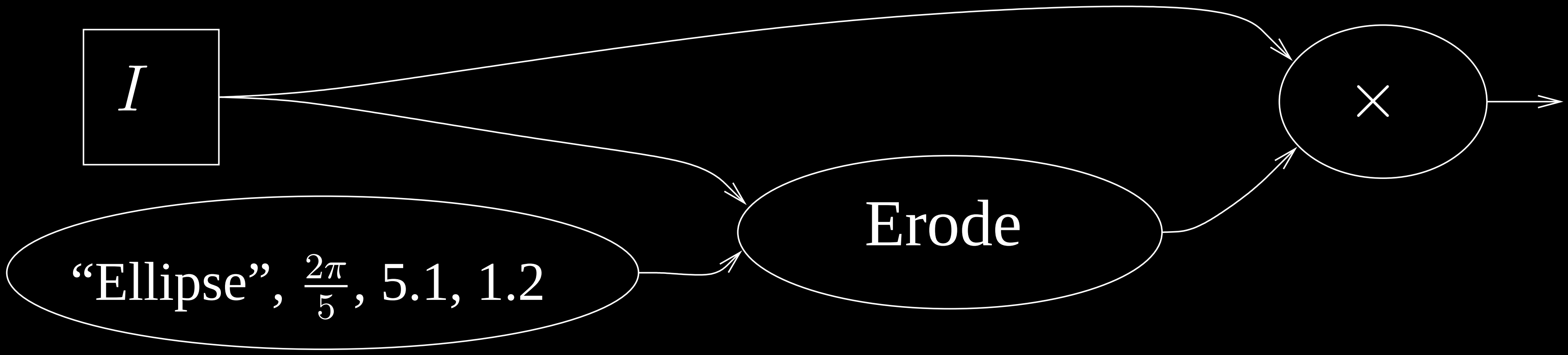
$$
\begin{aligned}
\text{Feature}(x) &\rightarrow \text{Binary}(\text{Unary}(x), \text{Unary}(x)) \mid \text{Unary}(x) \\
\text{Unary}(x) &\rightarrow x \mid \text{Erode}(x, \text{RandomSE}()) \\
\text{Binary}(x, y) &\rightarrow \text{Add}(x, y) \mid \text{Multiply}(x, y) \\
\text{RandomSE}() &\rightarrow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

$$f(x) = \text{Multiply}(x, \text{Erode}(x, \text{Ellipse}(\tfrac{2\pi}{5}, 5.1, 1.2)))$$

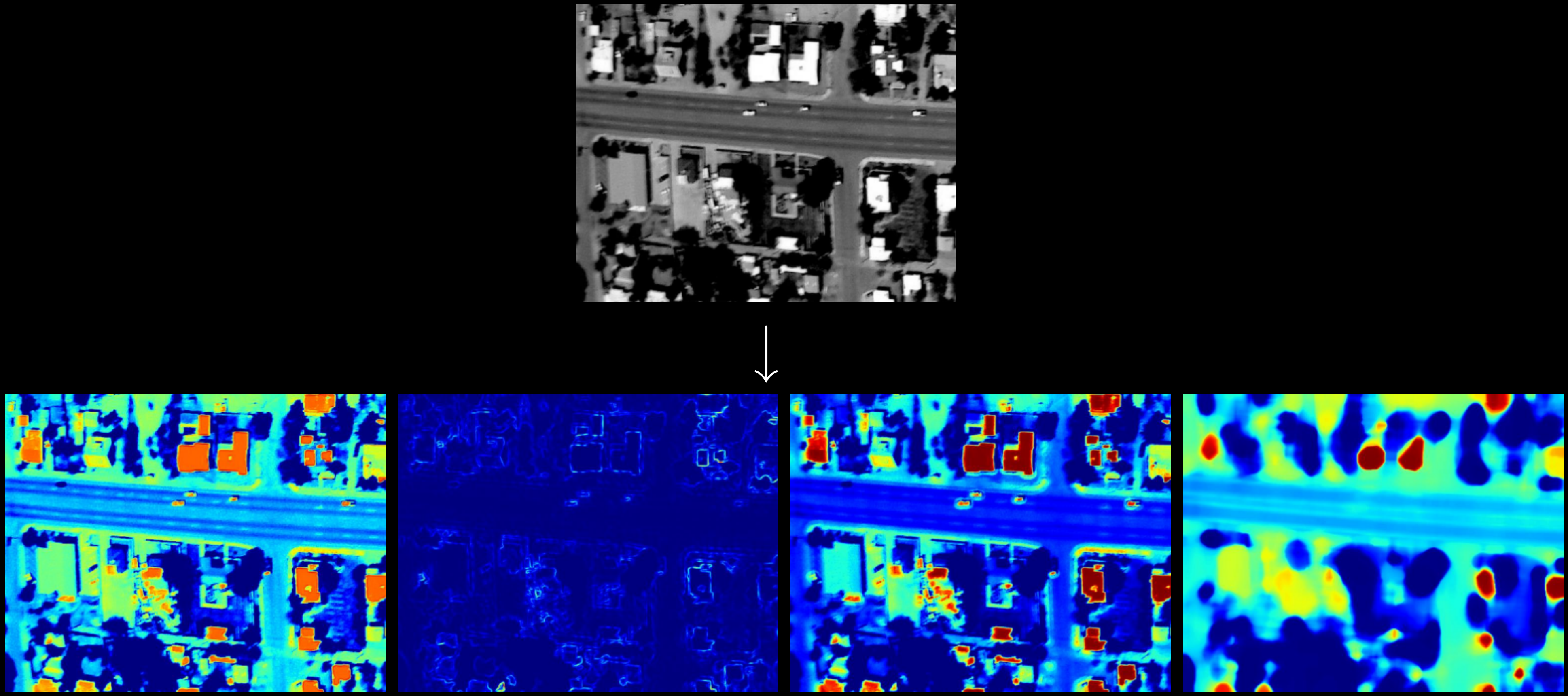
Example

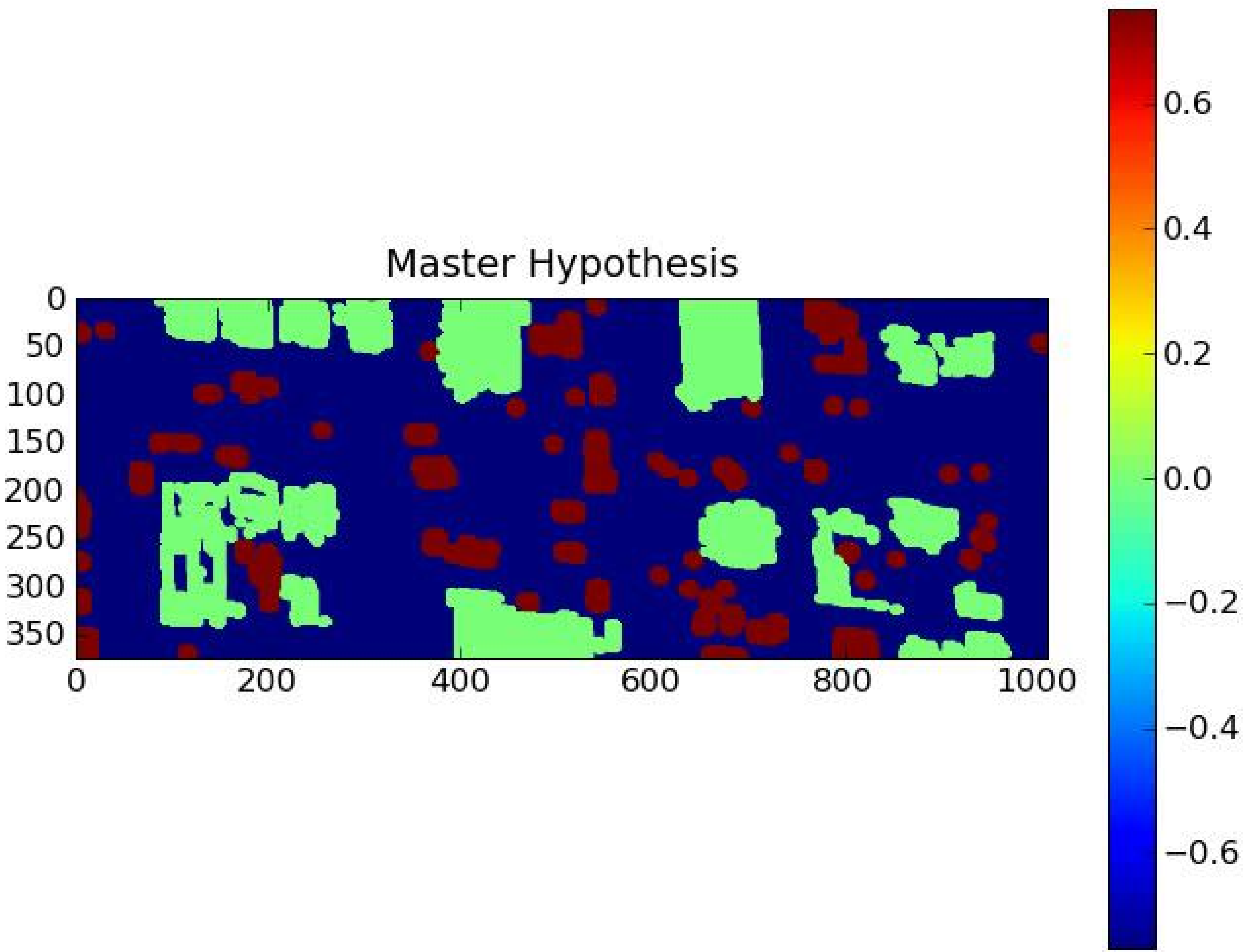
$$
\begin{aligned}
\text{Feature}(x) &\rightarrow \text{Binary}(\text{Unary}(x), \text{Unary}(x)) \mid \text{Unary}(x) \\
\text{Unary}(x) &\rightarrow x \mid \text{Erode}(x, \text{RandomSE}()) \\
\text{Binary}(x, y) &\rightarrow \text{Add}(x, y) \mid \text{Multiply}(x, y) \\
\text{RandomSE}() &\rightarrow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

$$f(x) = \text{Multiply}(x, \text{Erode}(x, \text{Ellipse}(\tfrac{2\pi}{5}, 5.1, 1.2)))$$



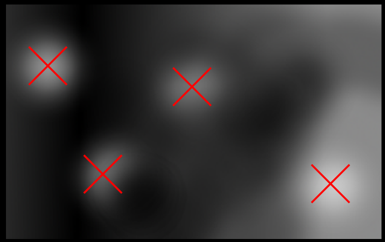
Some random features



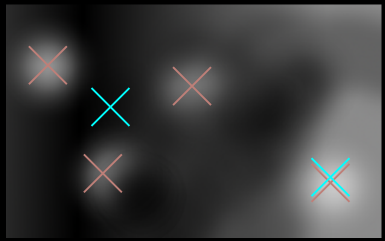


Master Hypothesis
0
50
100
150
200
250
300
350
0
200
400
600
800
1000
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

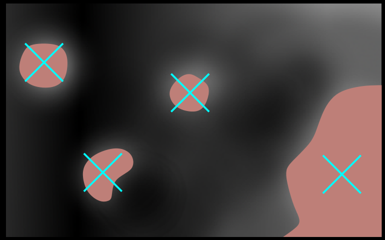
Turning pixels into objects



- Large local maxima
  Choice of pre-smoothing radius



- KDE on large local maxima
  Also kernel size

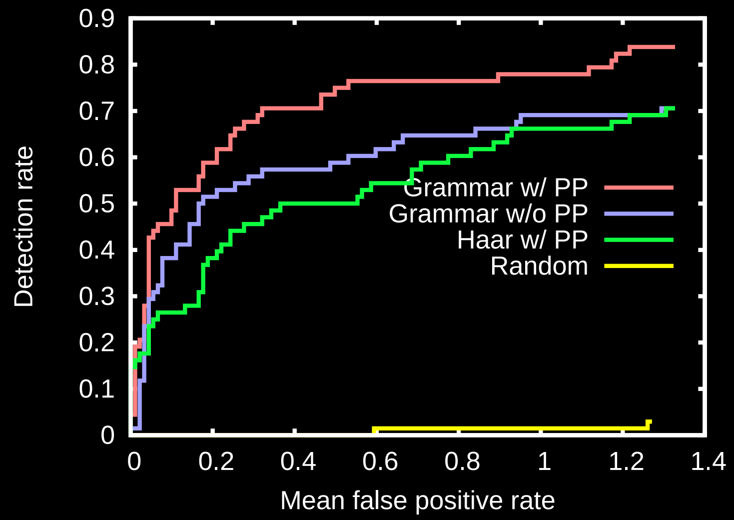


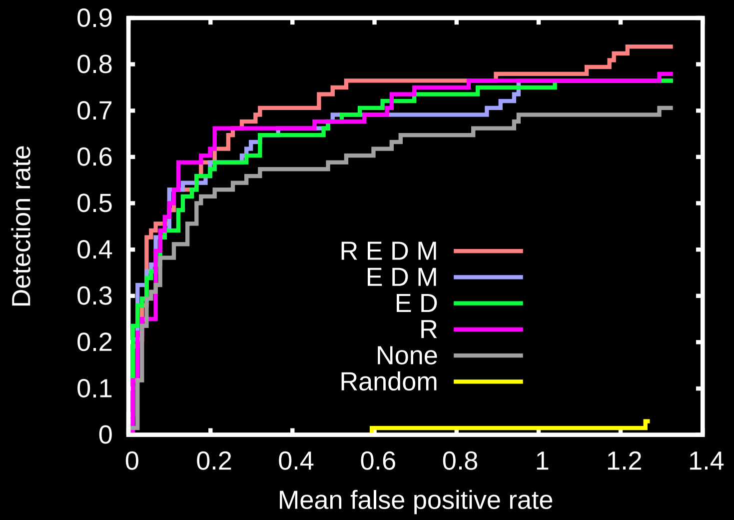
- Connected components
  Choice of threshold

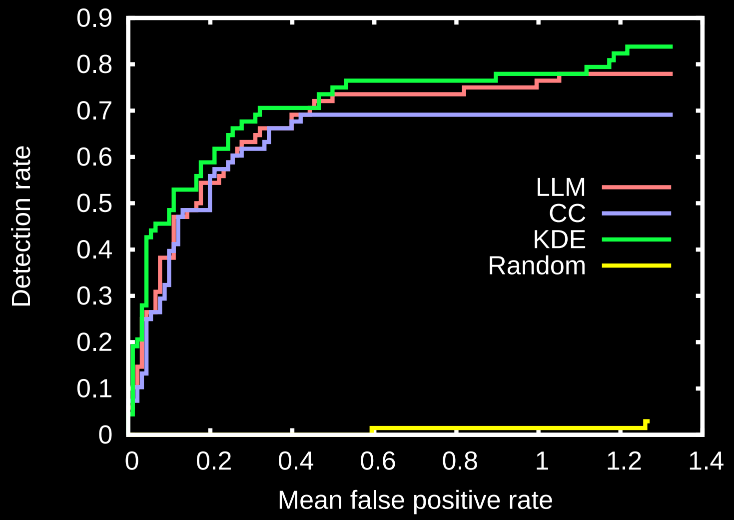
Optimize over data not used for boosting.

Results

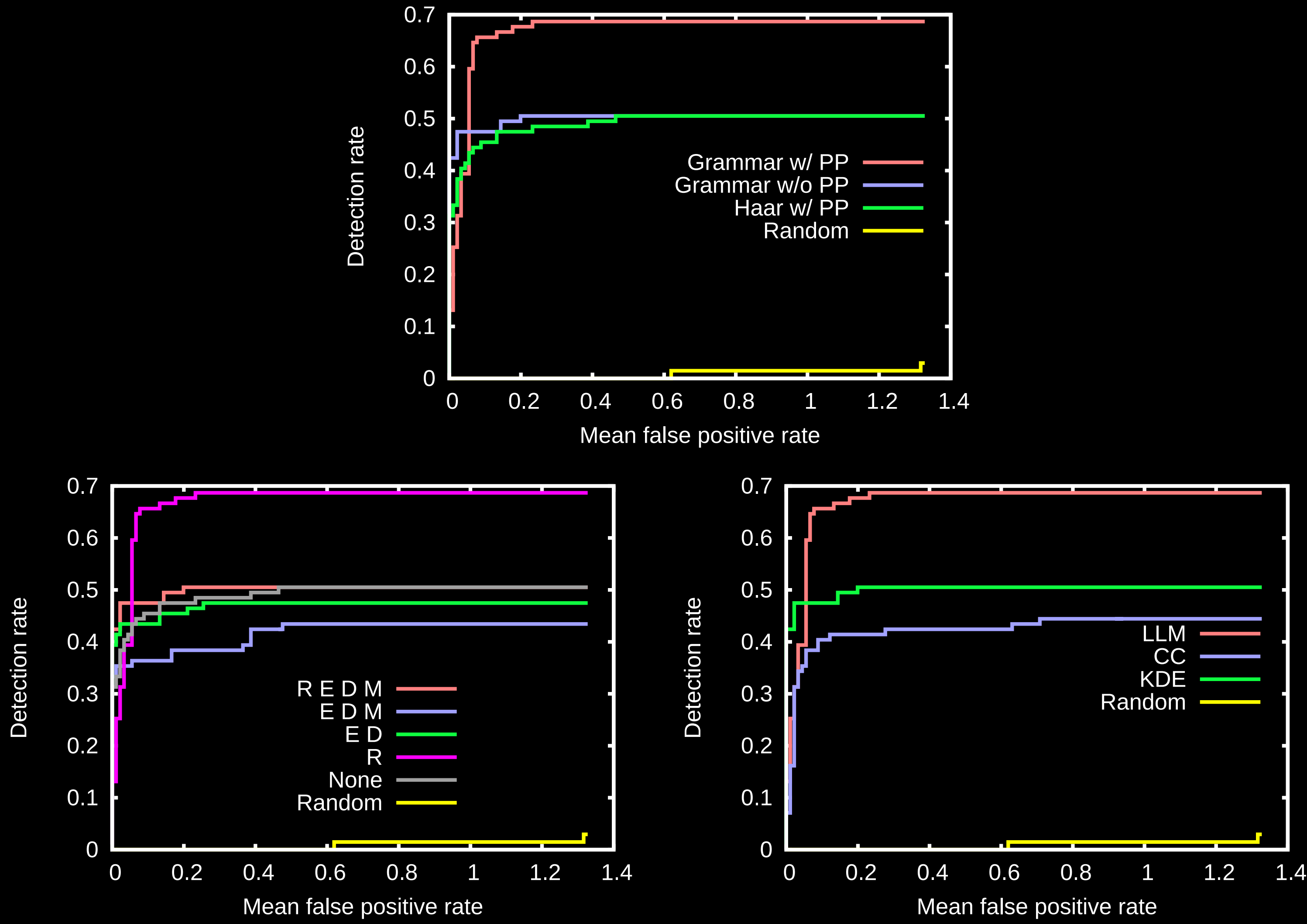
Results: Target detection







Results: Tracking



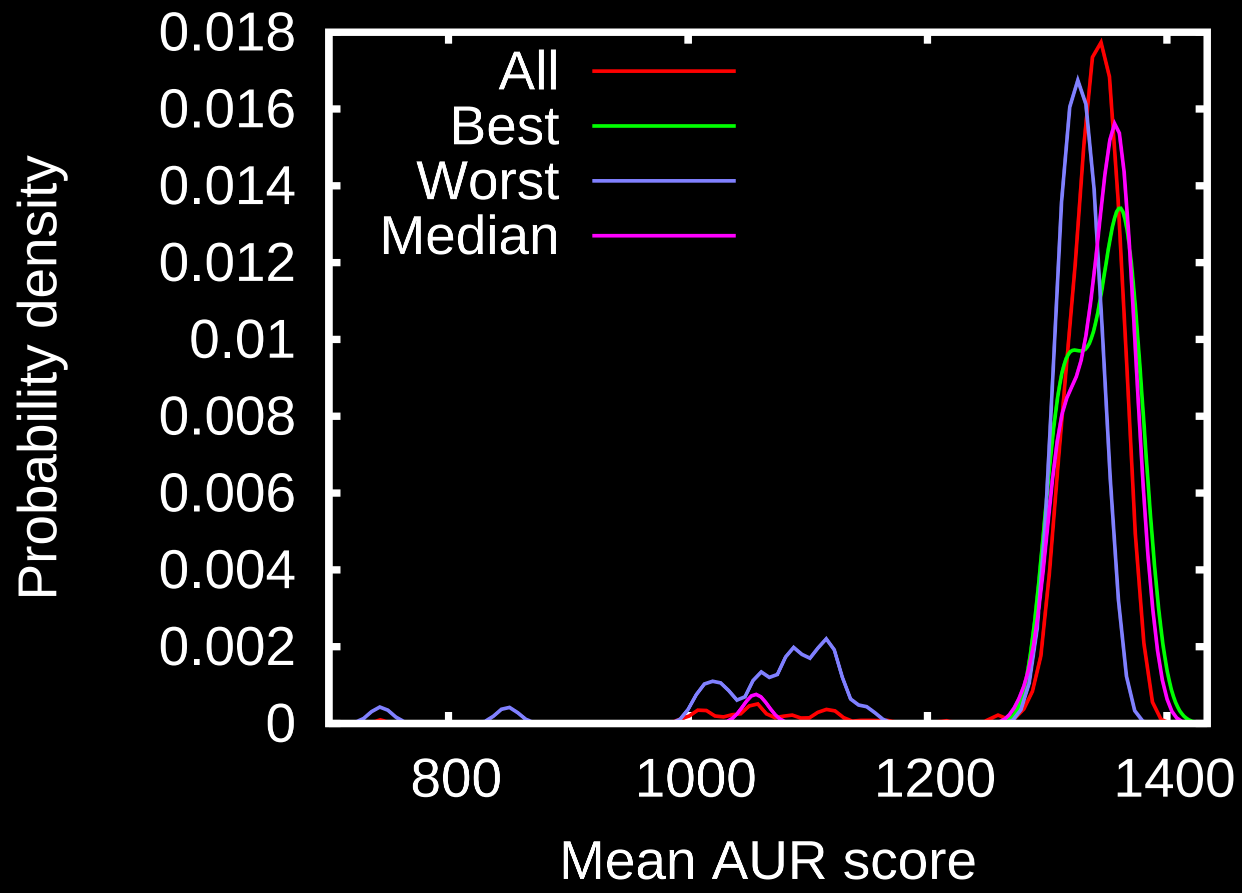
Conclusions

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

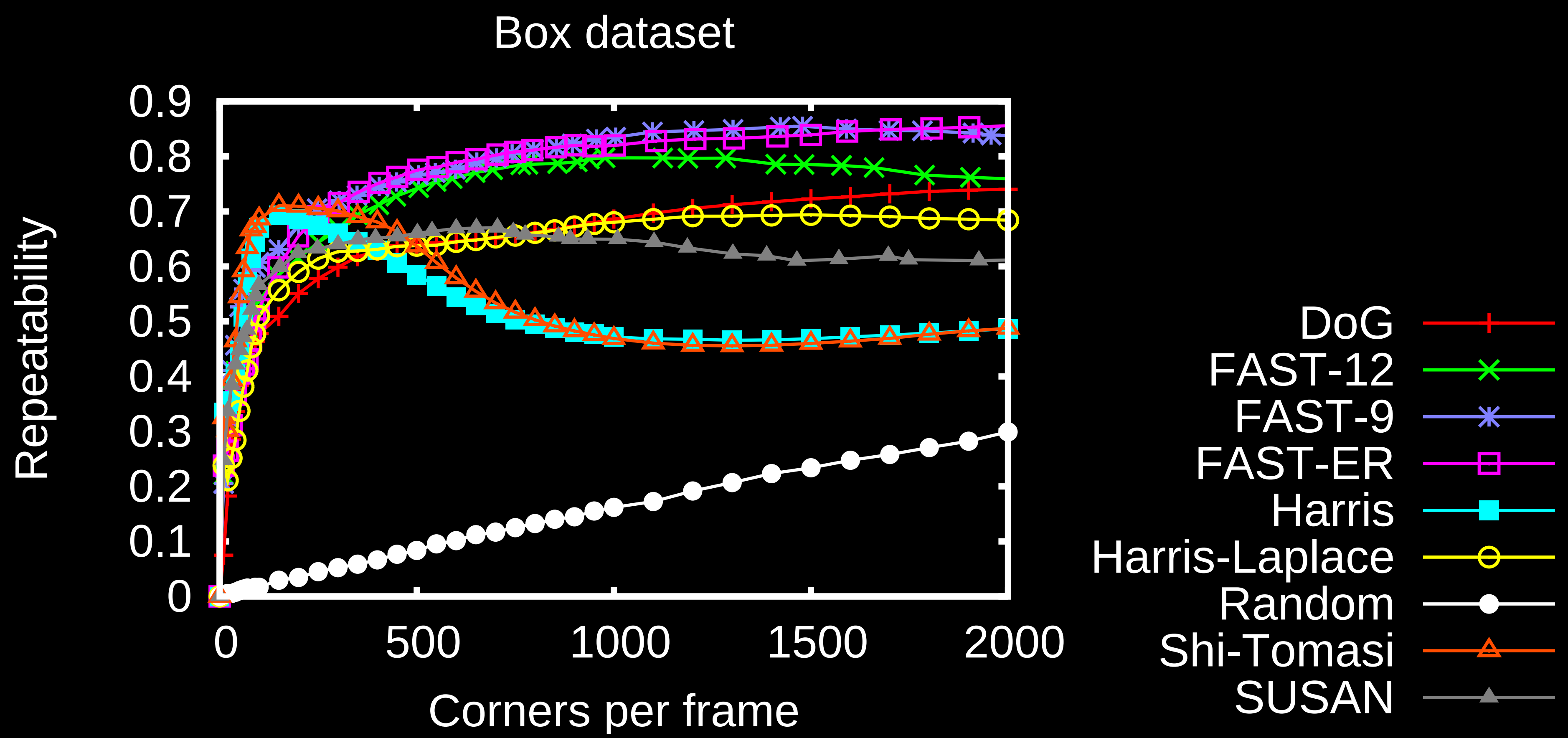
`http://users.soe.ucsc.edu/~eads/software.shtml`

More results

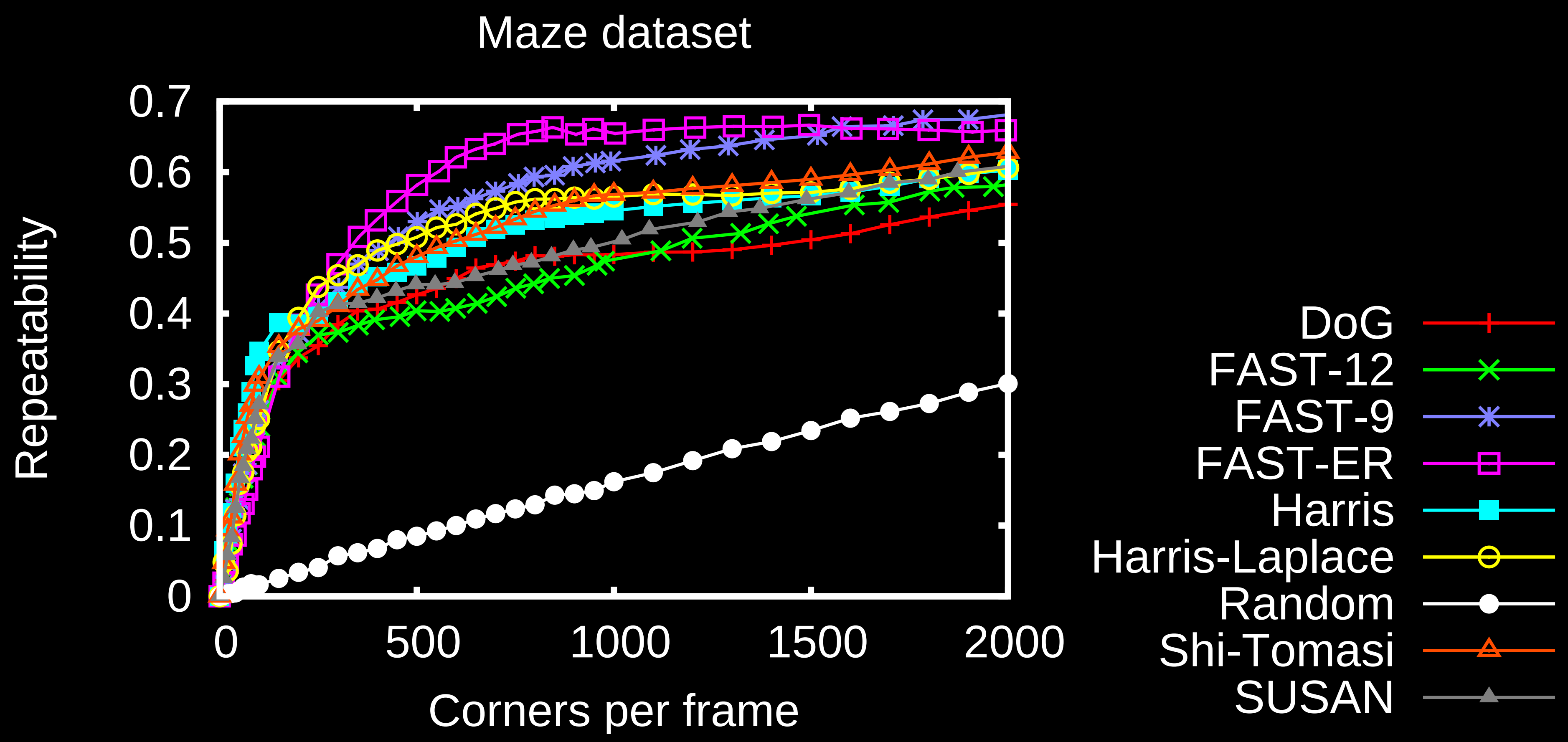
Sensitivity to w_i



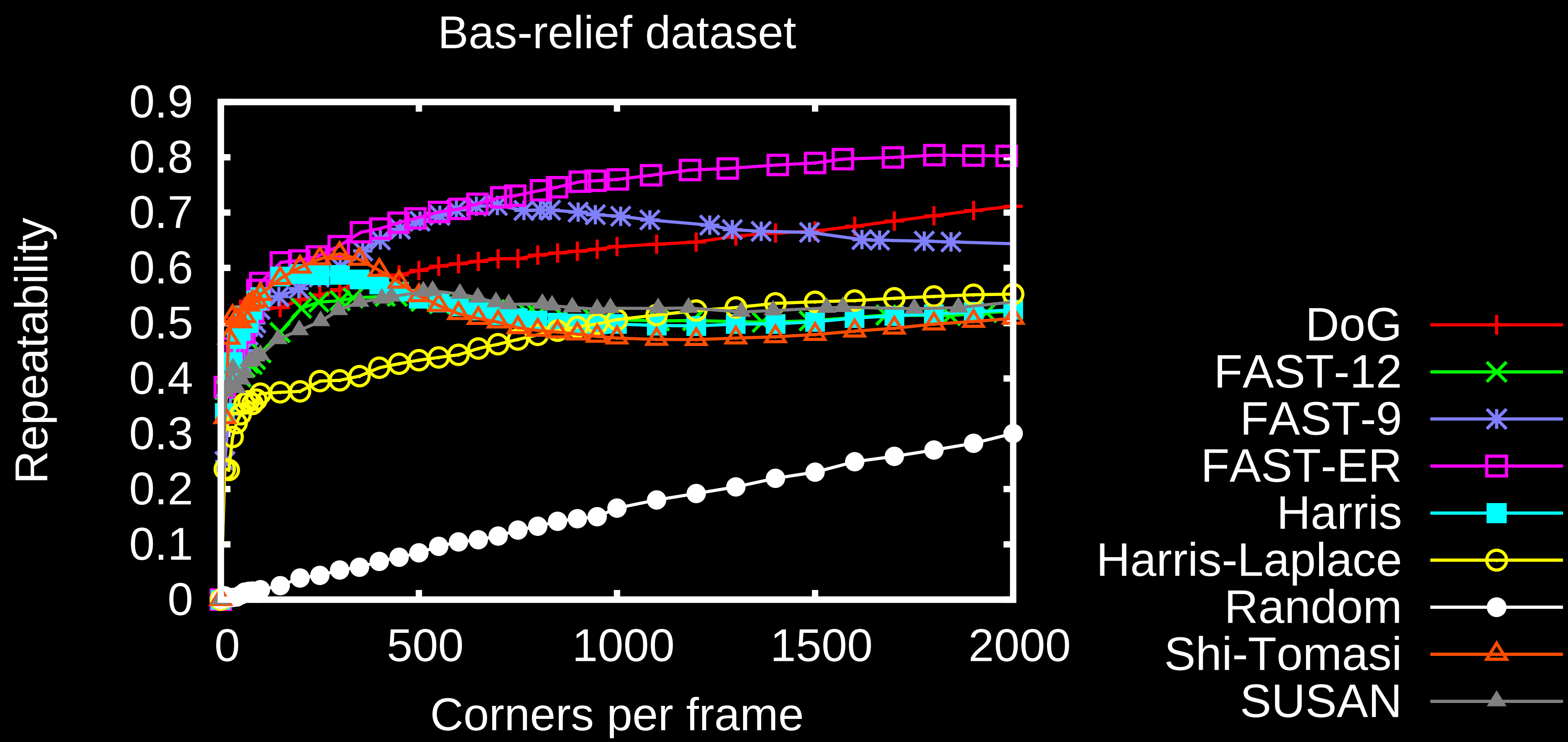
Results: Perspective (box) dataset



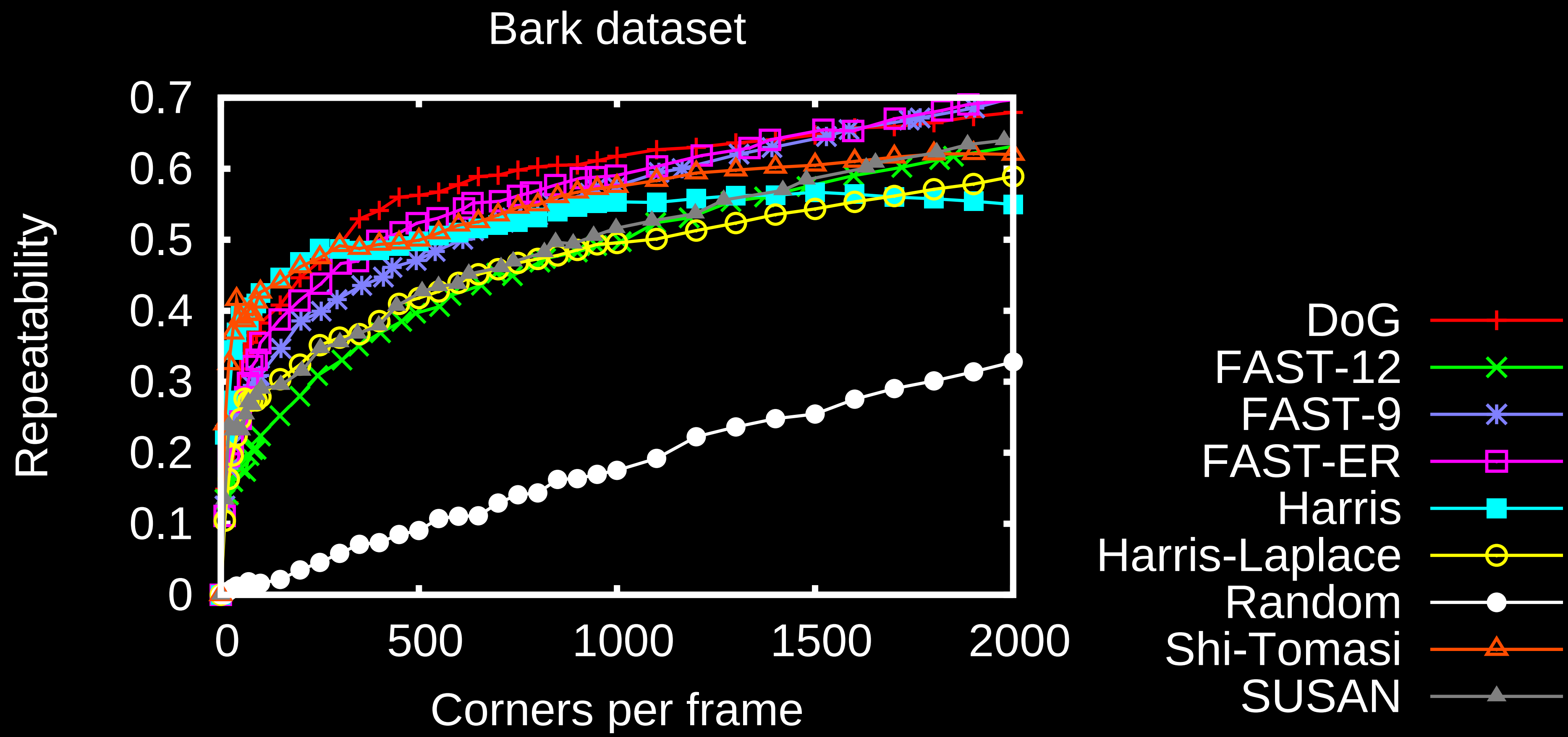
Results: Geometric dataset



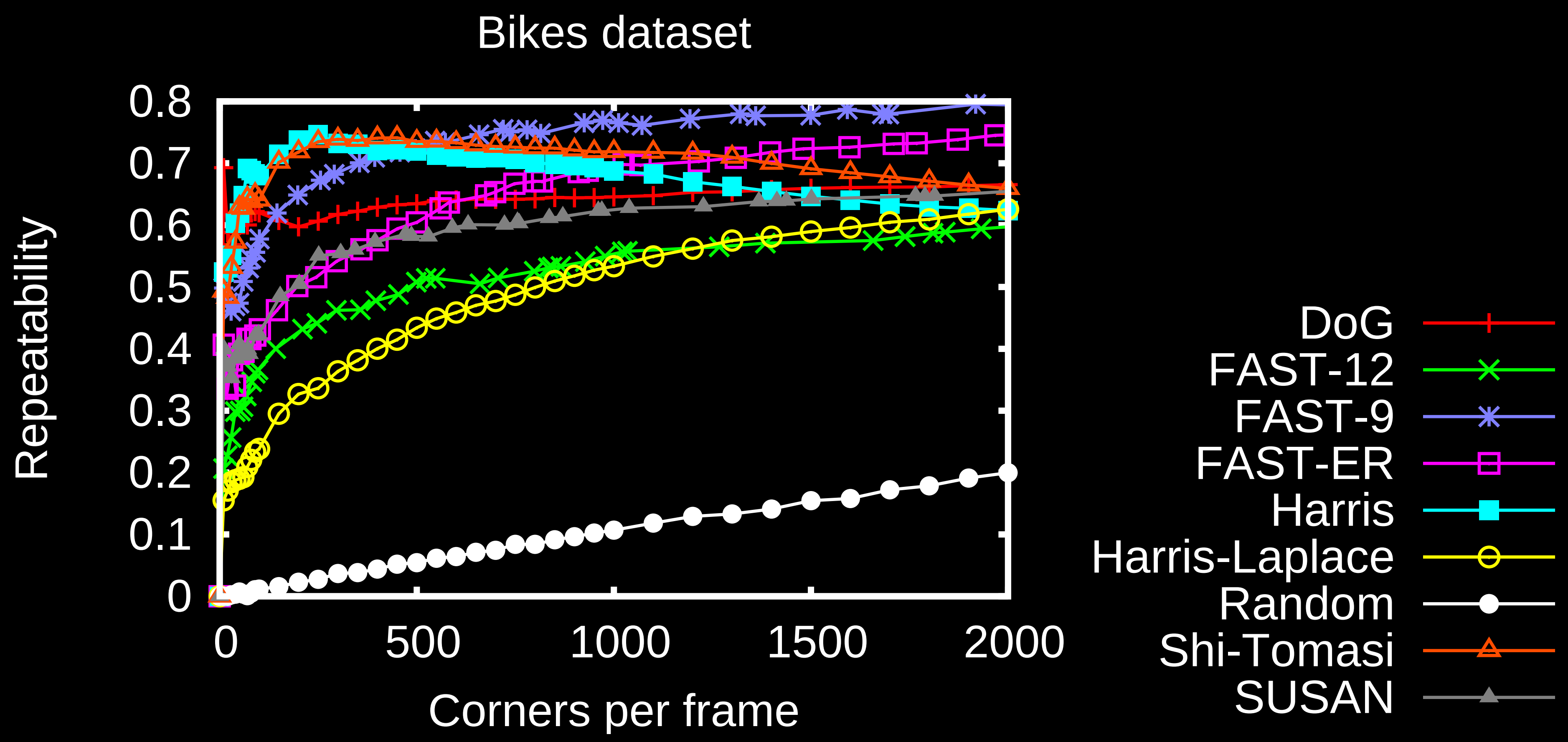
Results: Bas-relief dataset



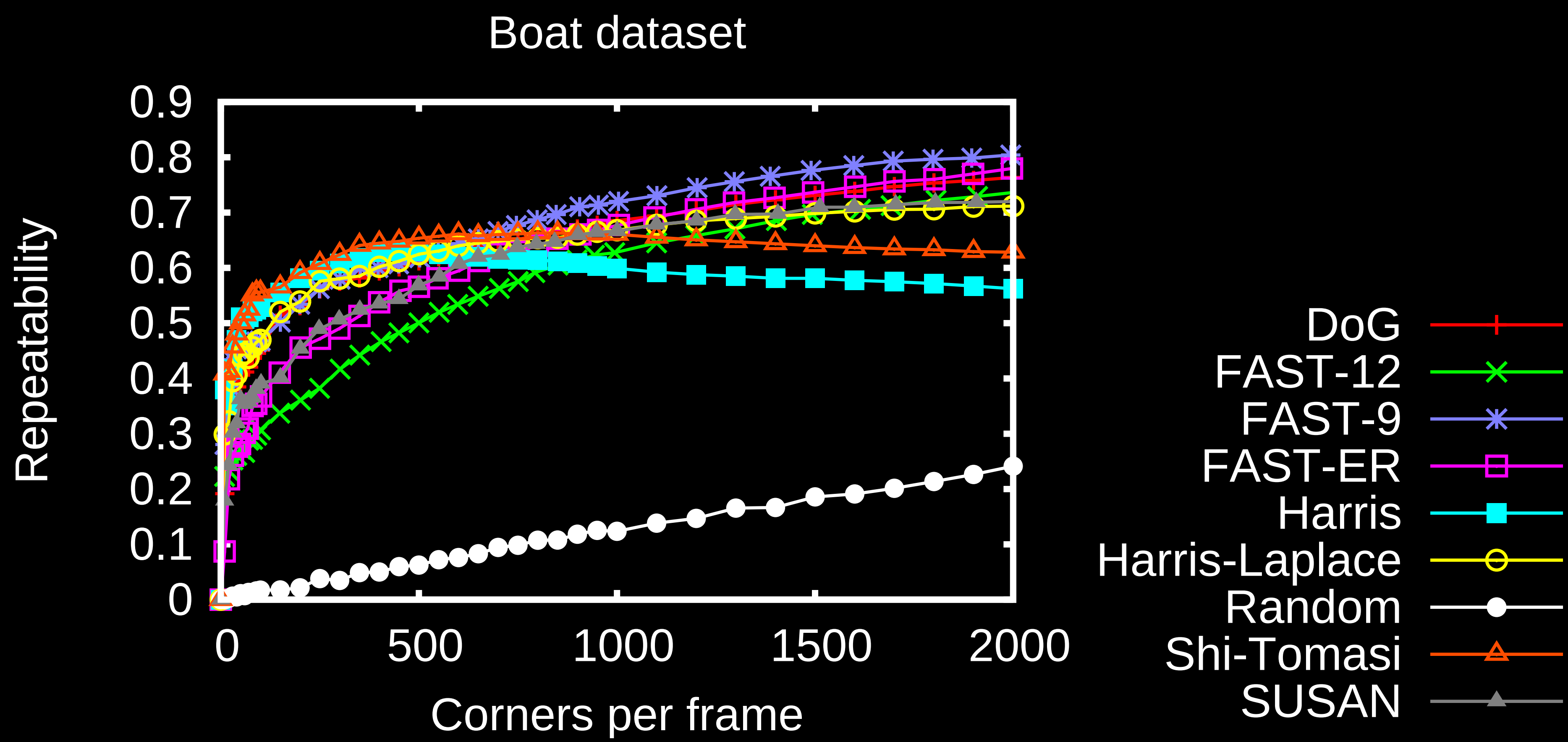
Results: Scale and rotation (bark) dataset



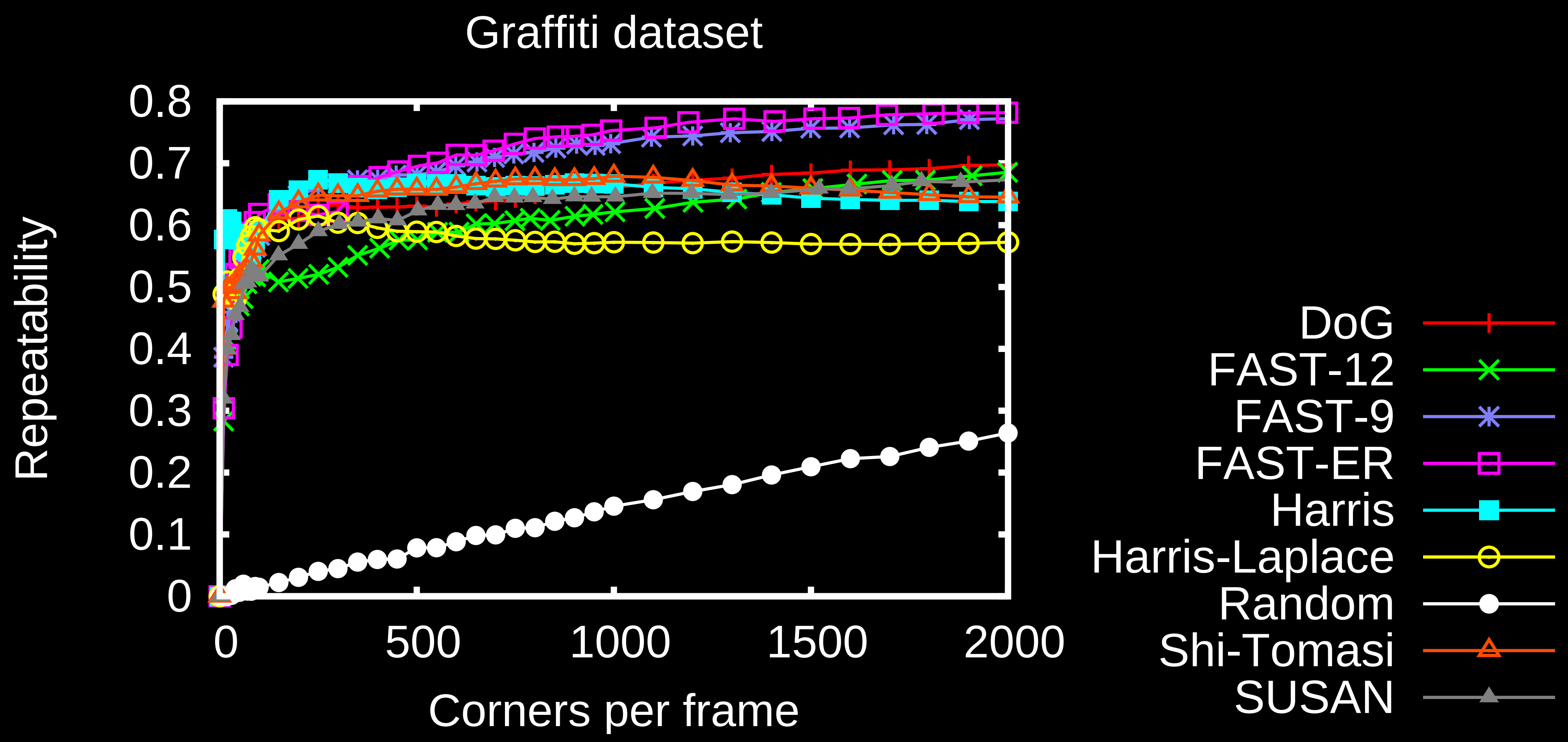
Results: Blur (bikes) dataset



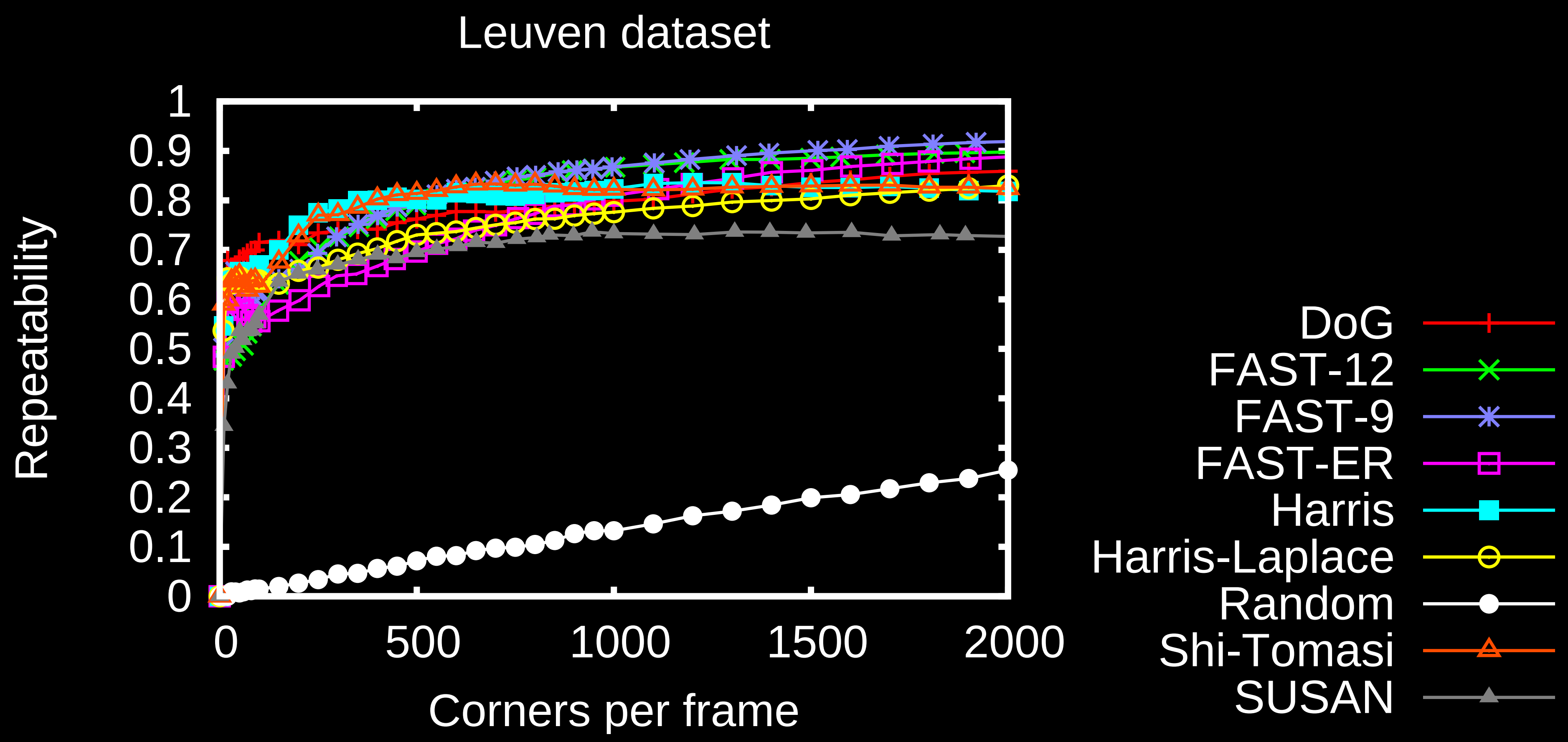
Results: Scale and rotation (boat) dataset



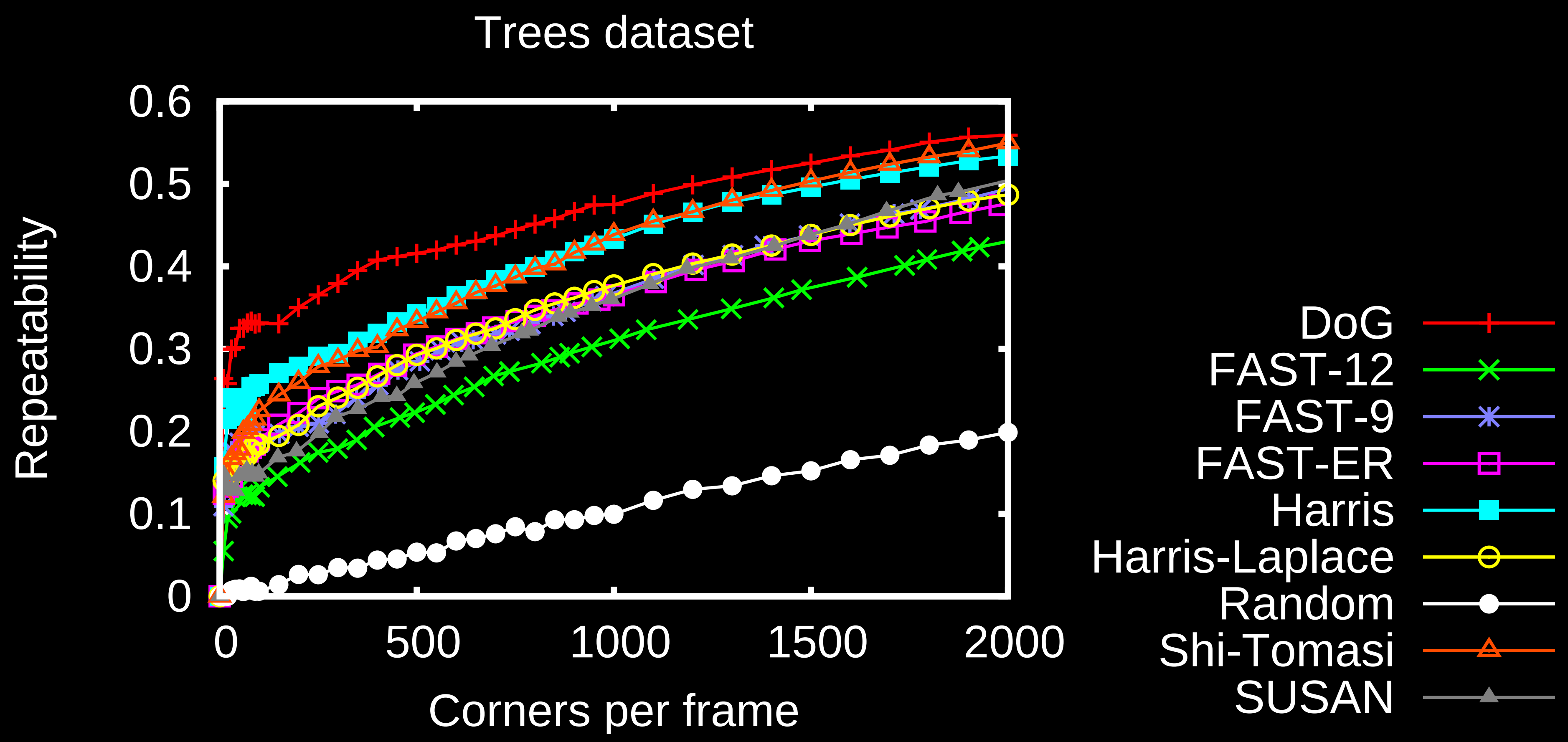
Results: Perspective (graffiti) dataset



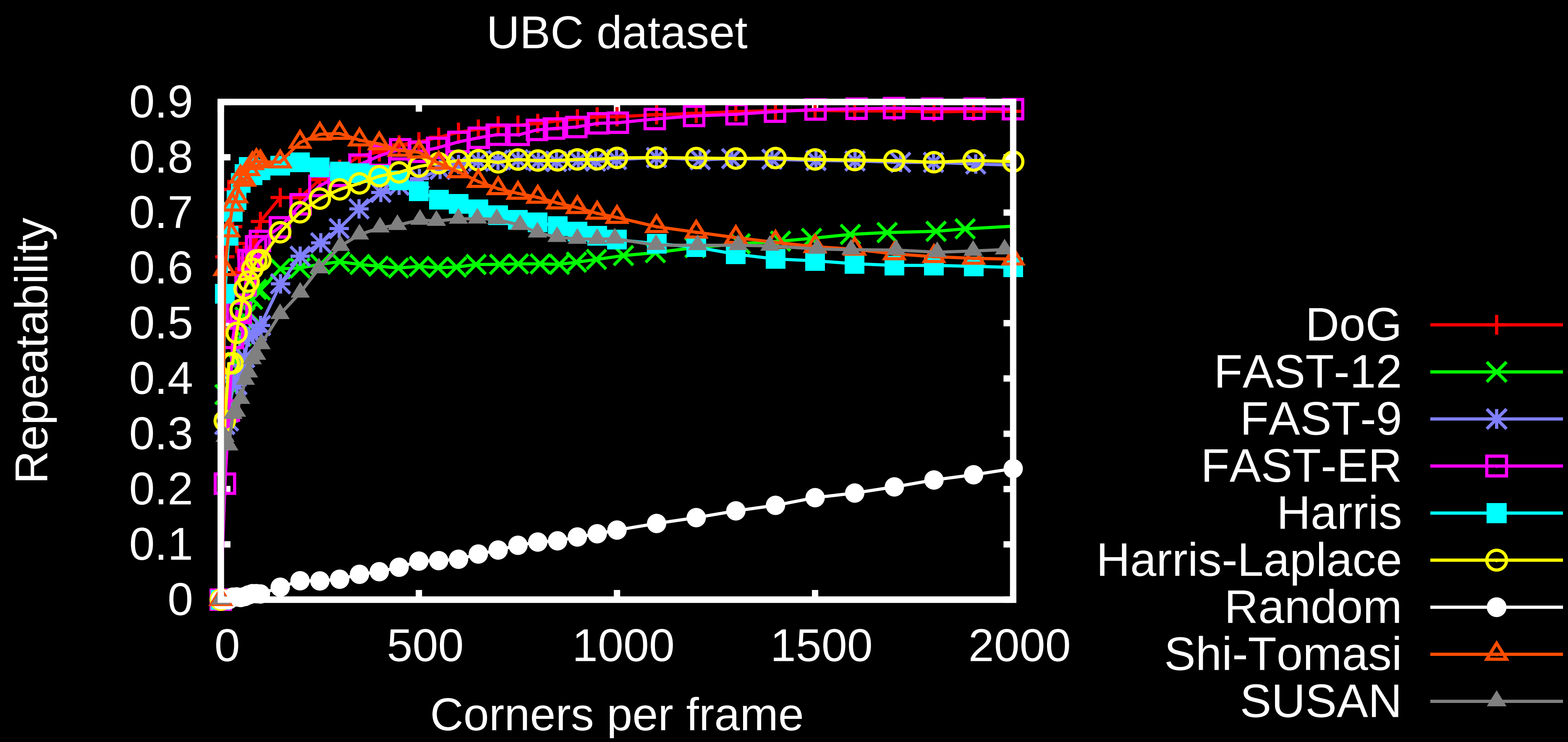
Results: Lighting dataset



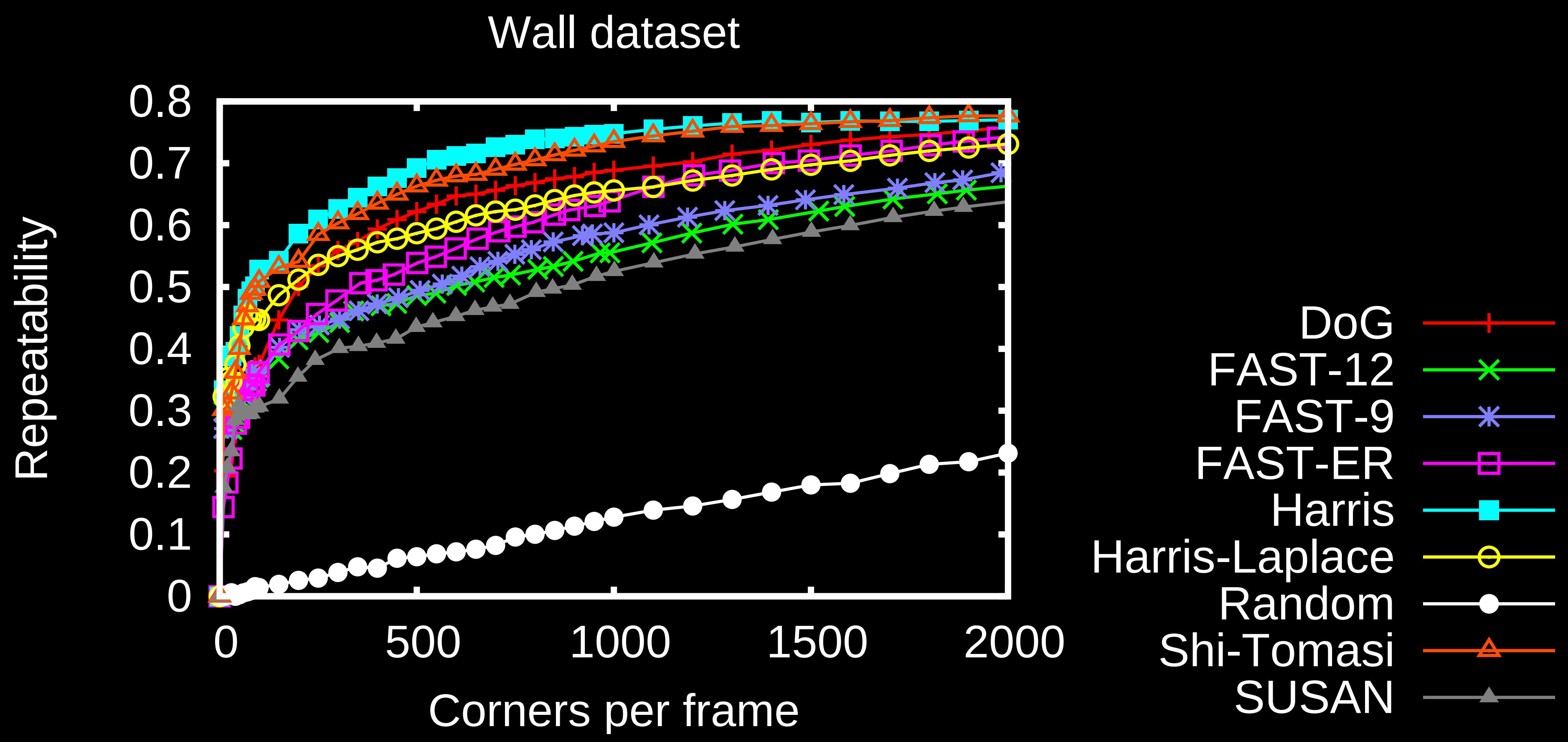
Results: Blur (trees) dataset



Results: JPEG compression dataset



Results: Perspective (wall) dataset

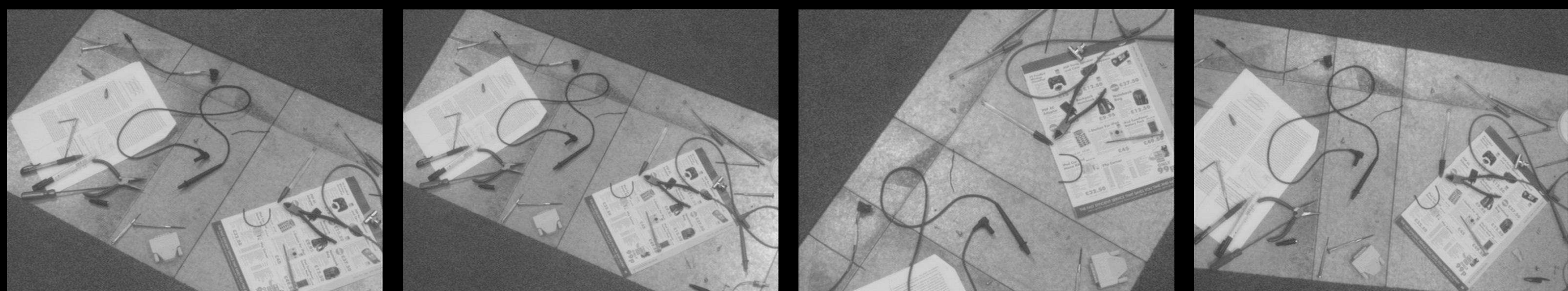


Evaluation: Datasets (3D Models)

14 images:

15 images:

8 images:



Evaluation: Homographies

6 images per set: