Software

Feature implementation and computation

Andrea Vedaldi, University of Oxford

https://sites.google.com/site/eccv12features/

goo.gl/jKC2J

VLFeat

MATLAB ease of use

Covariant detectors, MSER, SIFT, etc ...

OpenCV

One-shop experience

The most popular vision library

Others

There is certainly no lack of software!

VLFeat

MATLAB ease of use

Covariant detectors, MSER, SIFT, etc ...

OpenCV

One-shop experience

The most popular vision library

Others

There is certainly no lack of software!

VLFeat

An open and portable library of CV software

http://www.vlfeat.org/benchmarks/index.html

Quick Start with MATLAB

```
>> untar http://www.vlfeat.org/download/vlfeat-0.9.16-bin.tar.gz
>> run vlfeat-0.9.16/toolbox/vl_setup
```


Mac


Linux


Windows

Since 2008
ACM OSS award

SIFT detector and descriptor

1. **load an image**

```
imPath = fullfile('oxford.jpg') ;
im = imread(imPath) ;
```

2. **convert it to single precision gray scale**

```
imgs = im2single(rgb2gray(im)) ;
```

3. **run SIFT**

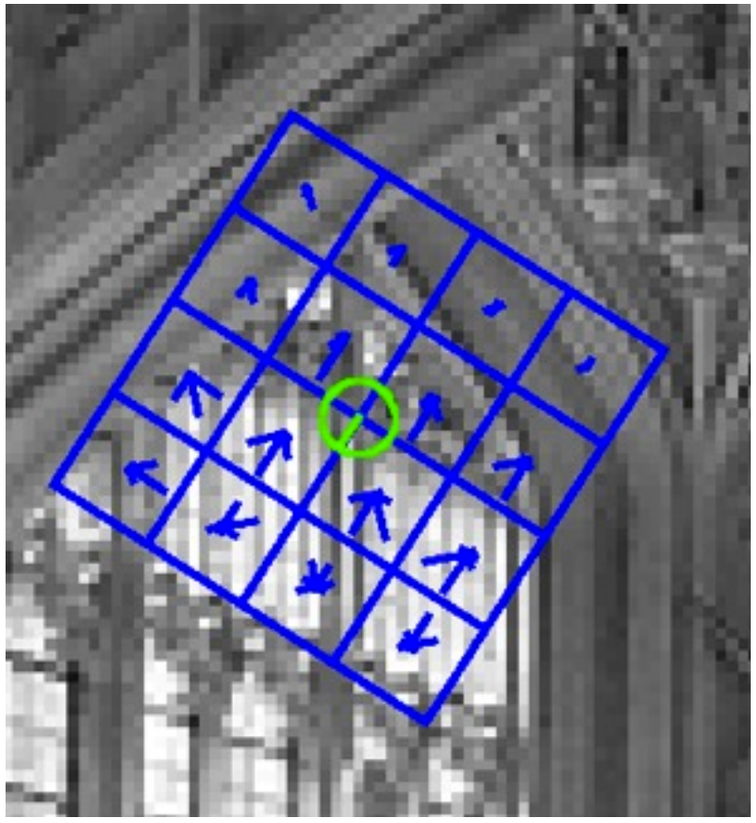
```
[frames, descrs] = vl_sift(imgs) ;
```

4. **visualise keypoints**

```
imagesc(im) ; colormap gray; hold on ;
vl_plotframe(frames) ;
```

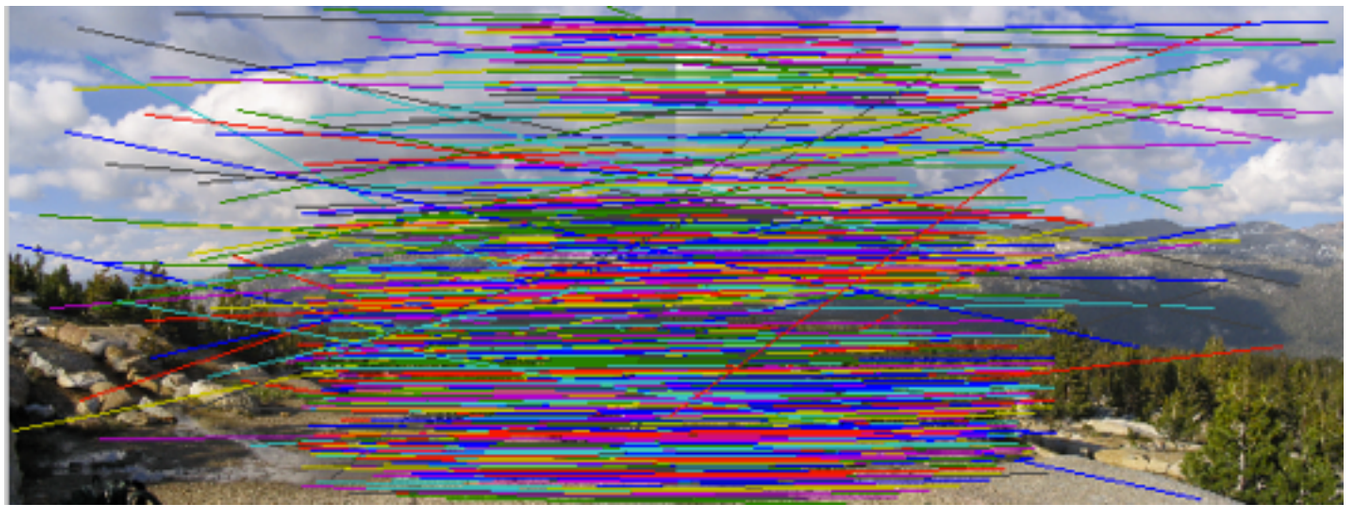
5. **visualise descriptors**

```
vl_plotsiftdescriptor(...
  descrs(:,432), frames(:,432)) ;
```



Example: matching features

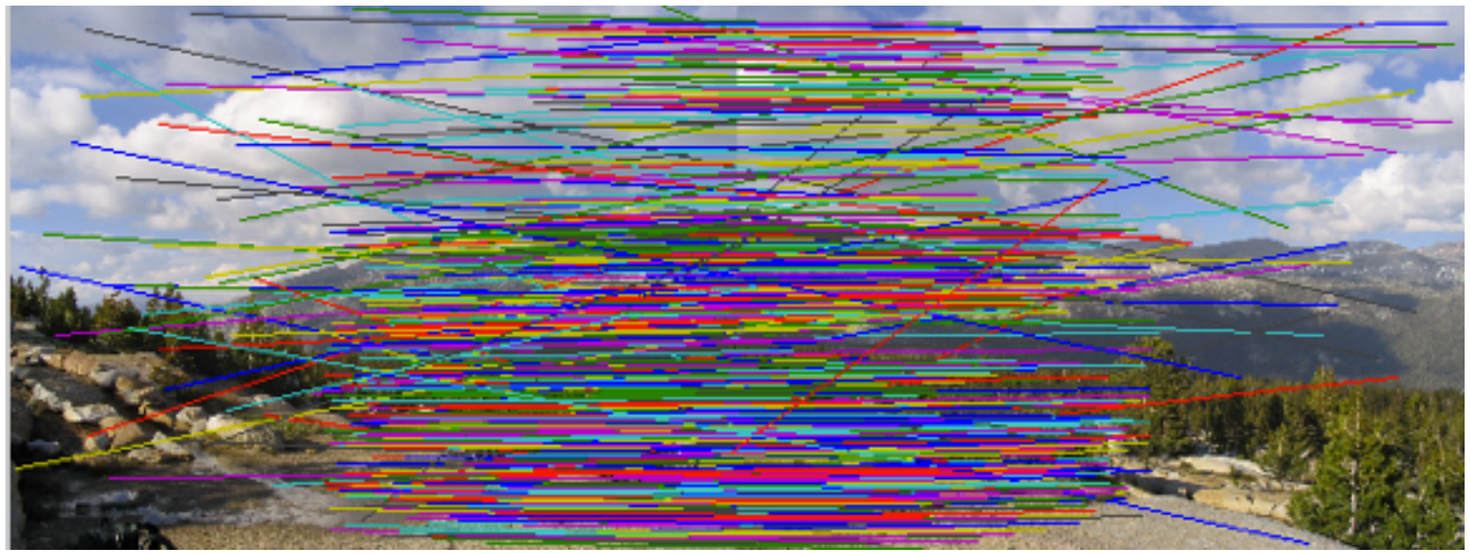
684 tentative matches



- ```
  [f1,d1] = vl_sift(im1) ;
  [f2,d2] = vl_sift(im2) ;
  [matches, scores] = vl_ubcmatch(d1,d2) ;
  ```
  ```

Example: stitching

684 tentative matches



492 (76%) inliers

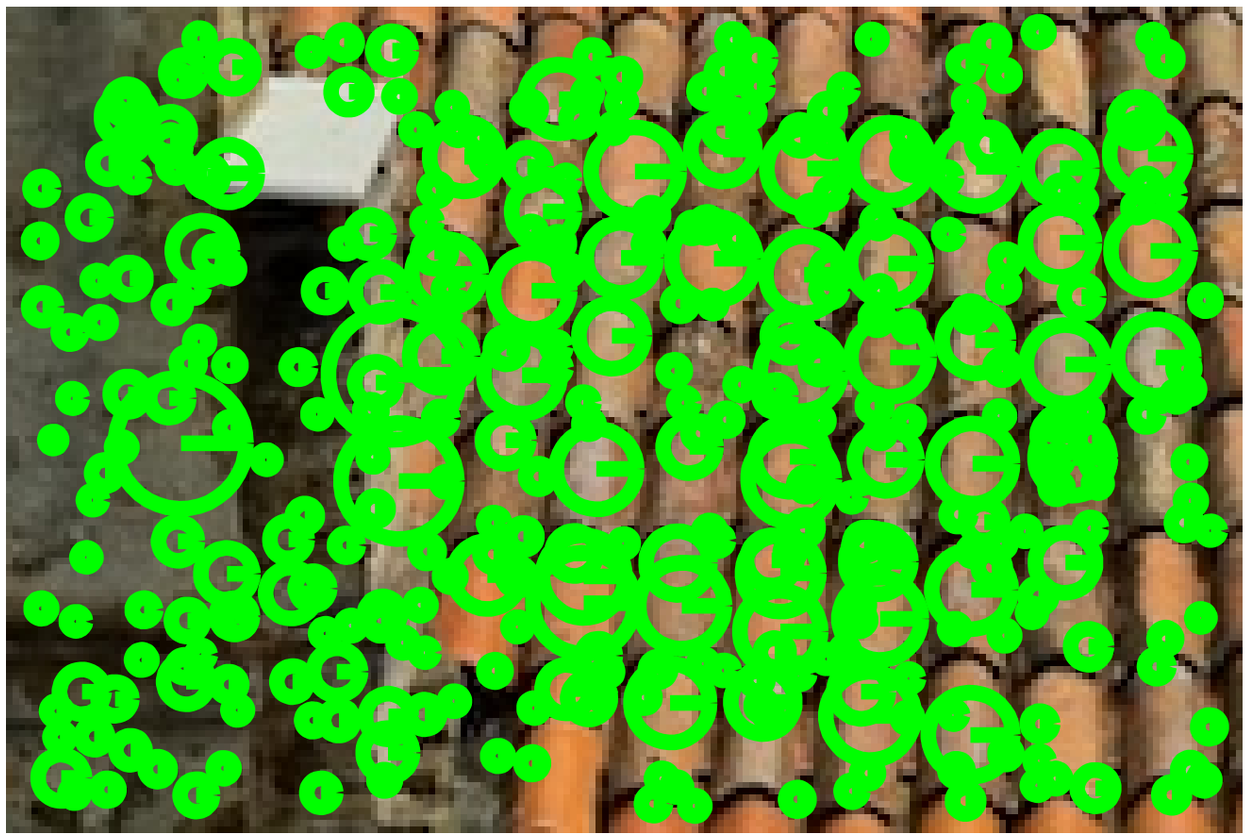
mosaic

- Live demonstration

Other covariant detectors


```
frames = vl_covdet(imgs, 'verbose') ;
```

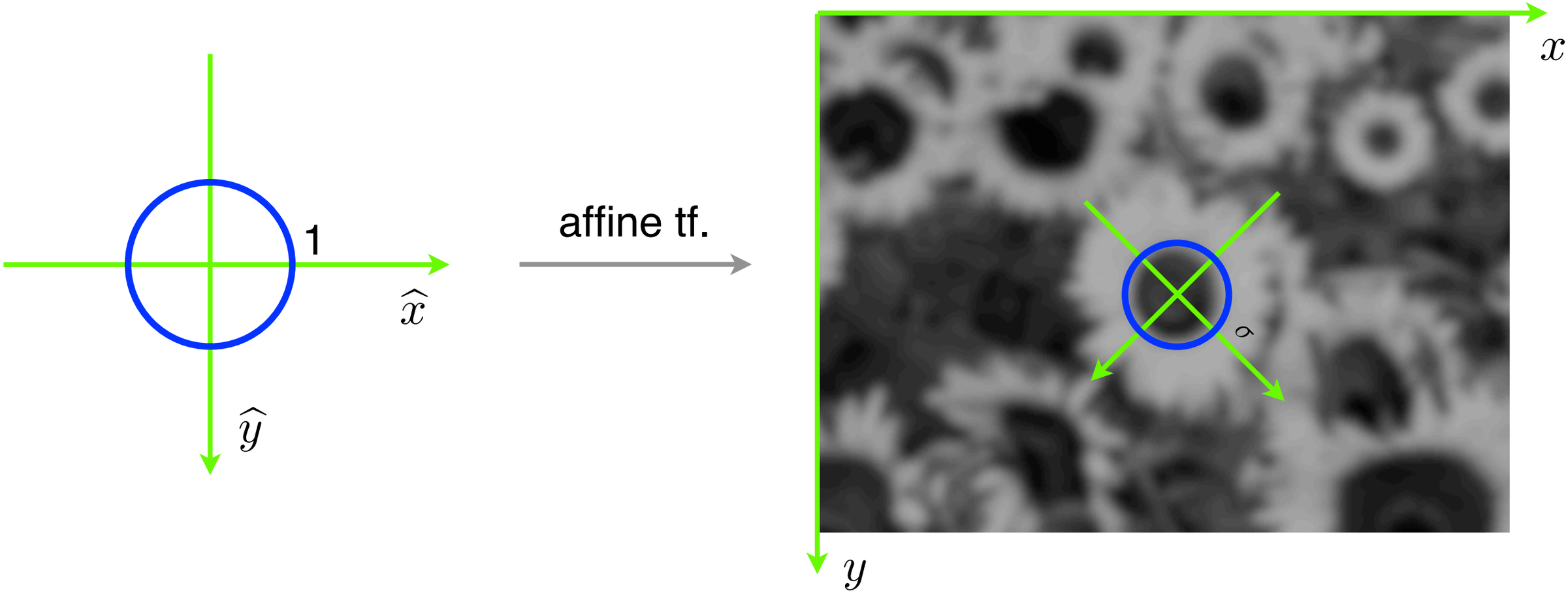
```
imagesc(im) ;
```



```
vl_plotframes(frames) ;
```

Do not forget to convert the images to grayscale in single precision. Use `im2single(im)` and not `single(im)` !

Understanding feature frames



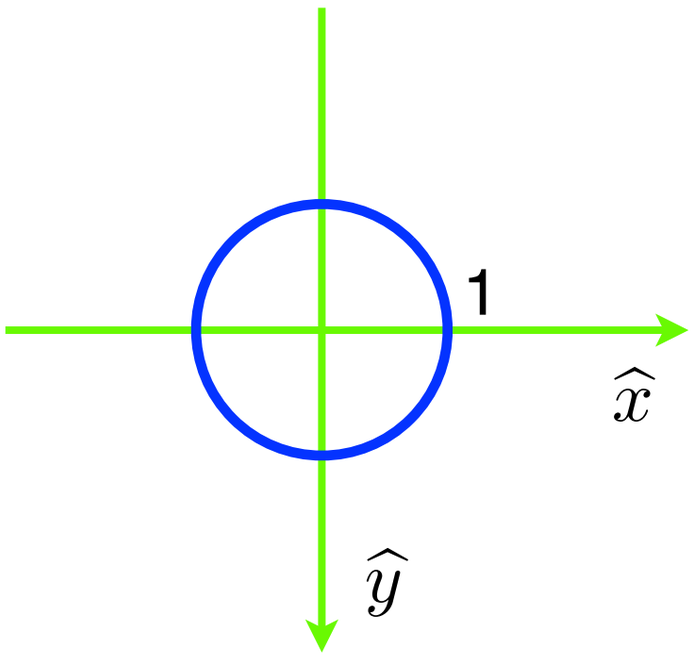
- **`frames`** has a feature frame for each column containing a stacked affine transformation (A,T)

$$\texttt{frames(:,i)} = \begin{bmatrix} t_1 \\ t_2 \\ a_{11} \\ a_{21} \\ a_{12} \\ a_{22} \end{bmatrix}$$

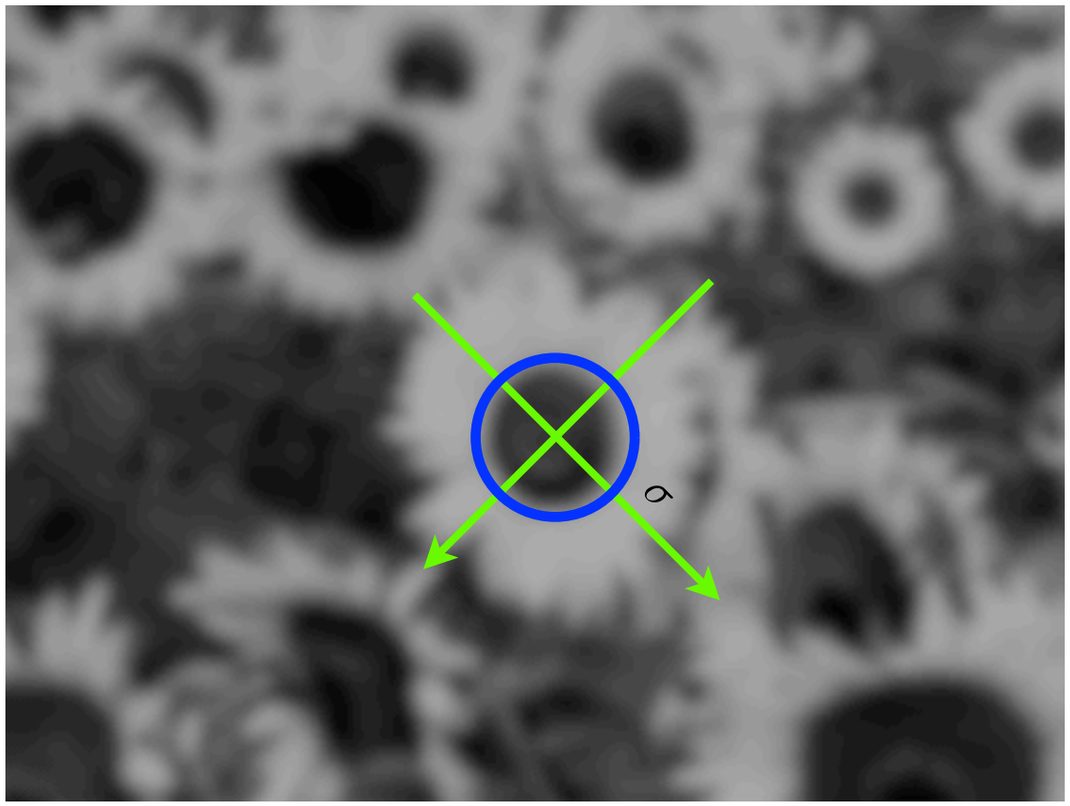
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \widehat{x} \\ \widehat{y} \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

A frame as an oriented ellipse

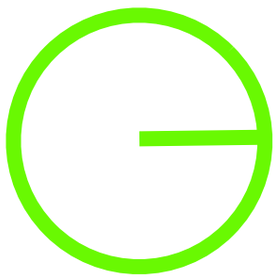
reference = two vectors

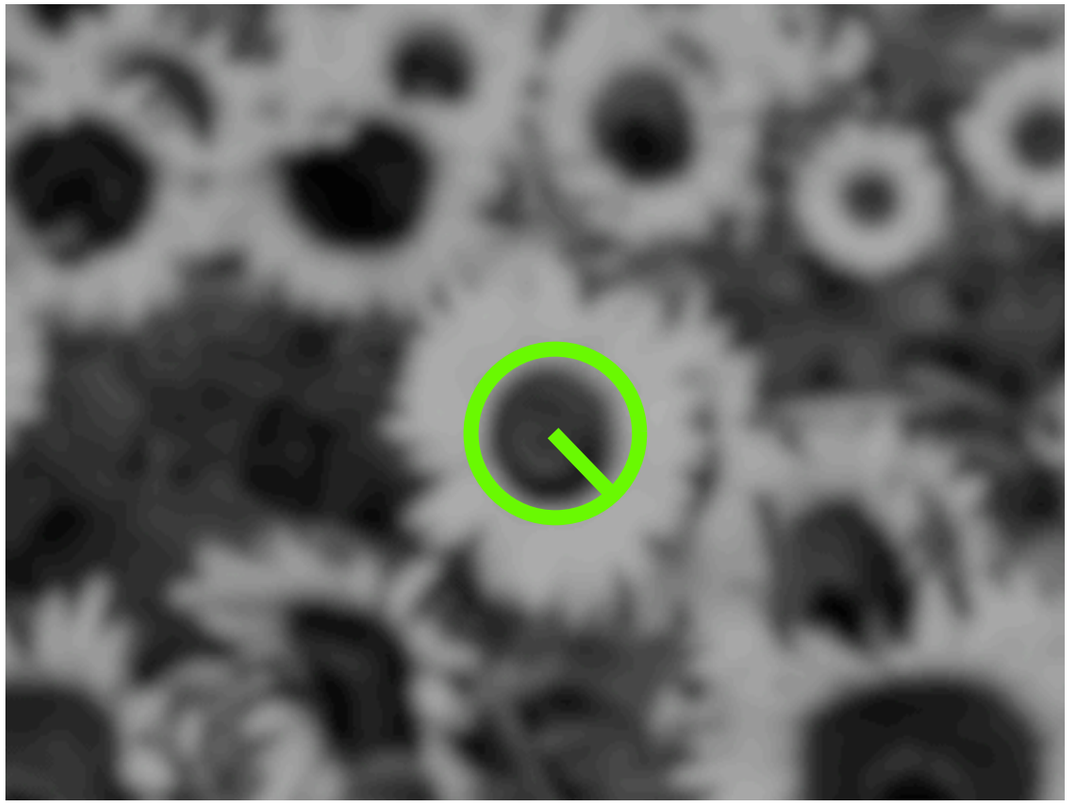


uniquely identifies the affine transformation



reference = oriented ellipse

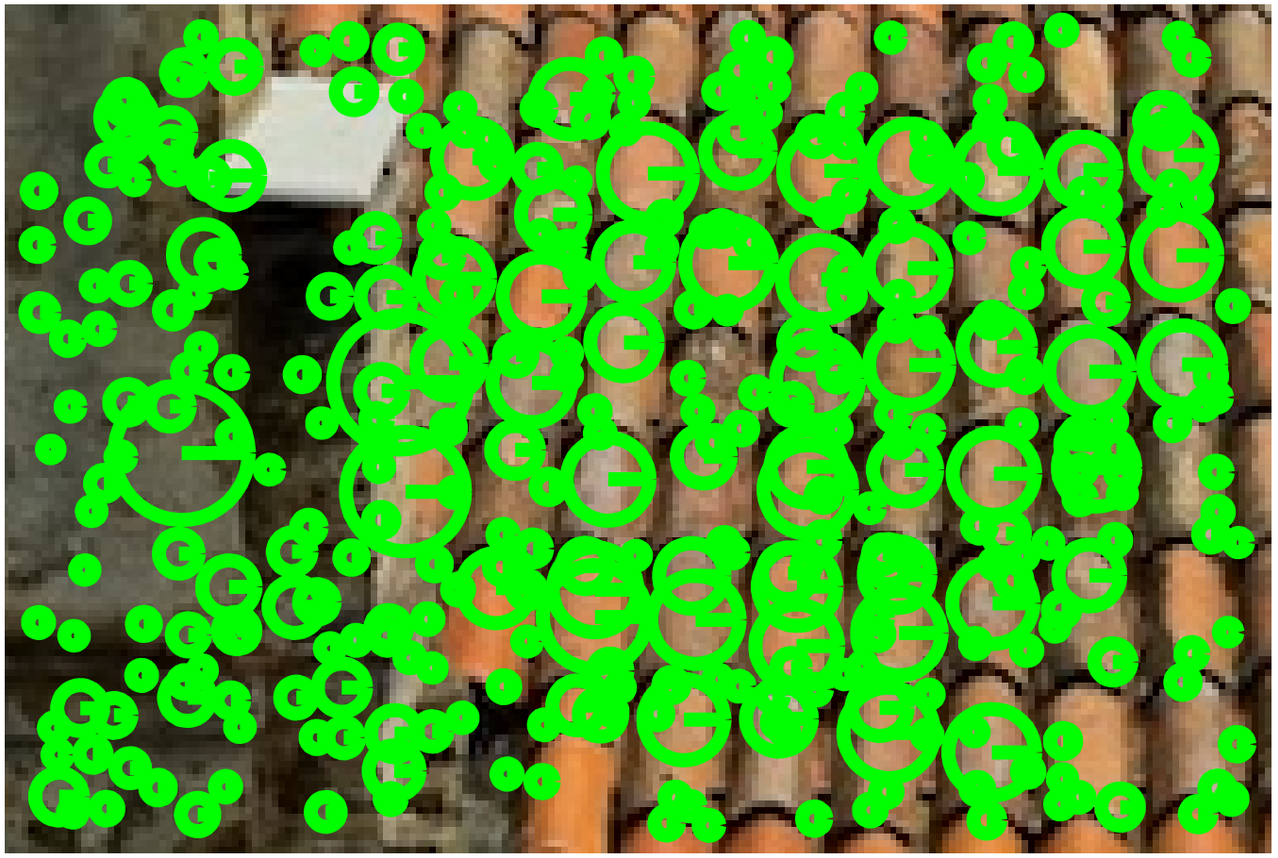




Different cornerness measures


```
frames = vl_covdet(imgs, 'method', 'Hessian') ;
```

DoG



Hessian

- Difference of Gaussian (Laplacian, trace of Hessian, or SIFT)
- Hessian (determinant of)

Different cornerness measures


```
frames = vl_covdet(imgs, 'method', 'HarrisLaplace') ;
```

HarrisLaplace

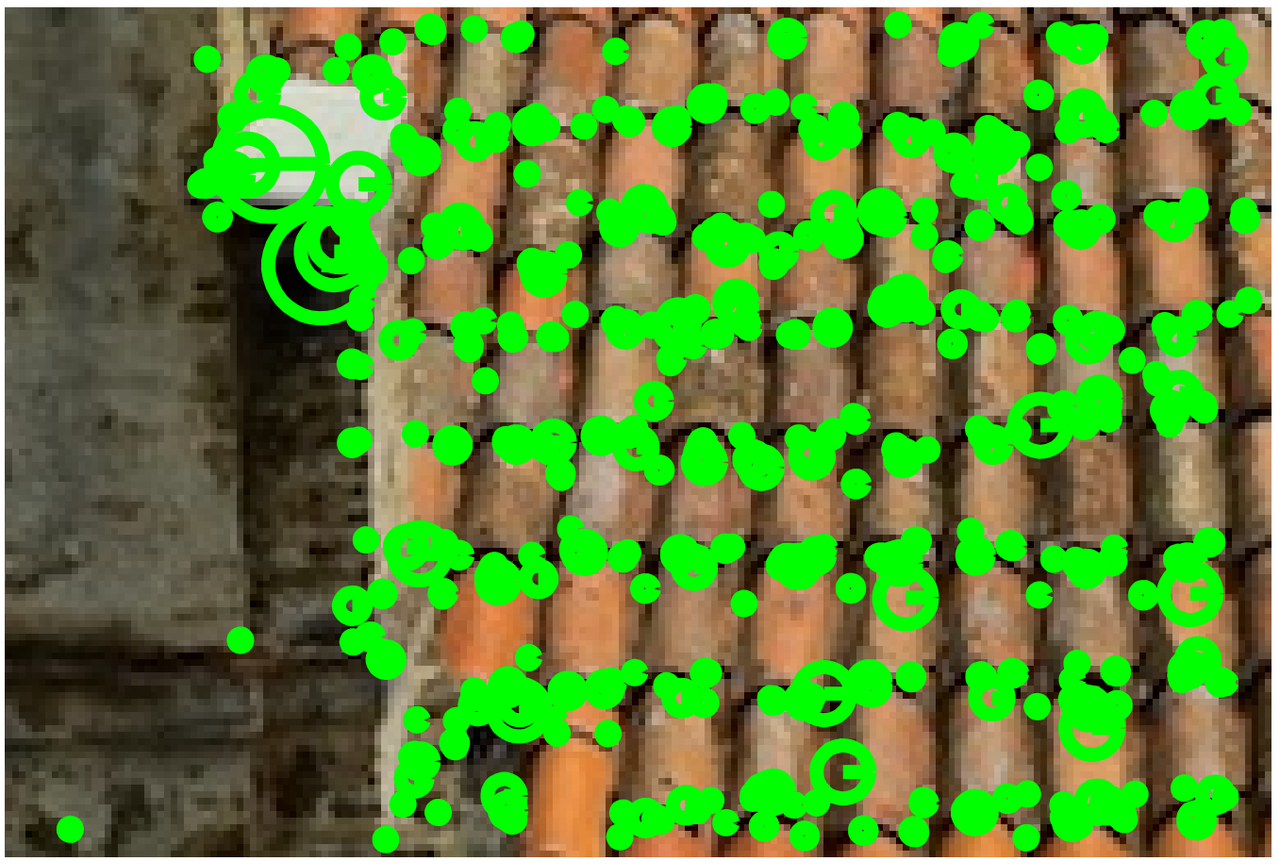
HessianLaplace

- Harris Laplace (Harris cornerness, Laplacian for scale selection)
- Hessian Laplace

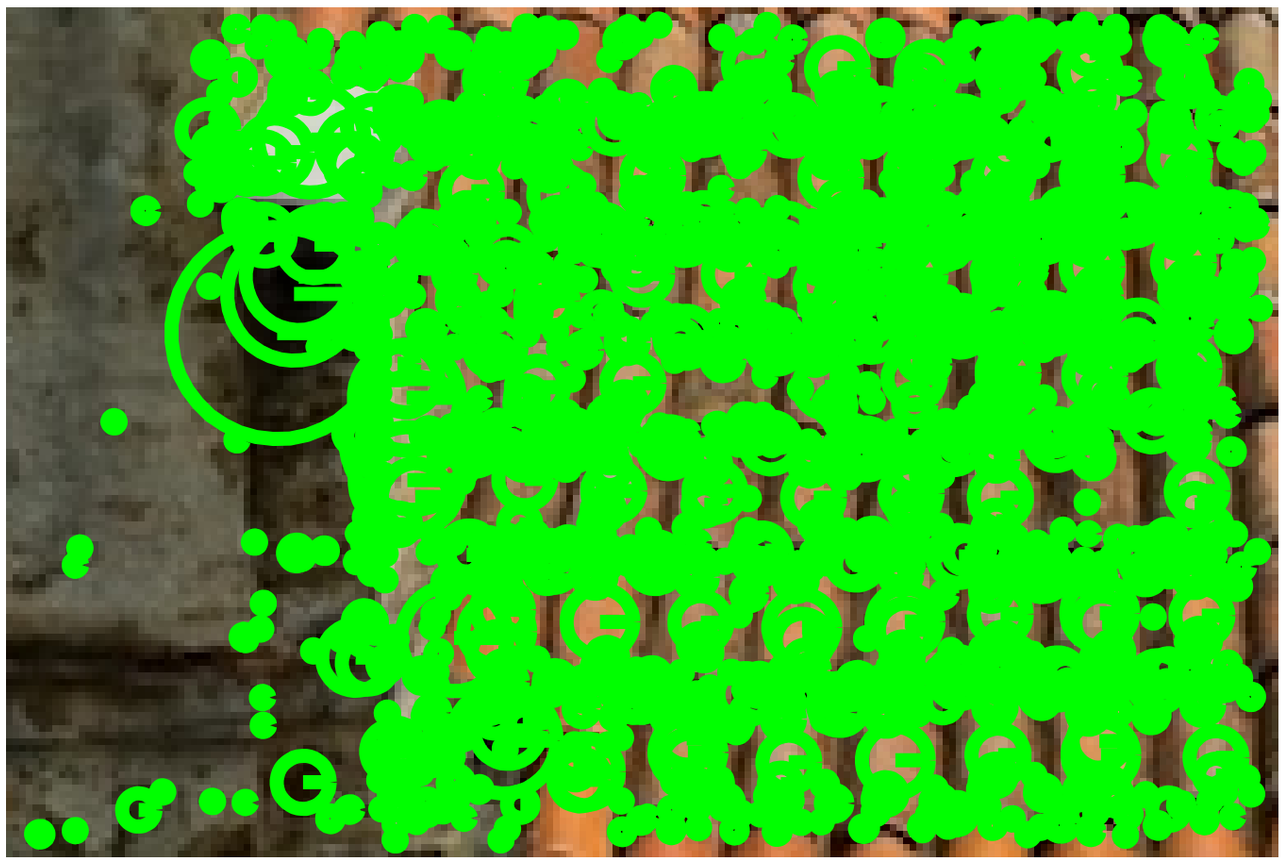
Different cornerness measures


```
frames = vl_covdet(imgs, 'method', 'MultiscaleHarris') ;
```

MultiscaleHarris



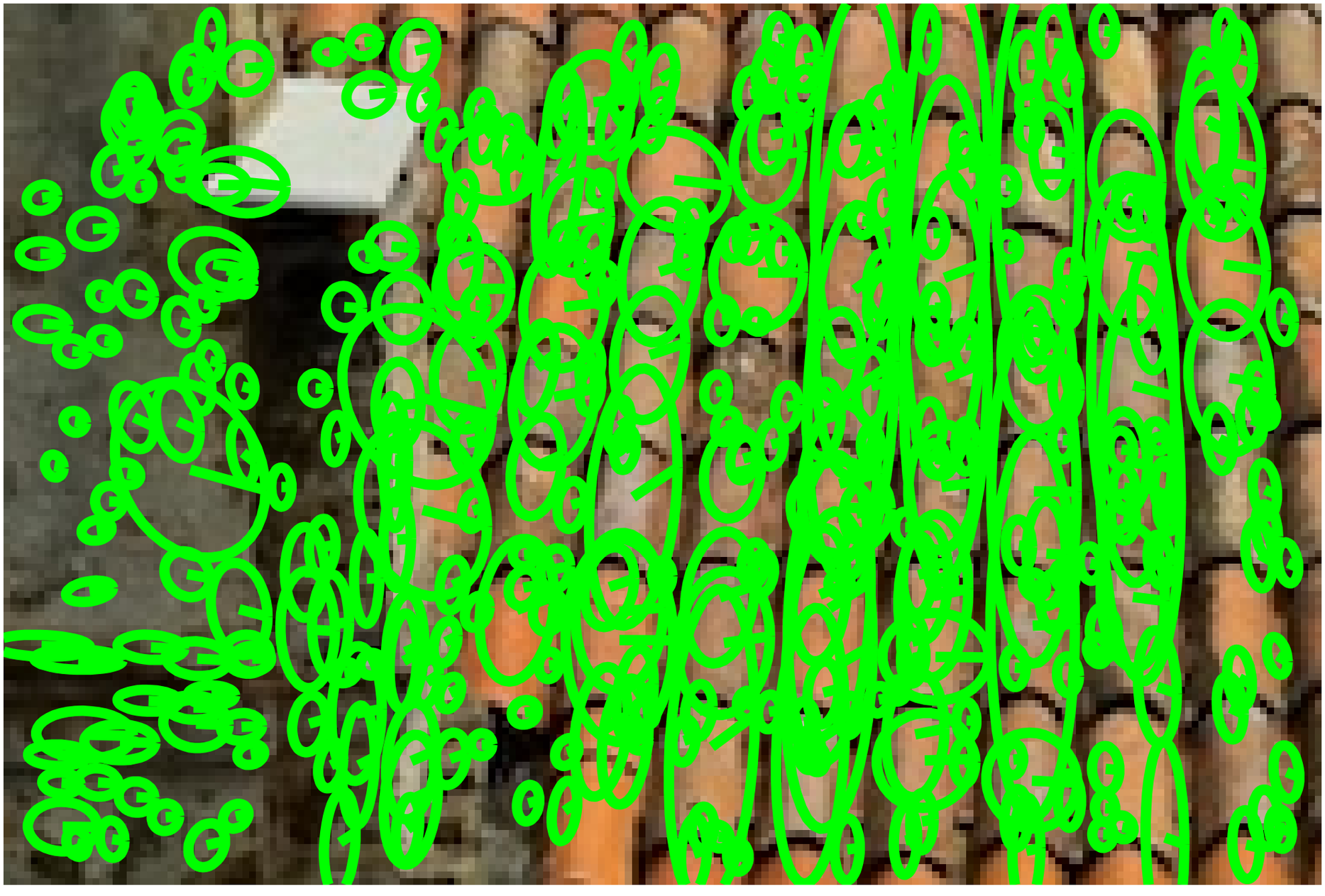
MultiscaleHessian



- Multiscale Harris (just a bunch of Harris)
- Multiscale Hessian

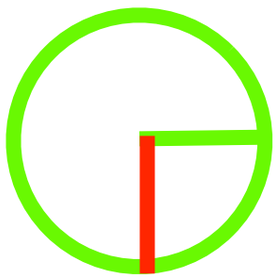
Affine adaptation

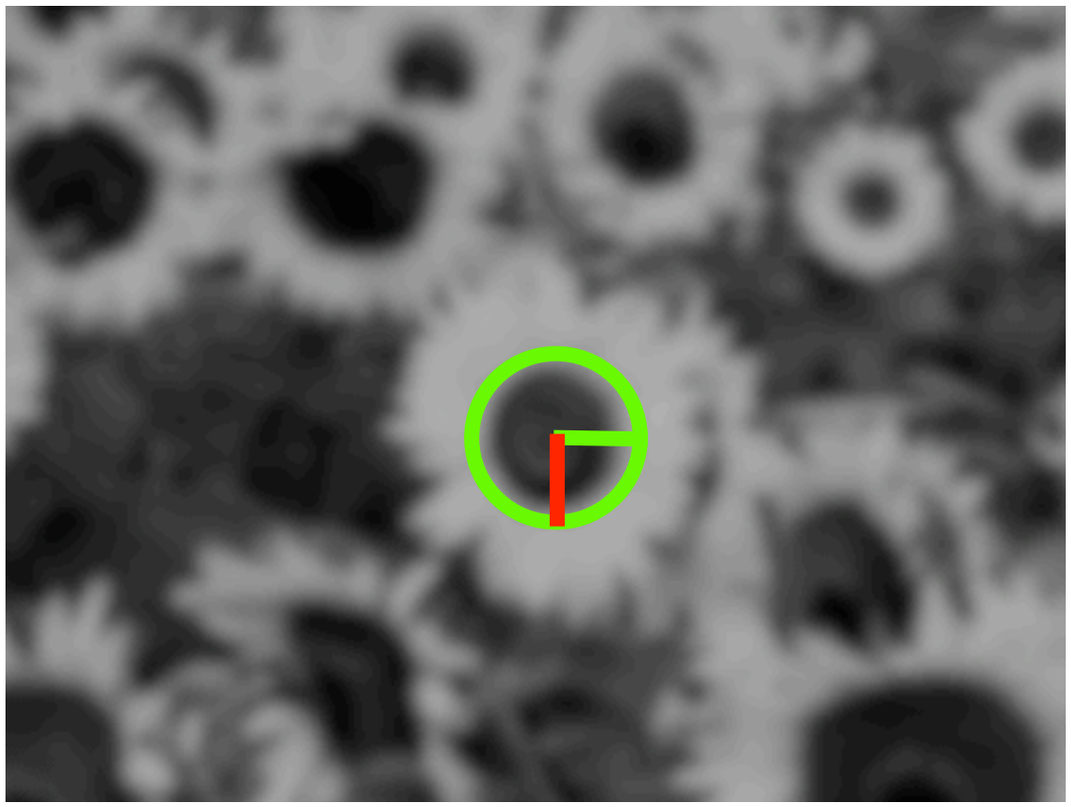

```
frames = vl_covdet(imgs, 'EstimateAffineShape', true) ;
```



Upright features

By default the rotation is selected so that the vertical axis stays vertical.





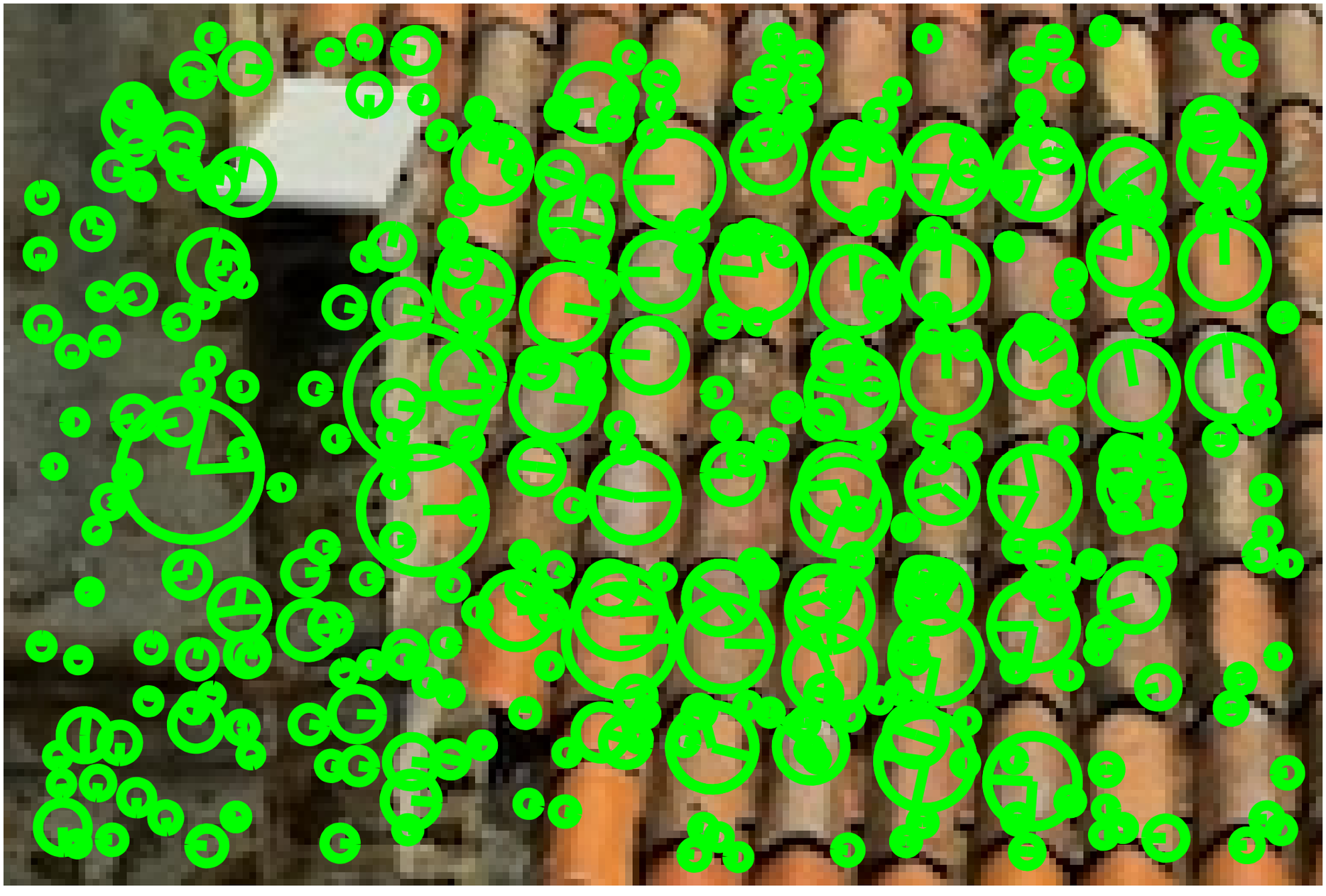
In formulas

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \widehat{x} \\ \widehat{y} \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Remark: for general affine transformations the horizontal direction may appear rotated even for upright features.

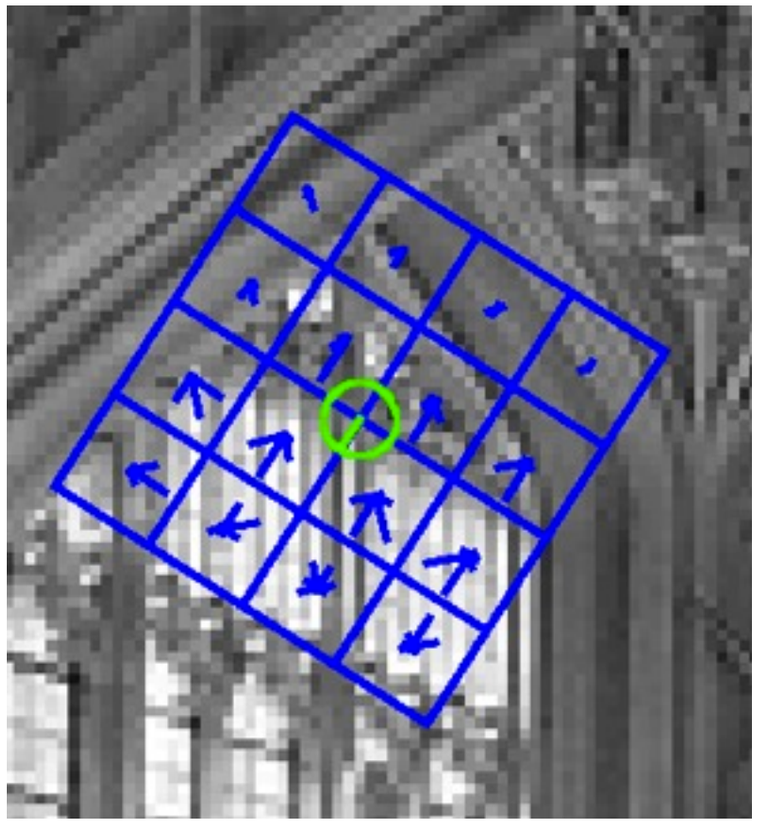
Orientation detection


```
frames = vl_covdet(imgs, 'EstimateOrientations', true) ;
```



Computing descriptors

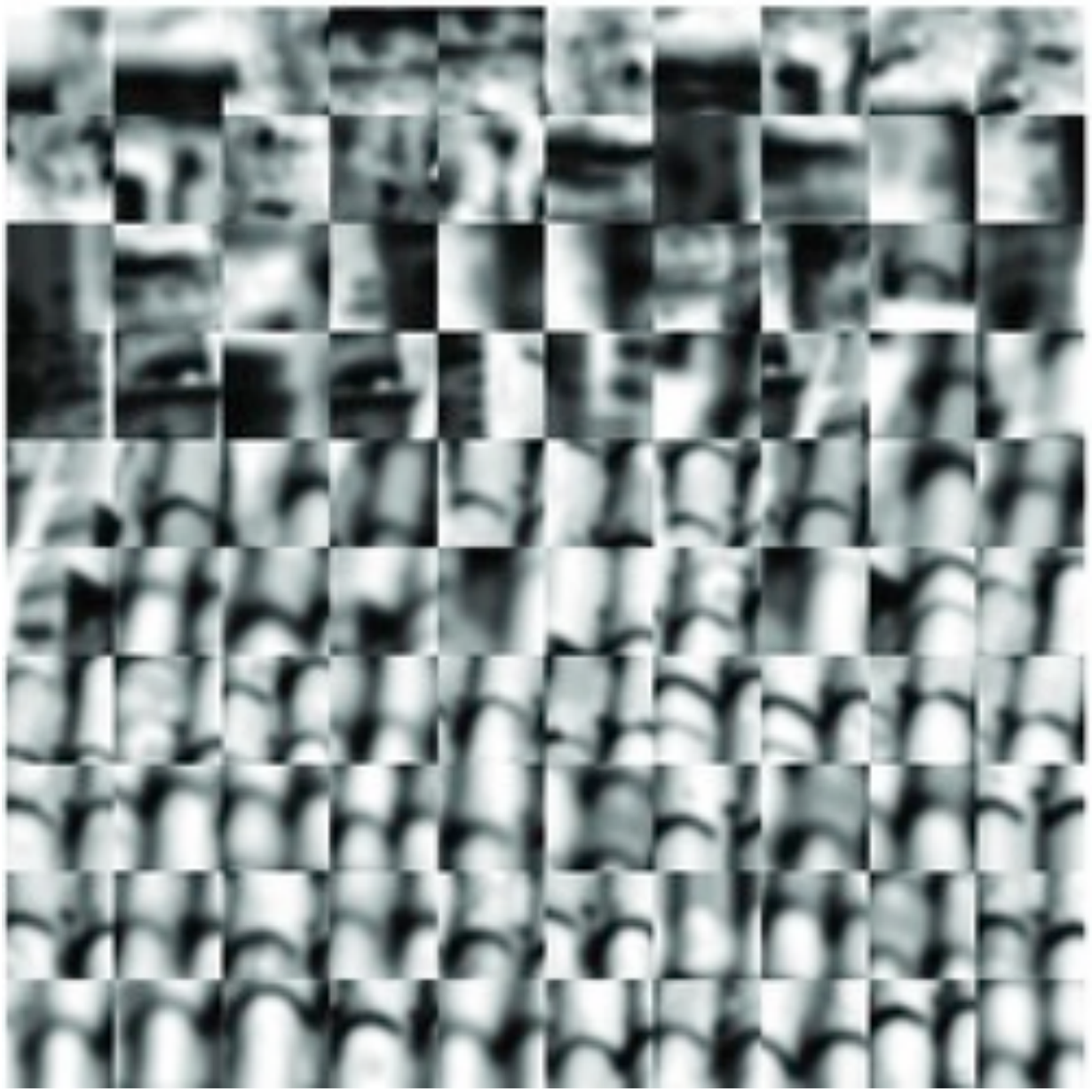

```
[frames, descrs] = vl_covdet(imgs) ;
```



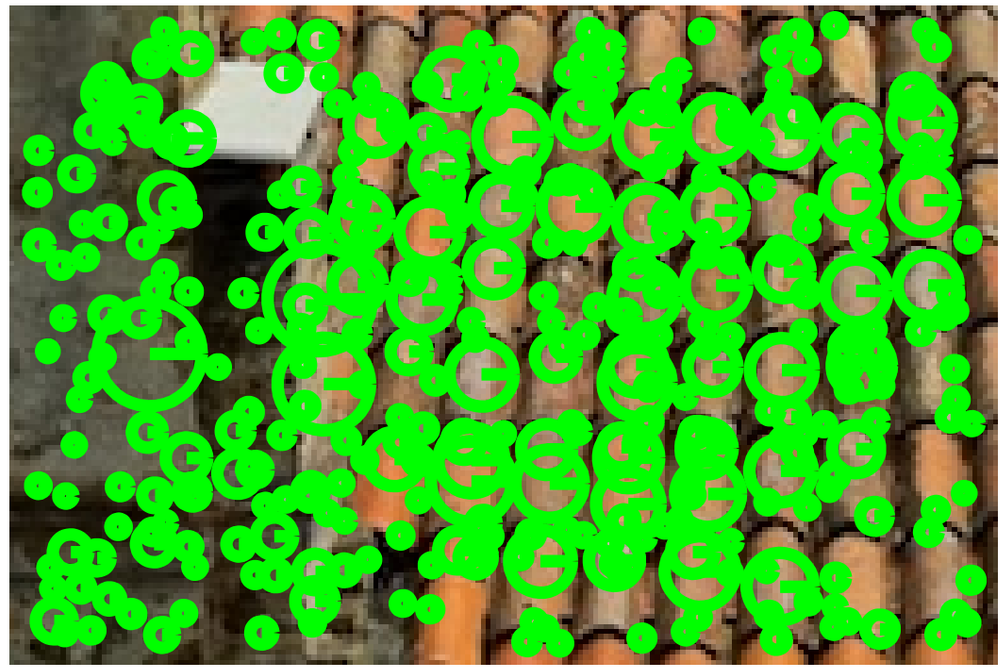
- Each column of `descrs` is a stacked SIFT descriptor
- How is a SIFT descriptor associated to a feature?
- What about other descriptors?

Computing descriptors


```
[frames, descrs] = vl_covdet(imgs, 'Descriptor', 'patch') ;
```

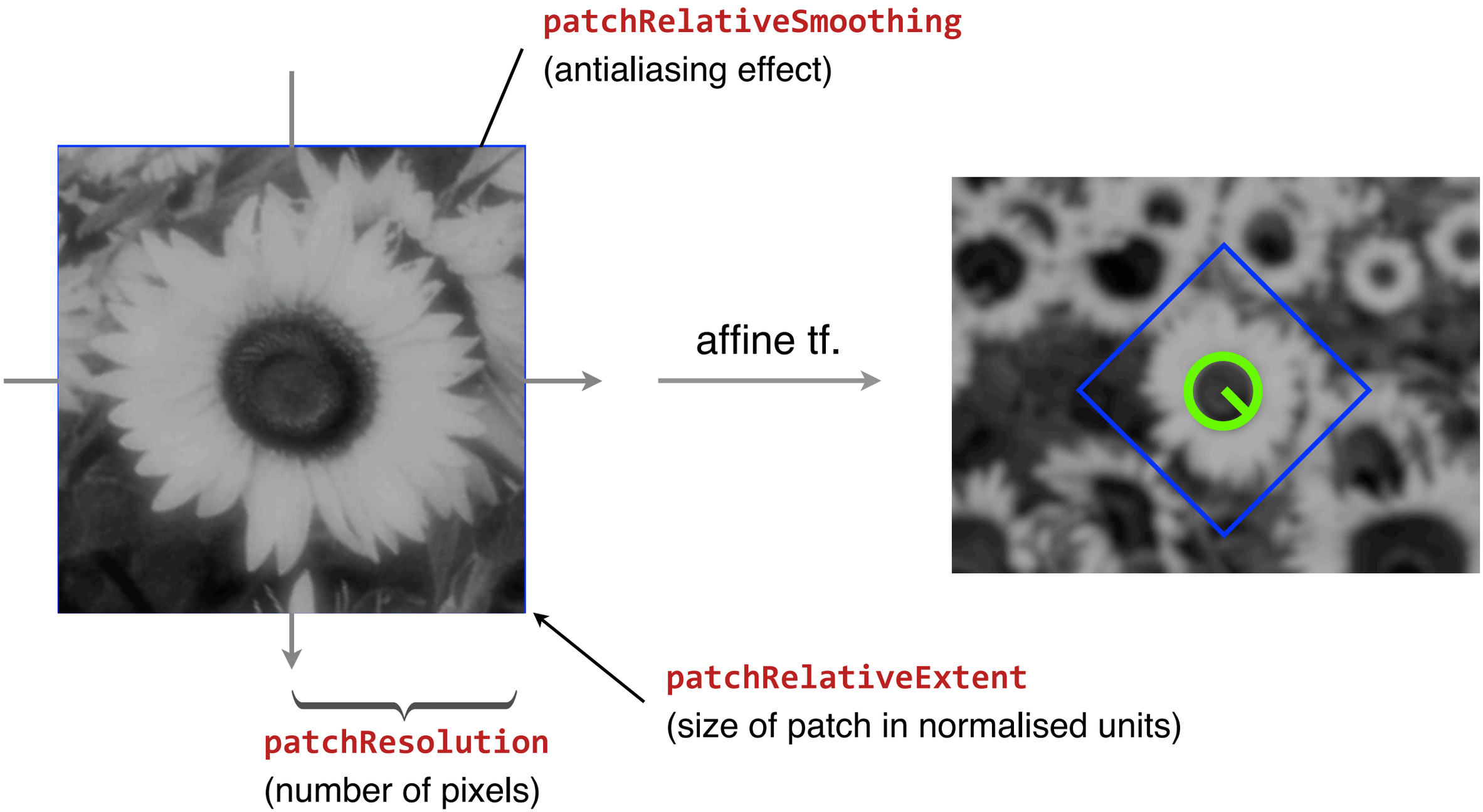


Each column of `descrs` is a stacked **normalised image patch**.

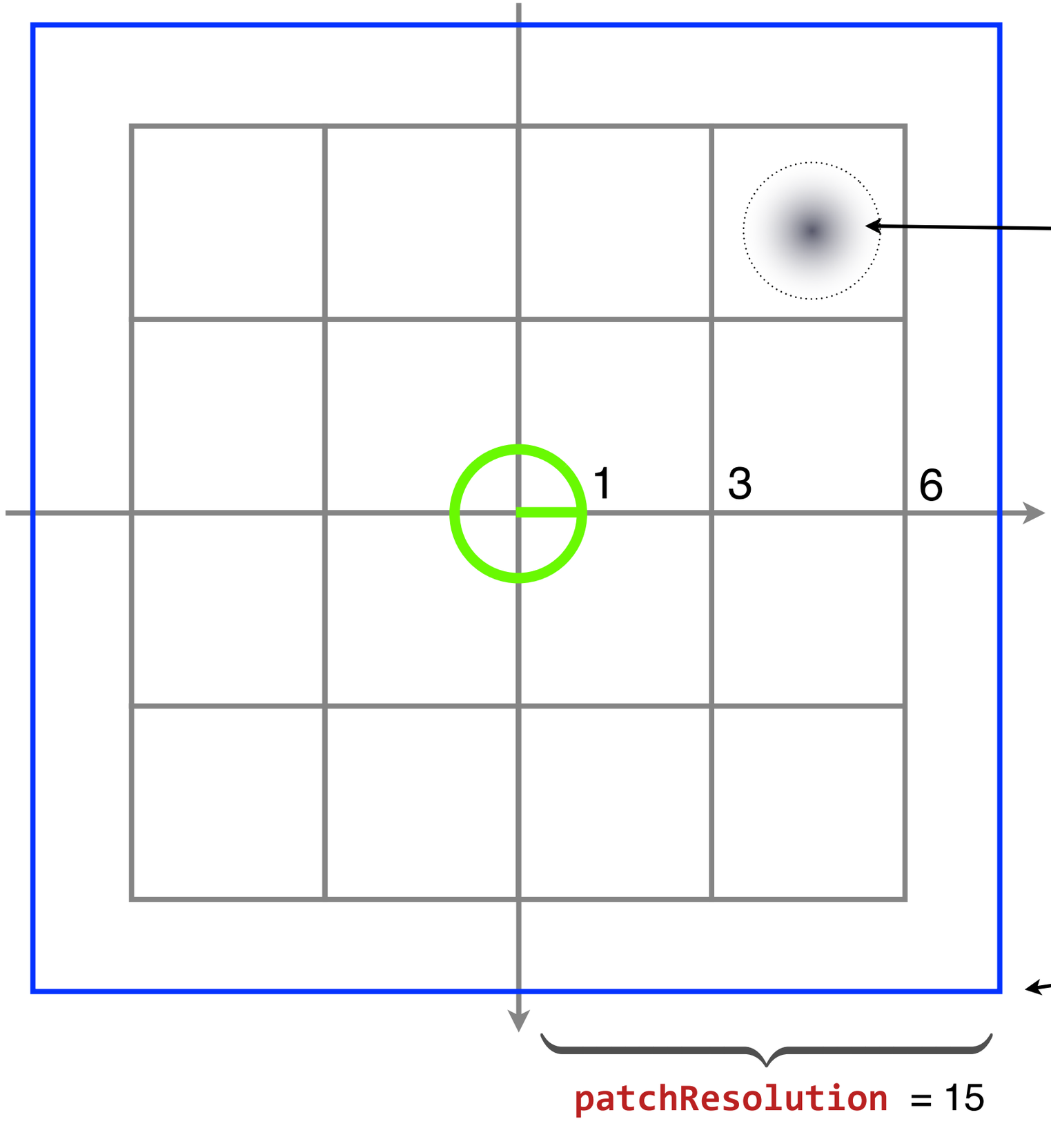


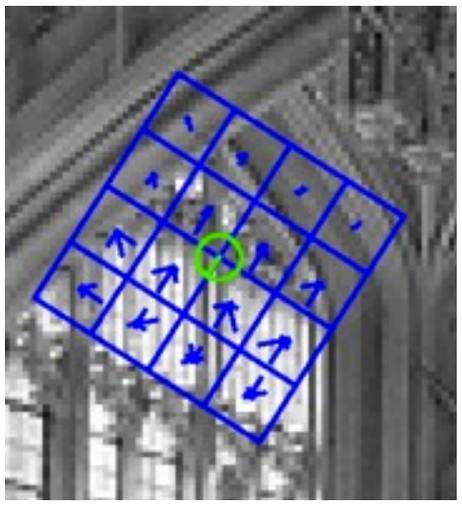
The geometry of an normalised patch

- Normalisation and patch resampling is controlled by three parameters:



Example: normalised patch for SIFT

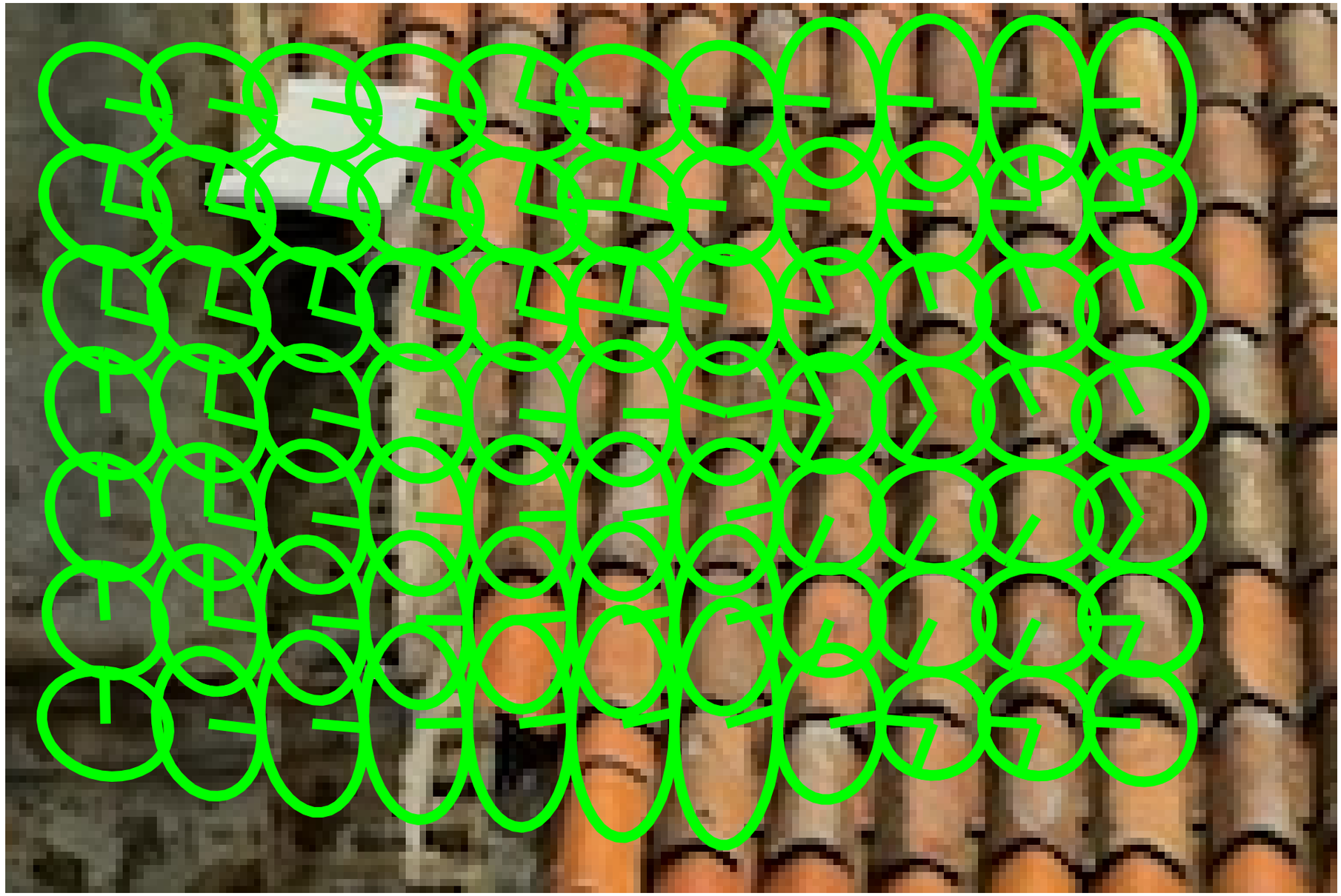




Custom frames

Example: running affine adaptation on a grid

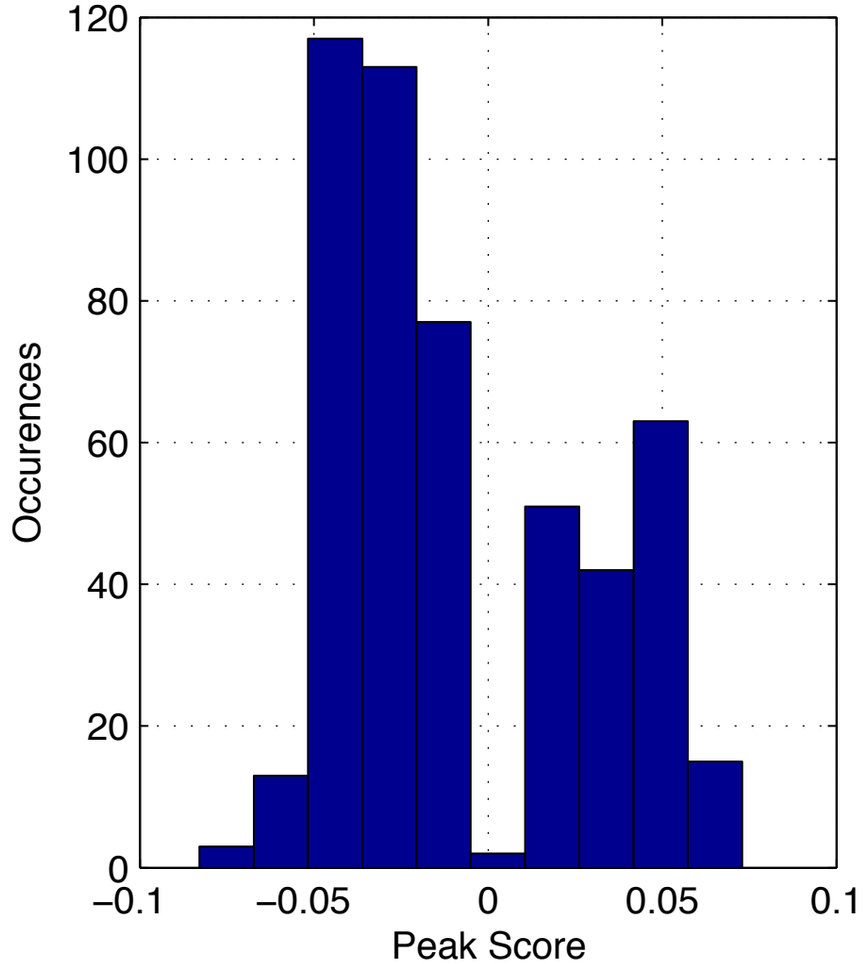
```
frames = vl_covdet(imgs, ...
    'Frames', frames, ...
'EstimateAffineShape', true, ...
 'EstiamteOrientation', true) ;
```

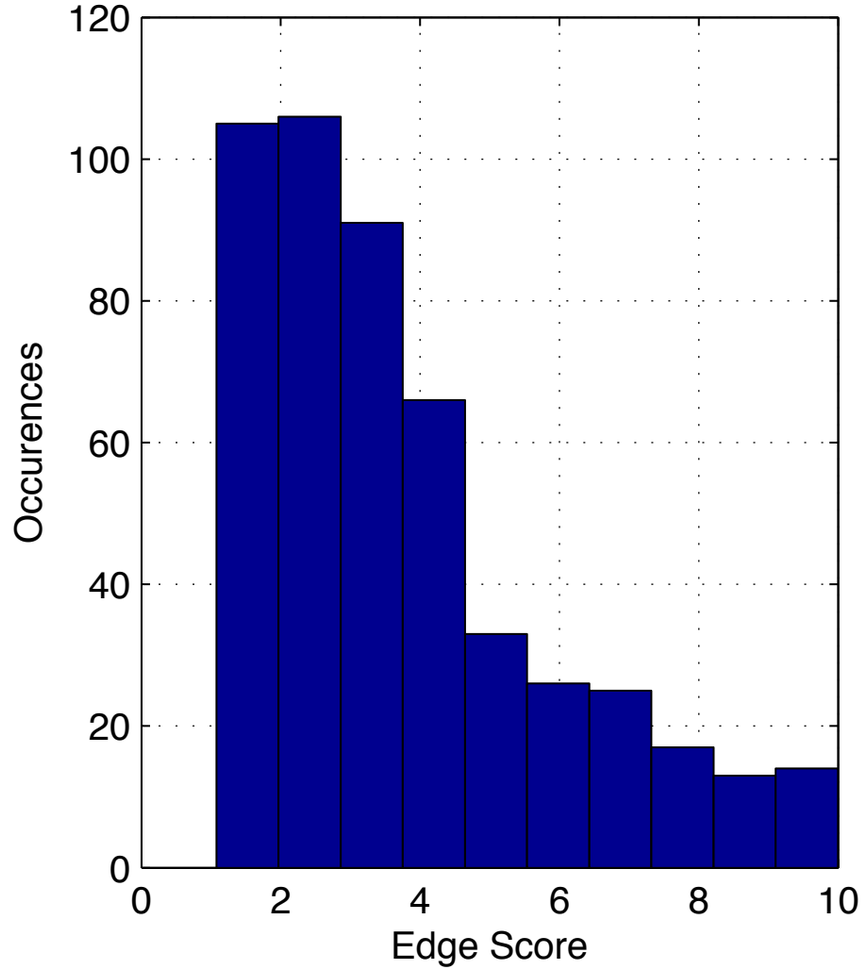


Obtaining additional information on the features


```
[frames, descrs, info] = vl_covdet(imgs);
```

```
info =
                        gss: [1x1 struct]
                        css: [1x1 struct]
                 peakScores: [1x351 single]
                 edgeScores: [1x351 single]
            orientationScores: [1x351 single]
          laplacianScaleScores: [1x351 single]
```





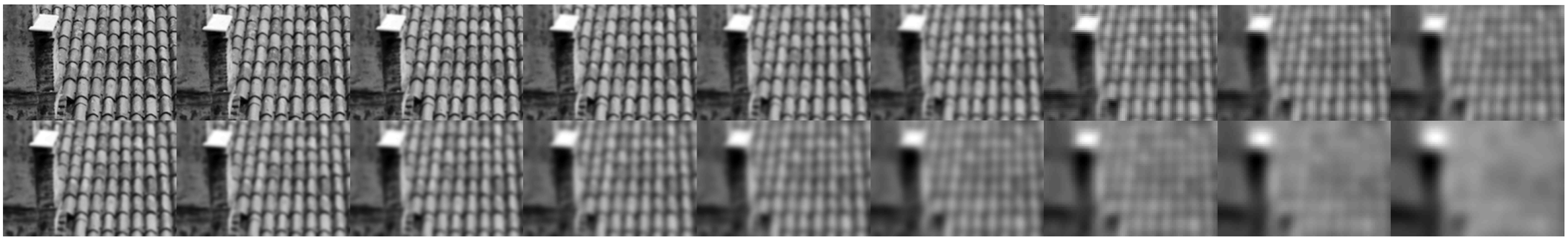
Obtaining the scale space

```
[frames, descrs, info] = vl_covdet(imgs);

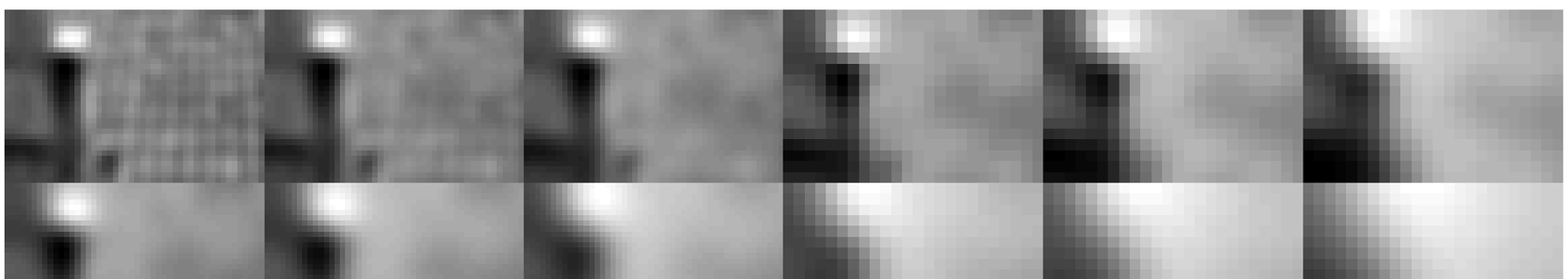
    info =
                          gss: [1x1 struct]
                          css: [1x1 struct]
                   peakScores: [1x351 single]
                   edgeScores: [1x351 single]
              orientationScores: [1x351 single]
            laplacianScaleScores: [1x351 single]
```

Gaussian scale space

octave –1 | octave 0 | octave 1



octave 2 | octave 3



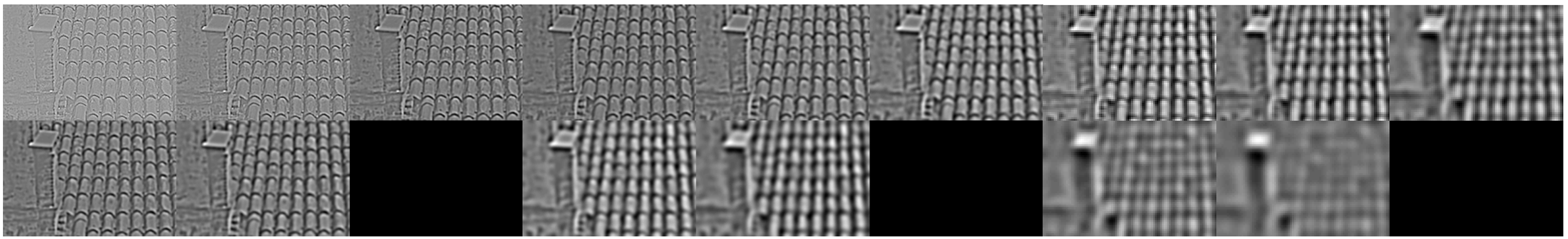
Obtaining the scale space

```
[frames, descrs, info] = vl_covdet(imgs);

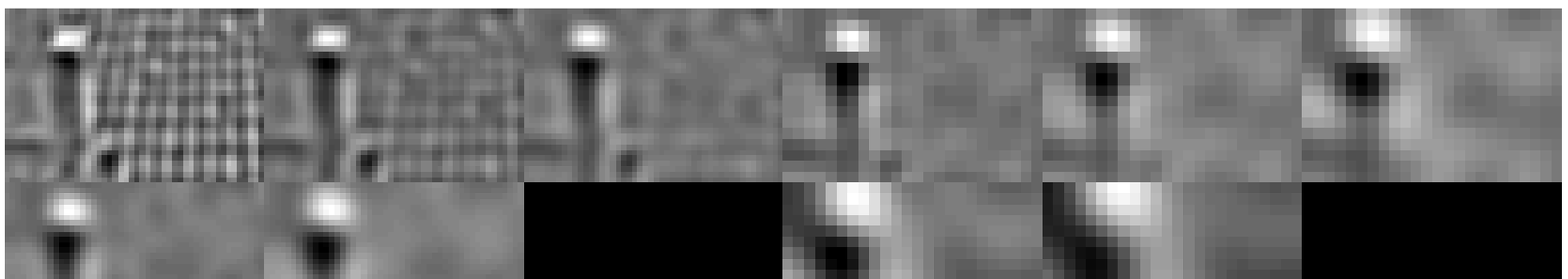
    info =
                          gss: [1x1 struct]
                          css: [1x1 struct]
                   peakScores: [1x351 single]
                   edgeScores: [1x351 single]
              orientationScores: [1x351 single]
            laplacianScaleScores: [1x351 single]
```

cornerness measure

octave –1 octave 0 octave 1

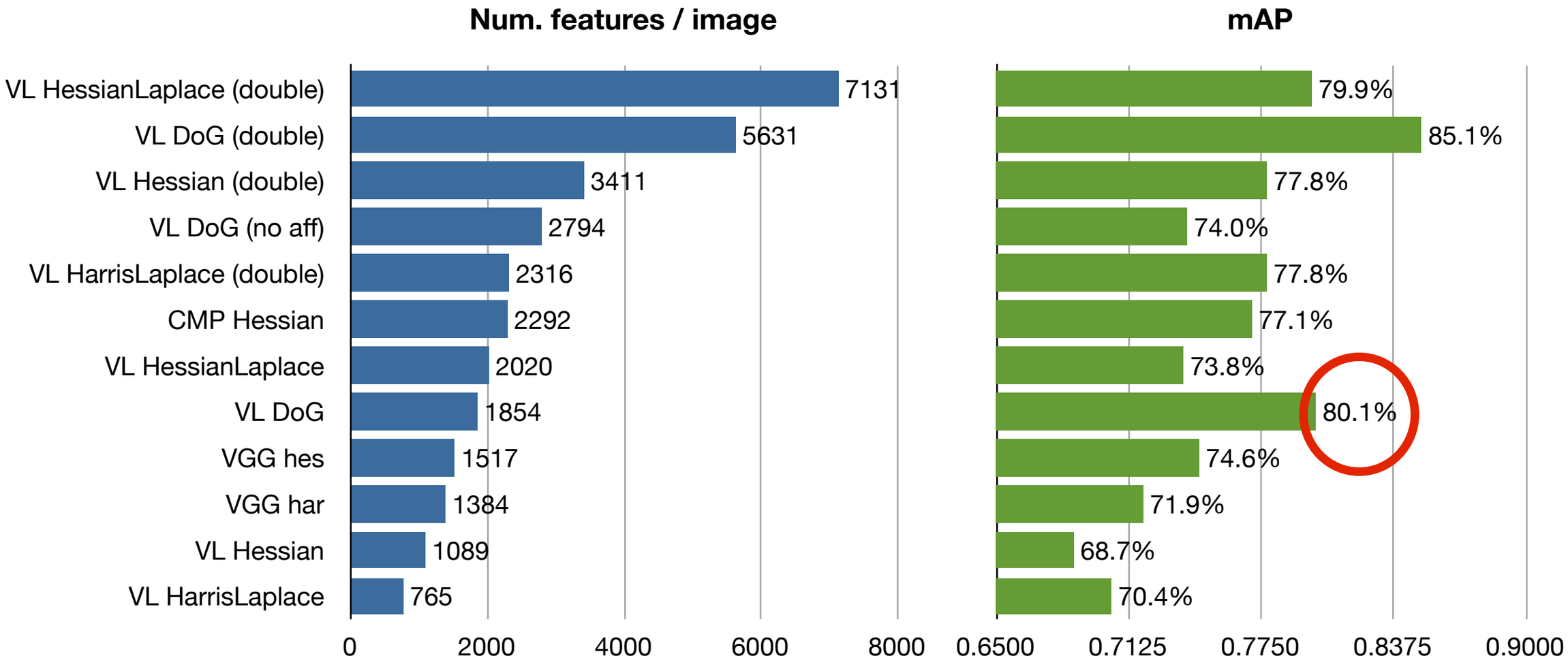


octave 2 octave 3

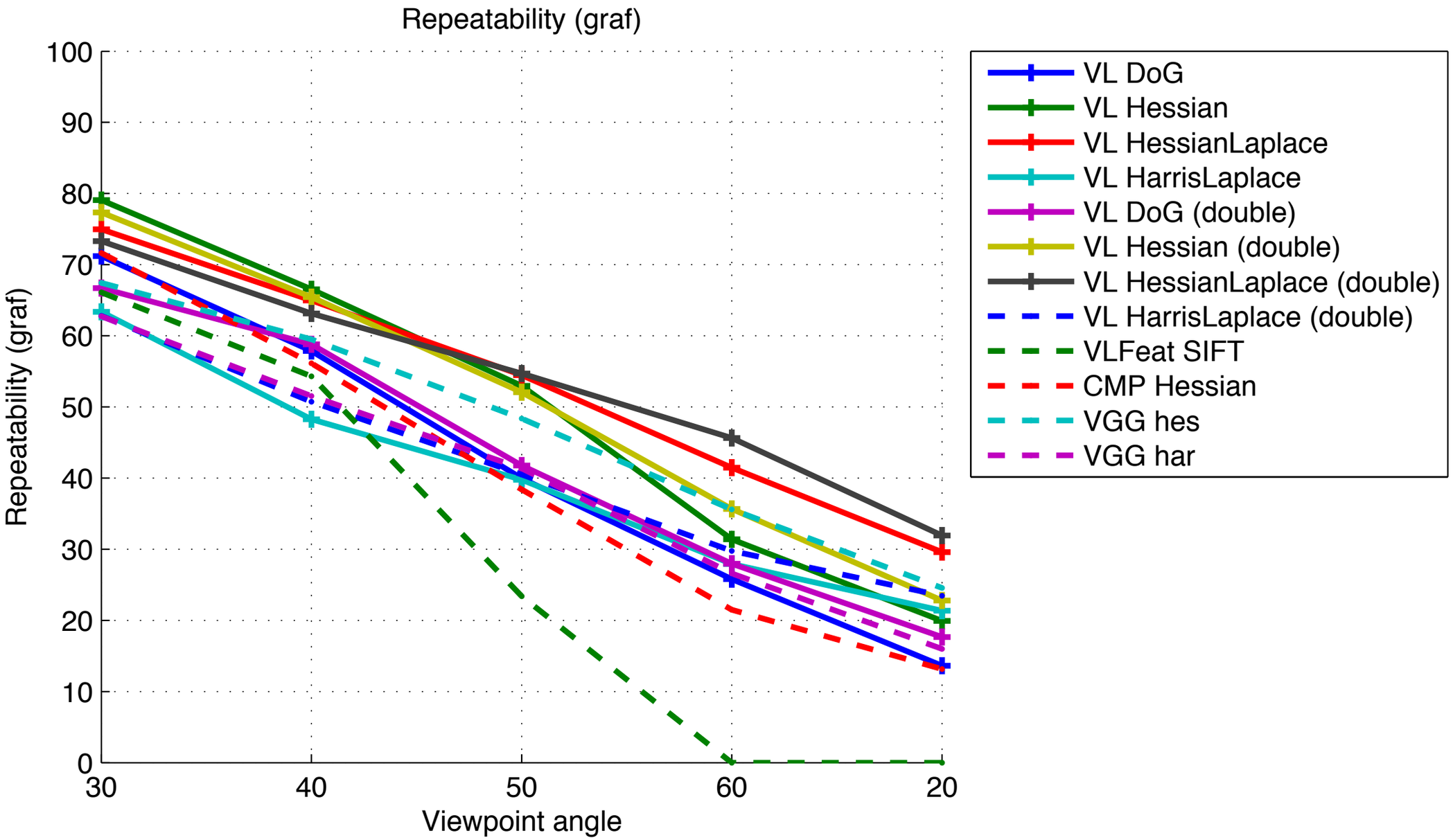


Performance comparison

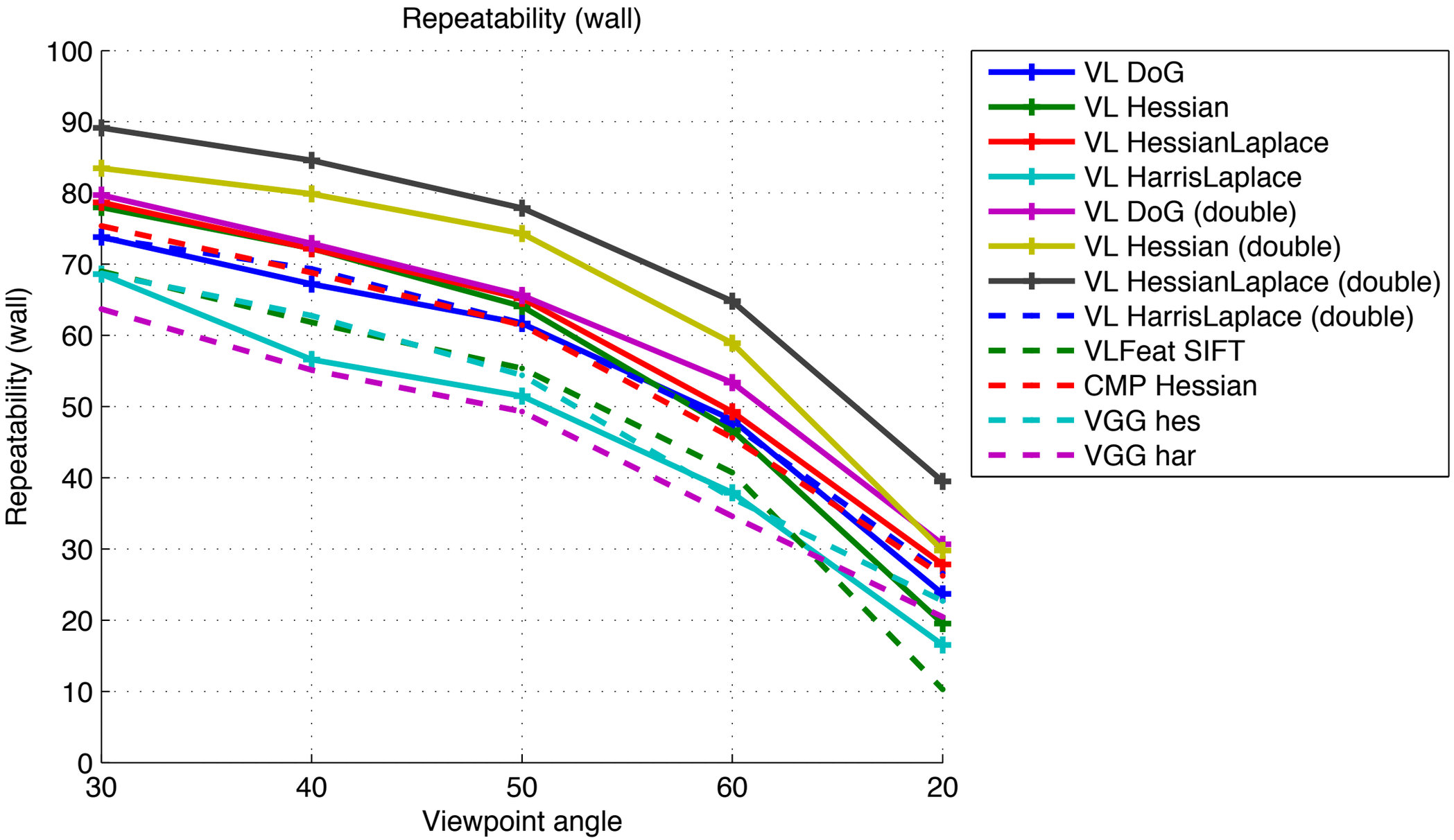
Oxford 5K lite retrieval benchmark



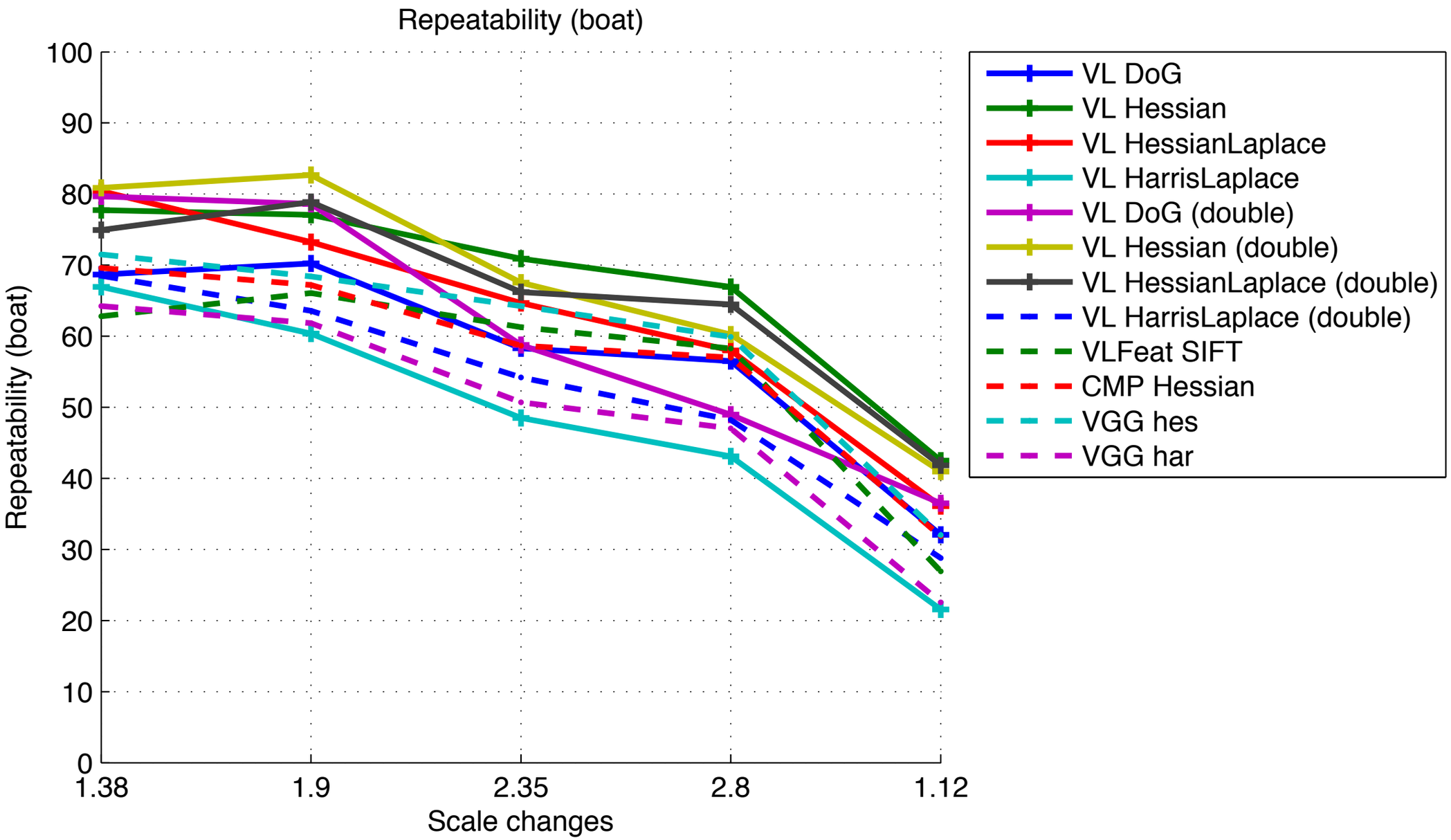
Repeatability: viewpoint



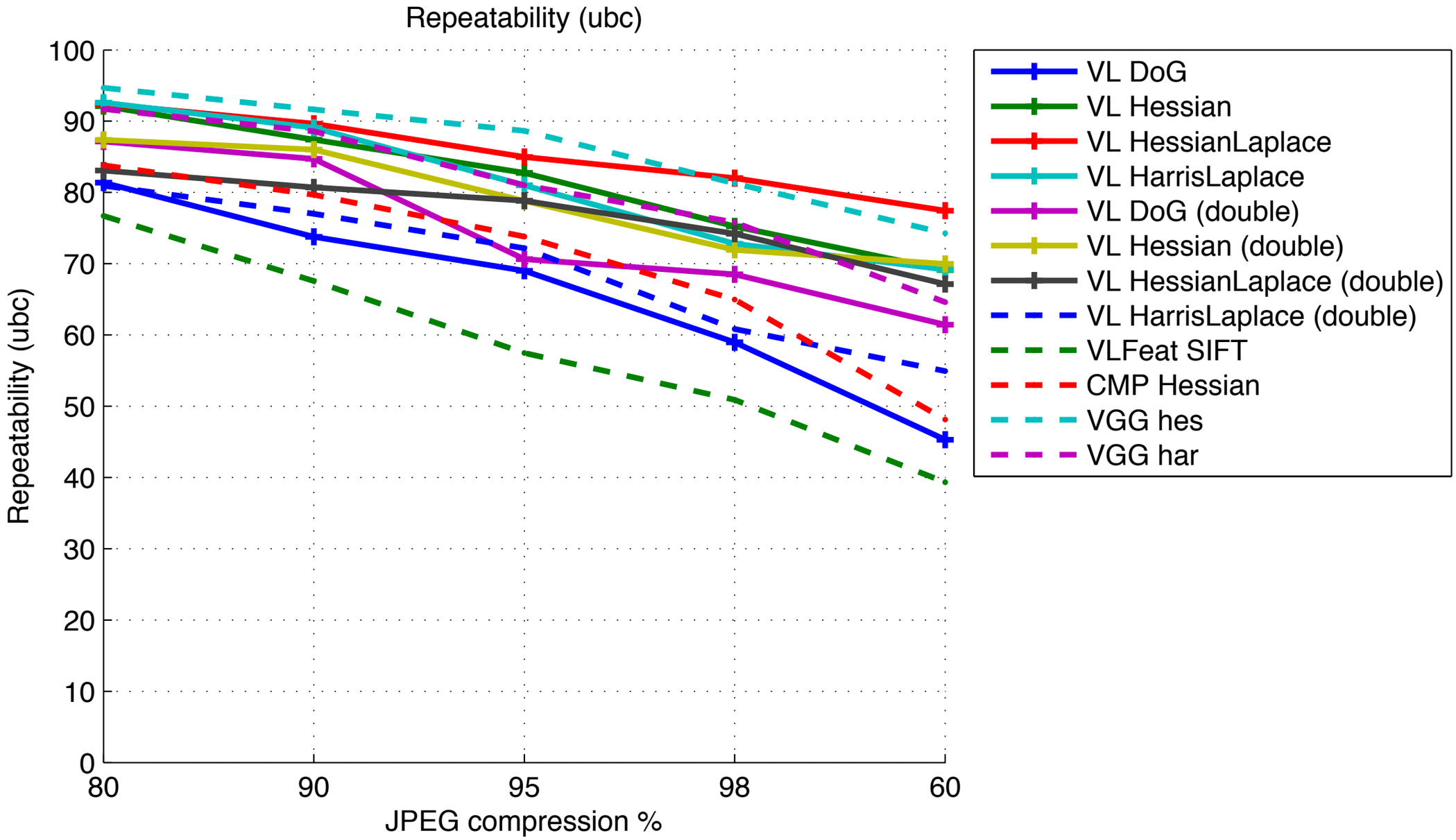
Repeatability: viewpoint



Repeatability: scale

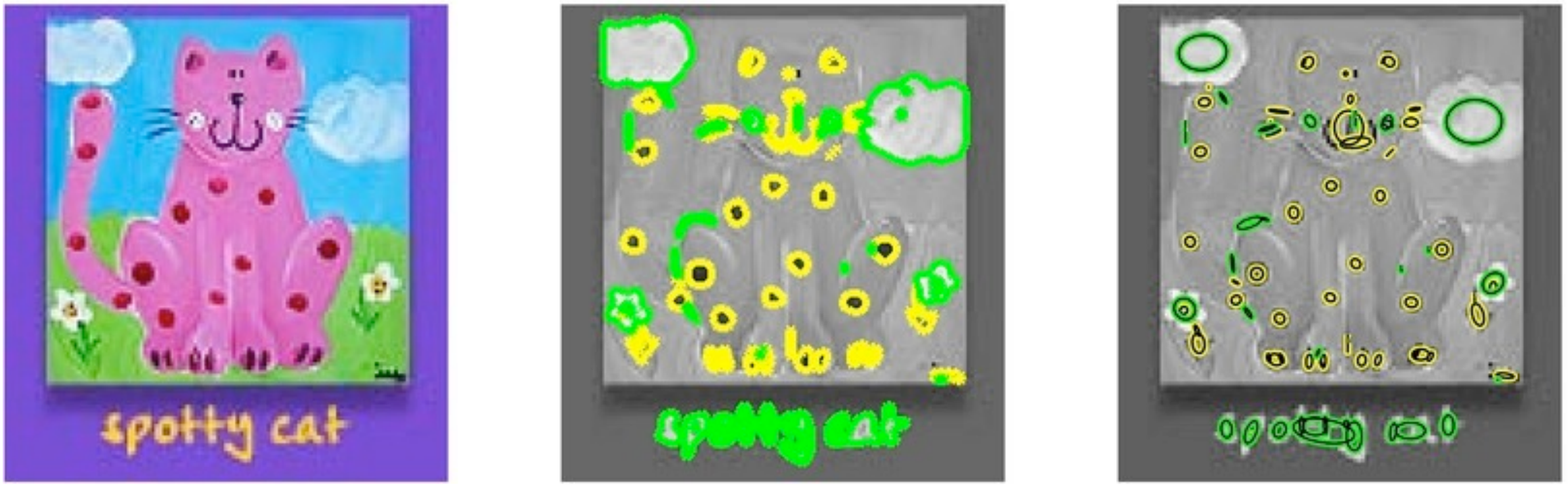


Repeatability: JPEG compression



Maximally Stable Extremal Regions (MSER)

[Matas *et al*. 2002]



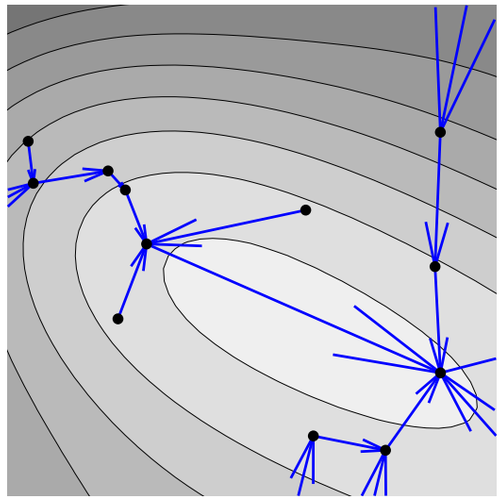
- Extracts MSER regions `regions` and fitted ellipses `frames`:

```
imgs = uint8(rgb2gray(I)) ;
[regions, frames] = vl_mser(imgs) ;
```

- Control region stability:

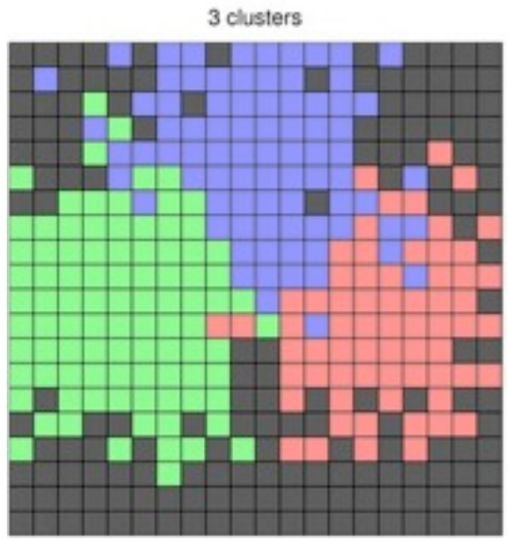
```
[regions, frames] = vl_mser(imgs, ...
                            'MinDiversity', 0.7, ...
                            'MaxVariation', 0.2, ...
                            'Delta', 10) ;
```

Many other features and tools



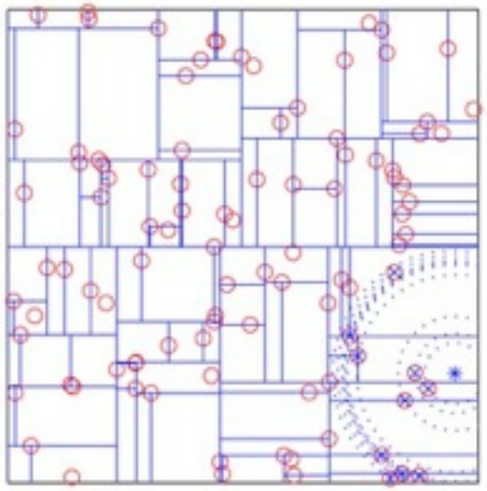
superpixels

MSER

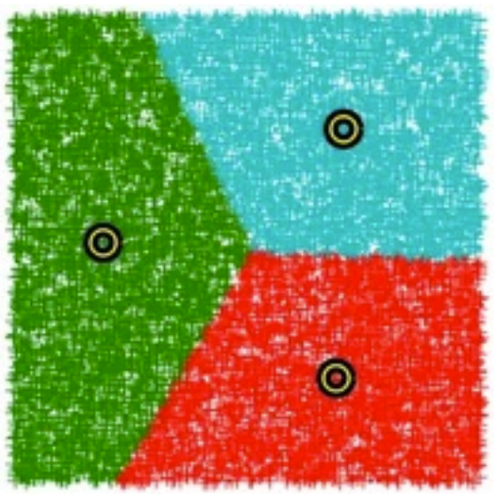


agglomerative information bottleneck

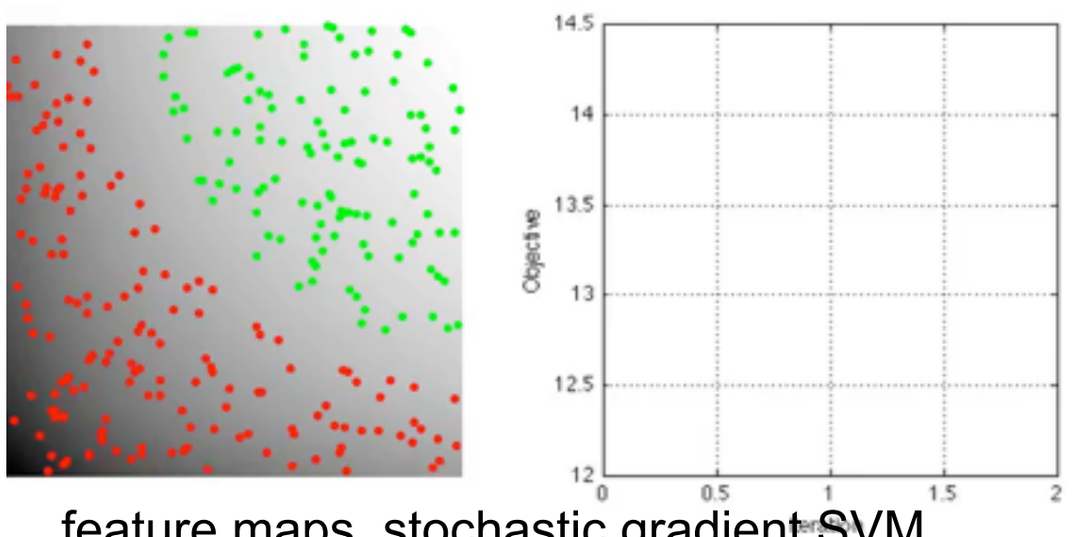
distance transform



randomized kd-trees



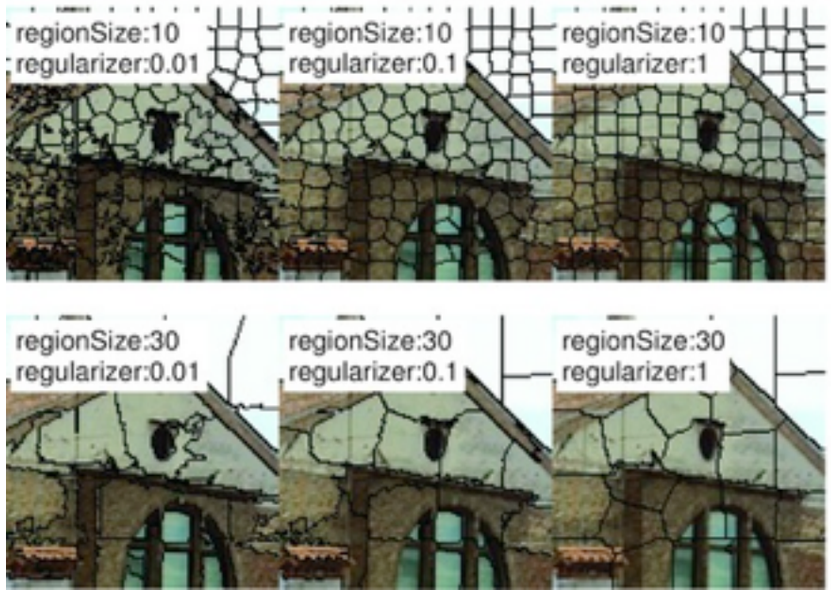
fast k-means



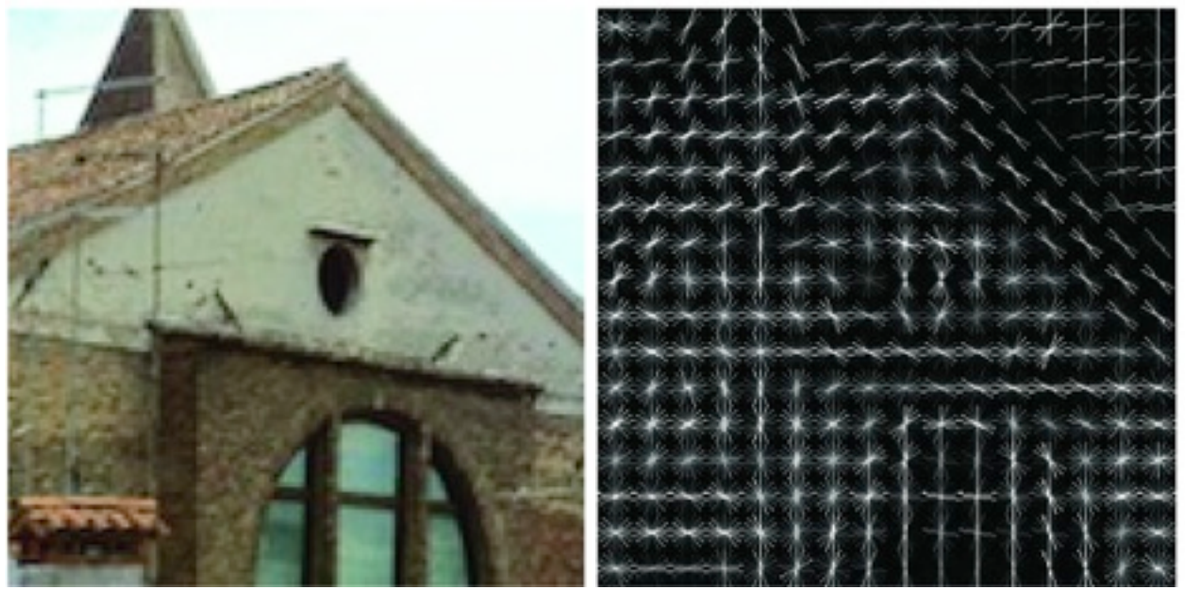
feature maps, stochastic gradient SVM

Fast Dense SIFT

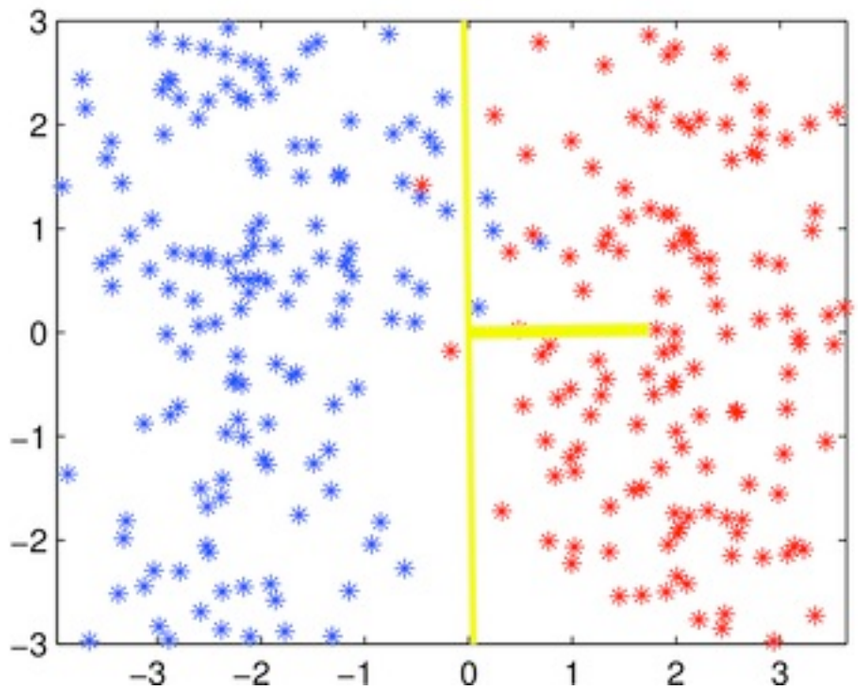
Many other features and tools



SLIC superpixels



Histogram of Oriented Gradients (HOG)



New SGD Learning of SVMs

PASCAL2
HARVEST Programme

OpenCV

- The largest, best known vision library
- **Features**
  - *OpenCV 2.1*:
    Harris, Tomasi&Shi, MSER, DoG, SURF, STAR and FAST,
    DoG + SIFT, SURF, HARRIS, GoodFeaturesToTrack
  - *OpenCV 2.4*:
    ORB, BRIEF, and FREAK

Example: computing ORB features.

```
#include <opencv2/core/core.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <vector>

Mat image = importImage8UC1(in[IN_I]);
Mat descriptors;
std::vector<KeyPoint> frames;

ORB orb(numFeatures, scaleFactor, numLevels, edgeThreshold, 0,
        wtak, scoreTypeToORBPar(scoreType), patchSize);

orb(image, Mat(), frames);
```

This is *not* all folks

- Abundance of open source / free software for CV
  - Always check out authors home pages
- **Examples**

Rob Hess SIFT

http://robwhess.github.com/opensift/

SURF

http://www.vision.ee.ethz.ch/~surf/

Open SURF

http://www.chrisevansdev.com/computer-vision-opensurf.html

FAST corners

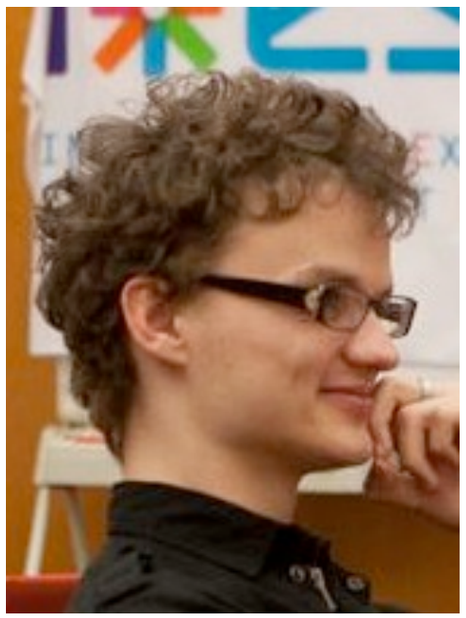
http://www.edwardrosten.com/work/fast.html

LIOP descriptors

http://vision.ia.ac.cn/Students/bfan/index.htm

■ ■ ■

Credits



Karel Lenc

Daniele Perrone

Michal Perdoch

Krystian Mikolajczyk

Tinne Tuytelaars

Jiri Matas

Cordelia Schmid

Andrew Zisserman

PASCAL2
HARVEST Programme

Feature evaluation

Old and new benchmarks, and new software

Andrea Vedaldi, University of Oxford

Benchmarking: why and how

- Dozens of feature detectors and descriptors have been proposed
- **Benchmarks**
  - compare methods empirically
  - select the best method for a task
- **Public benchmarks**
  - reproducible research
  - simplify your life!
- **Ingredients** of a benchmark:

Indirect evaluation

Repeatability and matching score

Data: affine covariant testbed

Direct evaluation

Image retrieval

Data: oxford 5k

Software

VLBenchmarks

Indirect evaluation

Repeatability and matching score

Data: affine covariant testbed

Direct evaluation

Image retrieval

Data: oxford 5k

Software

VLBenchmarks

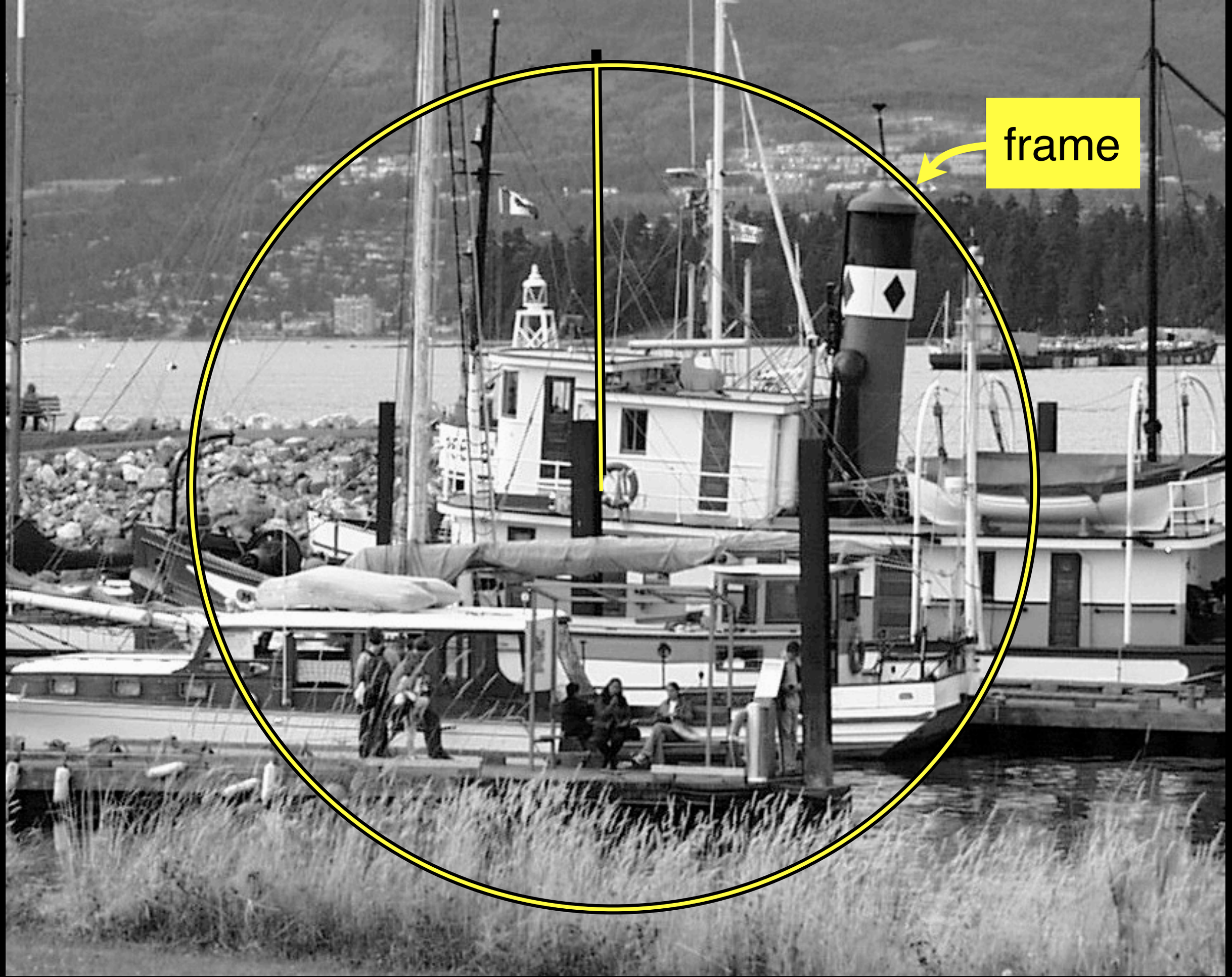
Indirect feature evaluation

- **Intuition**
  Test how well features **persist** and **can be matched** across image **transformations**.

- **Data**
  - *must be representative* of the transformations (viewpoint, illumination, noise, etc.)

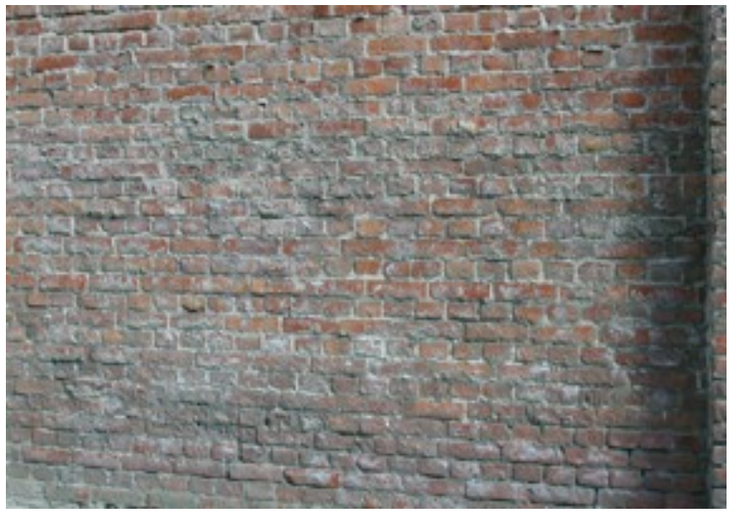
- **Performance measures**
  - **repeatability**
    persistence of features
  - **matching score**
    matchability of features

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. *A comparison of affine region detectors*. IJCV, 1(65):43–72, 2005.


frame

Affine Testbed

Viewpoint, scale, rotation







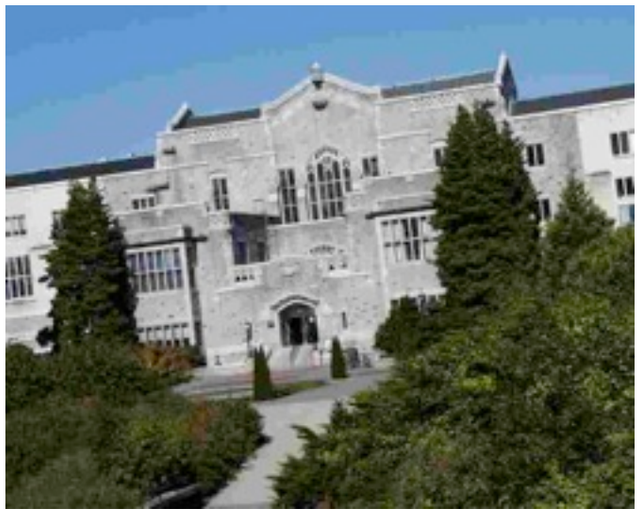


Affine Testbed

Lighting, compression, blur









Detector repeatability

Intuition

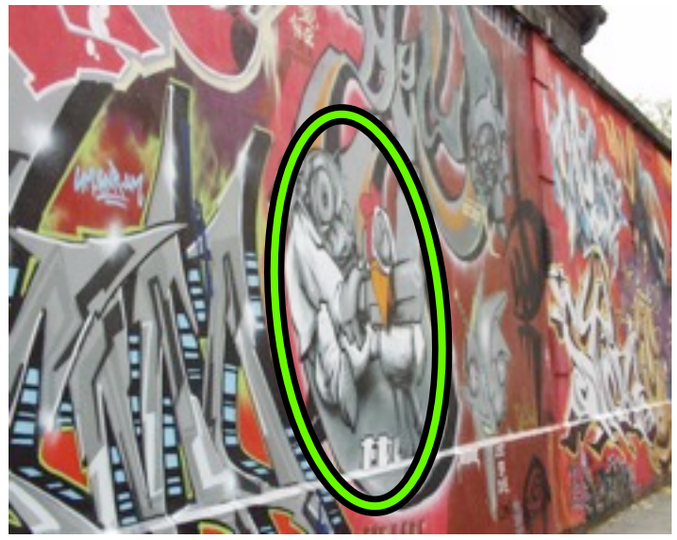
- **two pairs of features correspond**
- **another pair of features does not**
- repeatability = 2/3

Region overlap

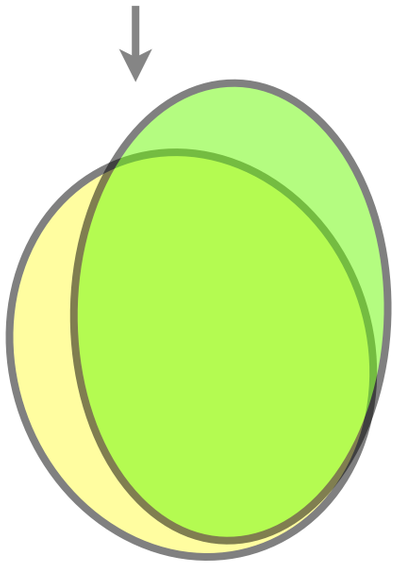
Formal definition

H

homography

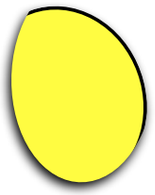


R_b



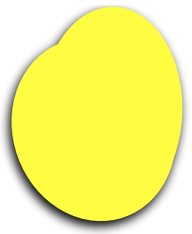
R_a HR_b

$$\text{overlap}(a,b) = \frac{|R_a \cap HR_b|}{|R_a \cup HR_b|}$$



area intersection

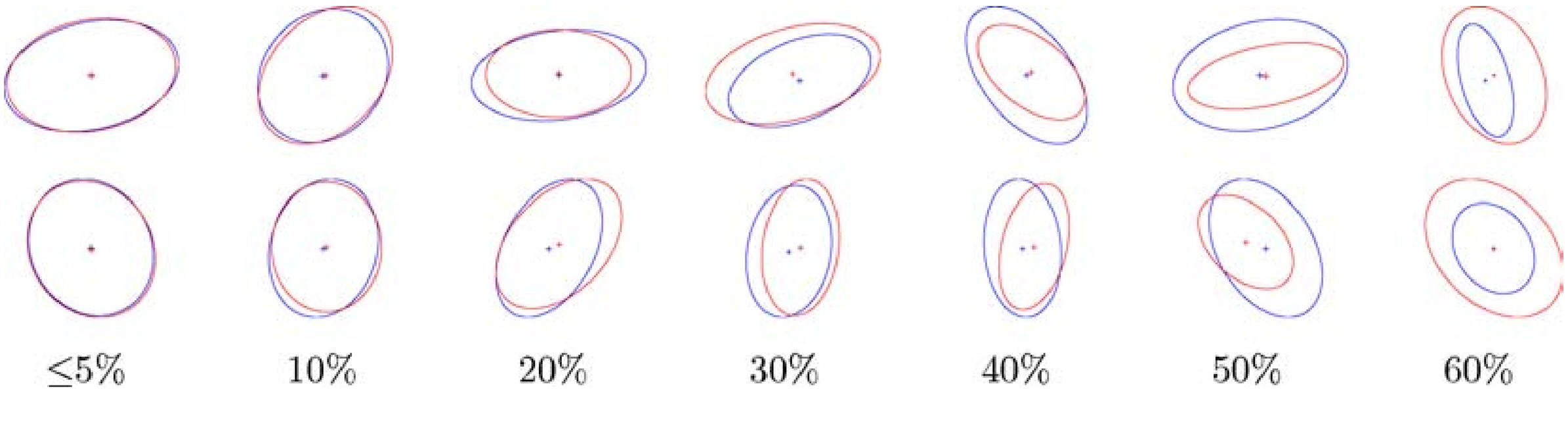
=



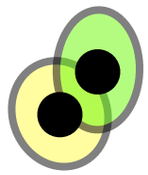
area union

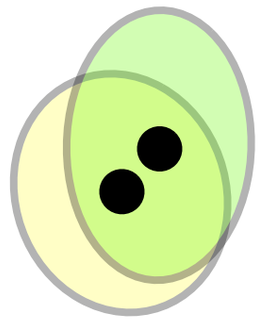
Region overlap

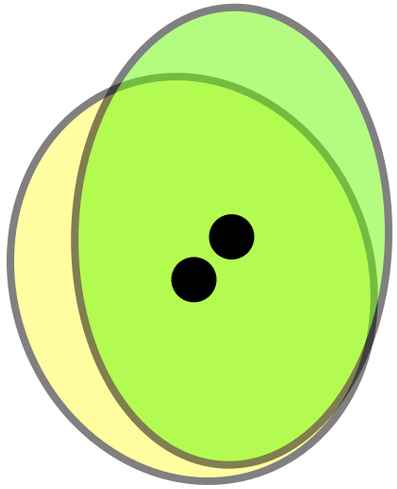
Intuition

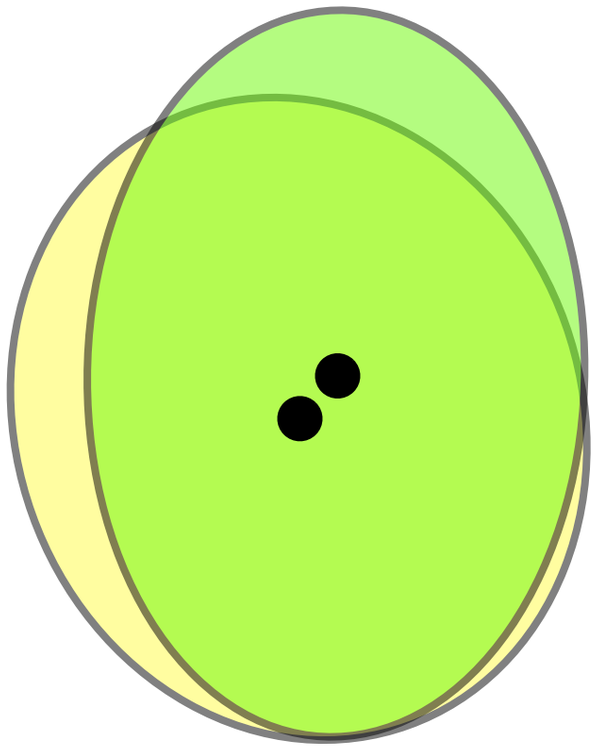


- Examples of ellipses overlapping by different amounts
- Usually, features are tested at 40% overlap *error* (= 60% overlap)



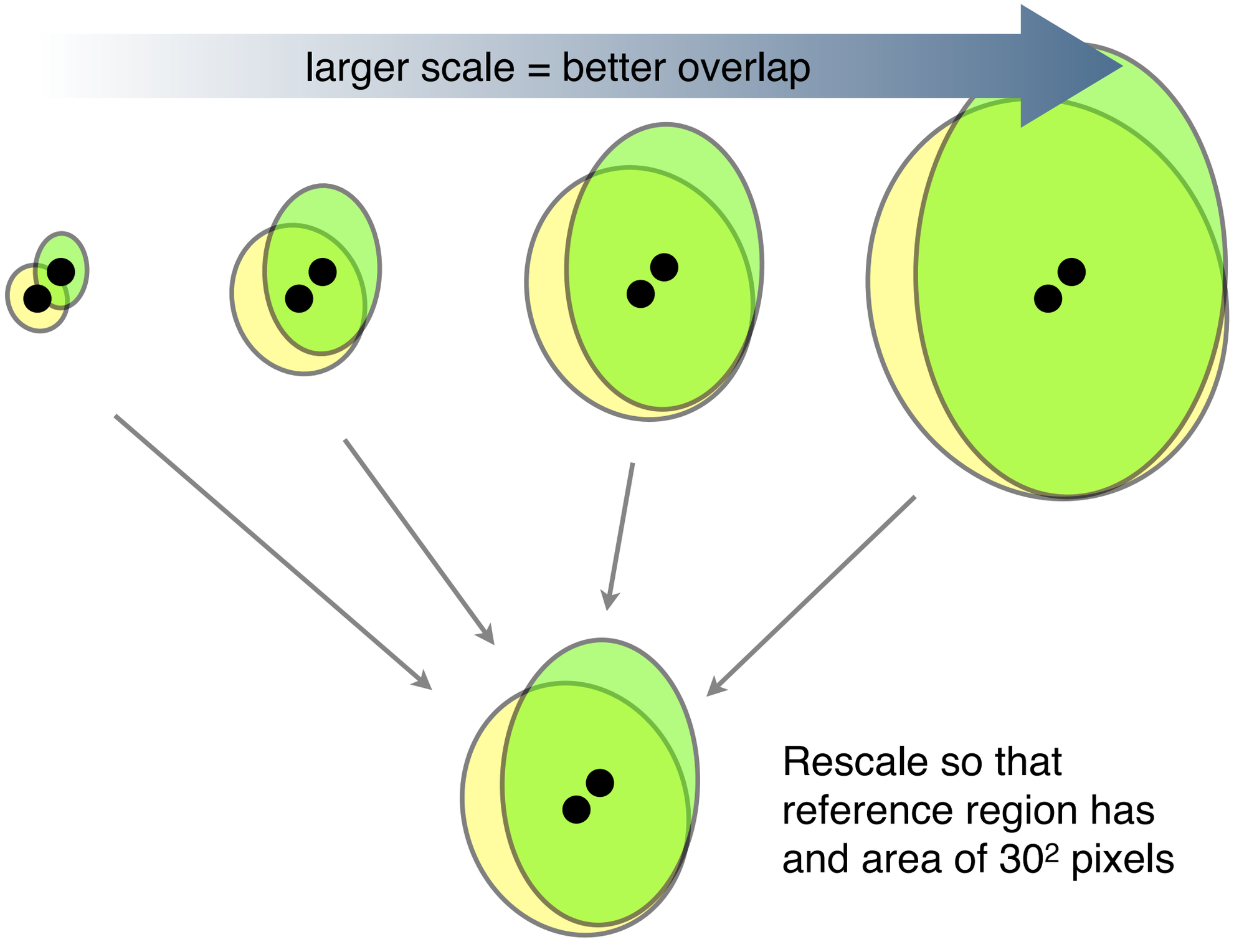






Normalised region overlap

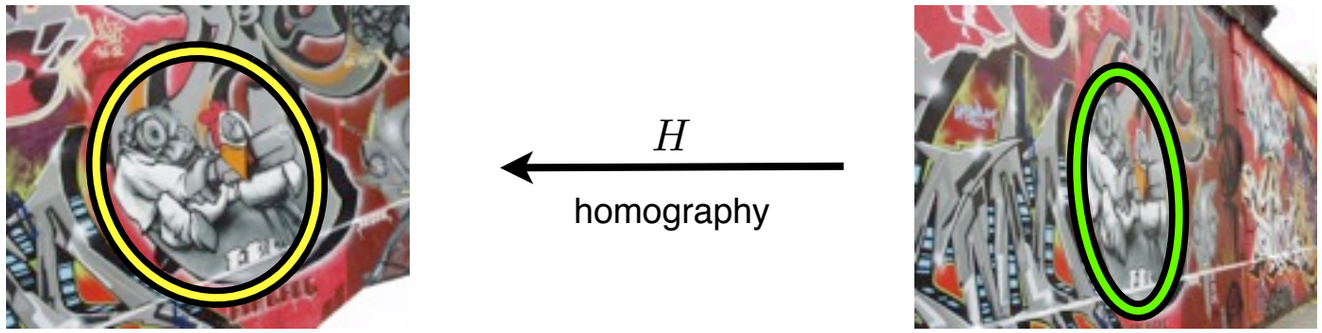
Intuition



Normalised region overlap

Formal definition

1. Detect



2. Warp R_a HR_b

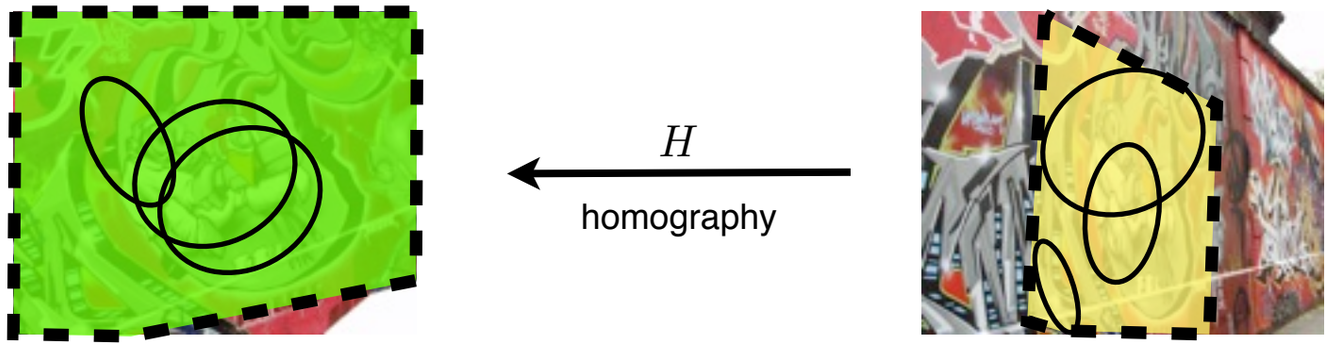
3. Normalise sR_a sHR_b $s = |R_a|/30^2$

4. Intersection over union $$\text{overlap}(a, b) = \frac{|sR_a \cap sHR_b|}{|sR_a \cup sHR_b|}$$

Detector repeatability

Formal definition

1. Find features in common area



$\{R_a : a \in A\}$ $\{R_b : b \in B\}$

2. Threshold the overlap score

$$s_{ab} = \begin{cases} \text{overlap}(a,b), & \text{overlap}(a,b) \geq 1 - \epsilon_o \\ -\inf, & \text{otherwise.} \end{cases}$$

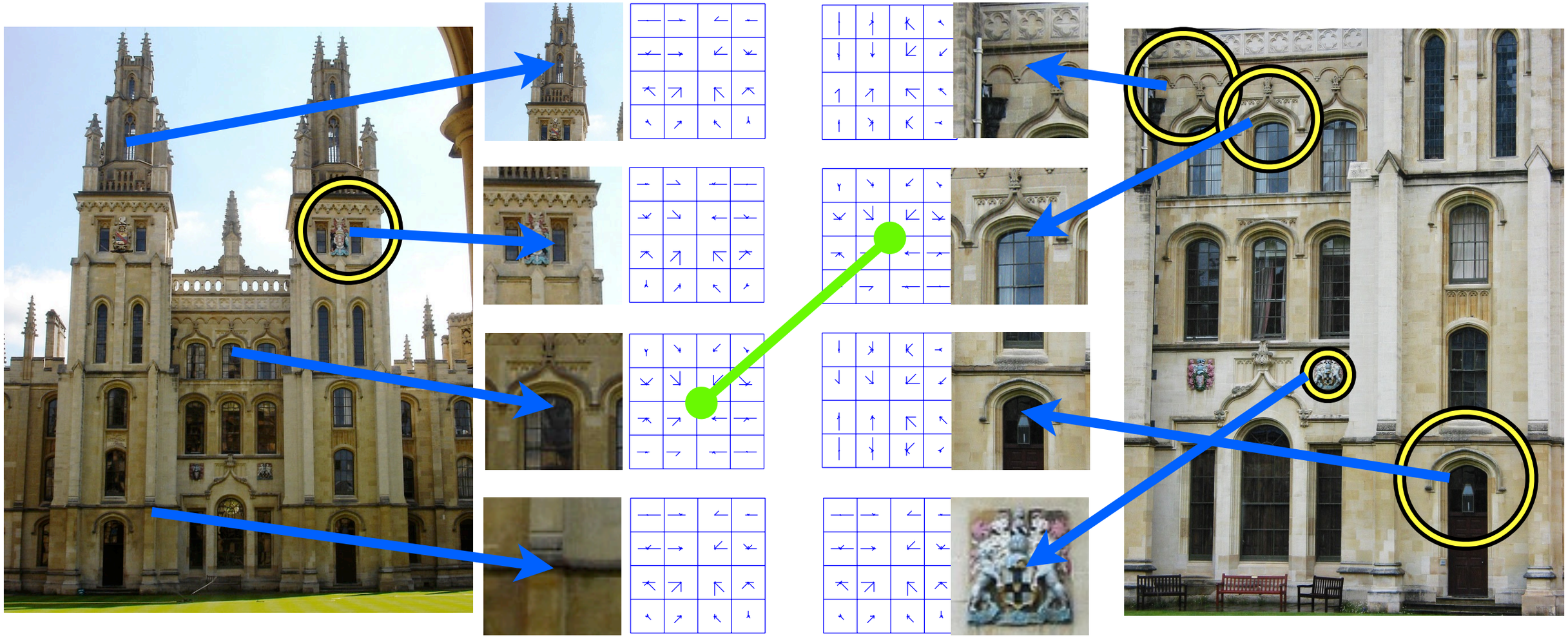
3. Find geometric matches

$$\mathcal{M}^* = \max_{\mathcal{M} \text{ bipartite}} \sum_{(a,b)\in\mathcal{M}} s_{ab}$$ ← greedy approximation

$$\text{repeatability}(A,B) = \frac{|\mathcal{M}^*|}{\min\{|A|,|B|\}}$$

Descriptor matching score

Intuition

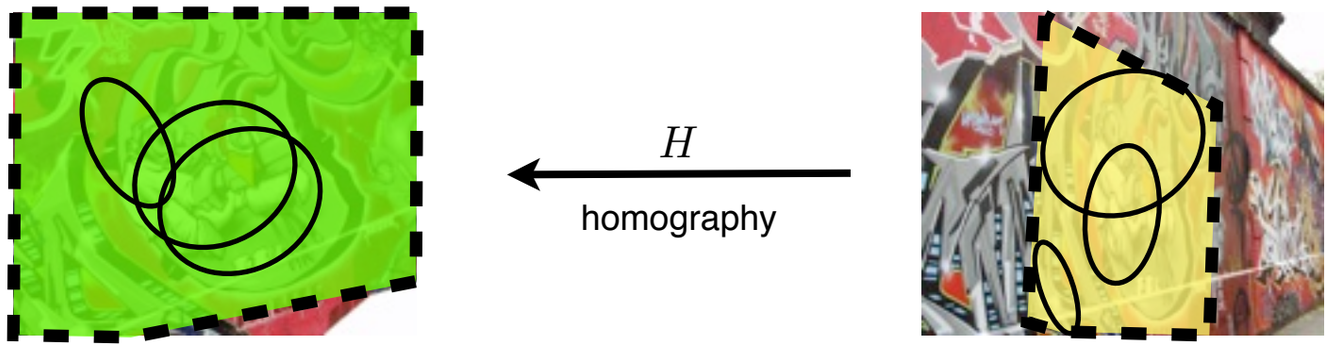


- In addition to being stable, features must be **visually distinctive**
- **Descriptor matching score**
  - similar to repeatability
  - but matches are constructed by comparing descriptors

Descriptor matching score

Formal definition

1. Find features in common area



$\{R_a : a \in A\}$ $\{R_b : b \in B\}$

2. Descriptor distances

$\{\mathbf{d}_a : a \in A\}$ $\{\mathbf{d}_b : b \in B\}$ $d_{ab} = \|\mathbf{d}_a - \mathbf{d}_b\|_2$

3. Descriptor matches

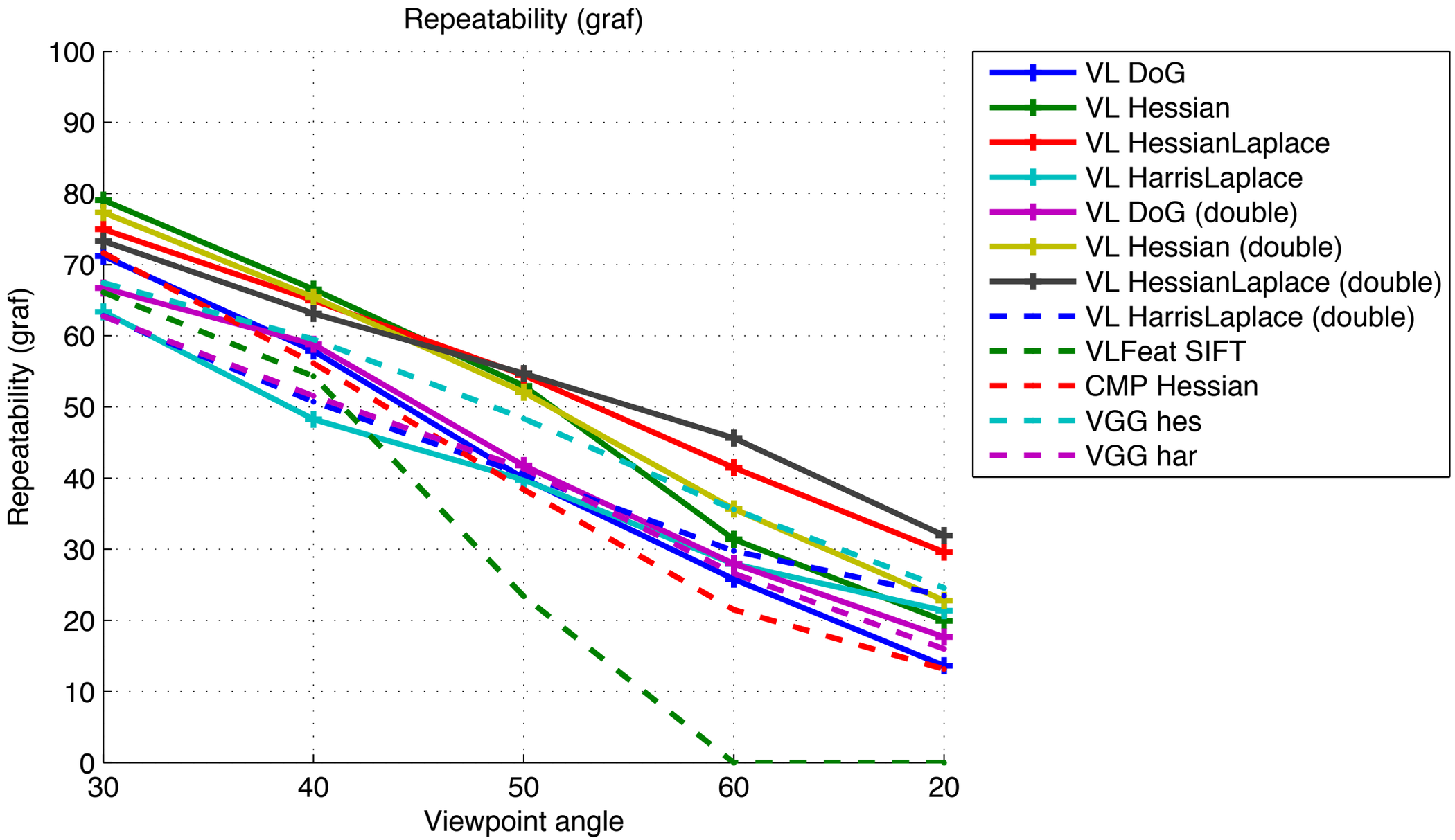
$$\mathcal{M}_d^* = \min_{\mathcal{M} \text{ bipartite}} \sum_{(a,b)\in\mathcal{M}} d_{ab}$$

4. Geometric matches (as before)

$$\mathcal{M}^* = \max_{\mathcal{M} \text{ bipartite}} \sum_{(a,b)\in\mathcal{M}} s_{ab}$$

$$\text{match-score}(A, B) = \frac{|\mathcal{M}^* \cap \mathcal{M}_d^*|}{\min\{|A|, |B|\}}$$

Example of a repeatability graph



Indirect evaluation

Repeatability and matching score

Data: affine covariant testbed

Direct evaluation

Image retrieval

Data: oxford 5k

Software

VLBenchmarks

Indirect evaluation

- **Indirect evaluation**
  - a “synthetic” performance measure in a “synthetic” setting
- **The good**
  - independent of a specific application / implementation
  - allow to evaluate single components, e.g.
    - repeatability of detector
    - matching score of descriptor
- **The bad**
  - difficult to design well
  - unclear correlation to the performance in applications

Direct evaluation

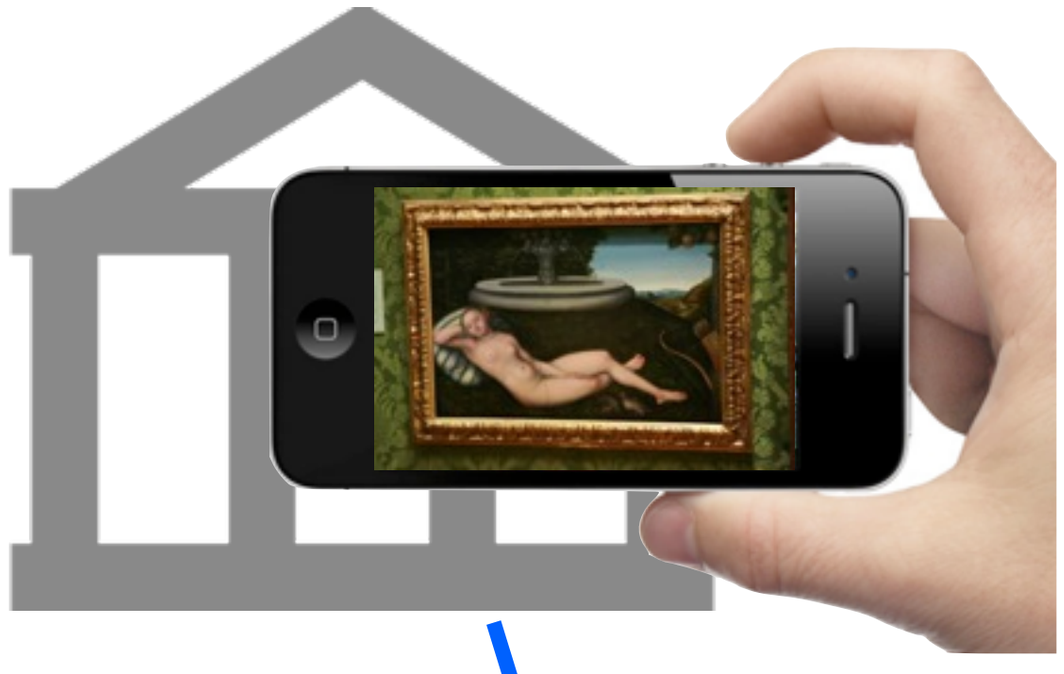
- **Direct evaluation**
  - performance of a real system using a feature
- **The good**
  - tied to the “real” performance of the feature
- **The bad**
  - tied to one *application*
  - worse, tied to one *implementation*
  - difficult to evaluate single aspects of a feature

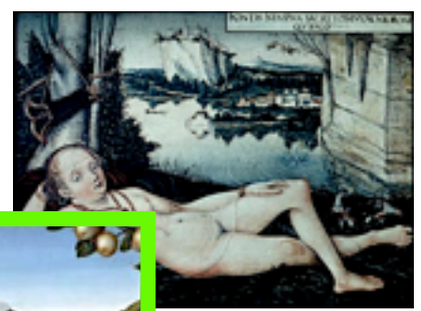
object instance retrieval
object category recognition
object detection
text recognition
semantic segmentation
...

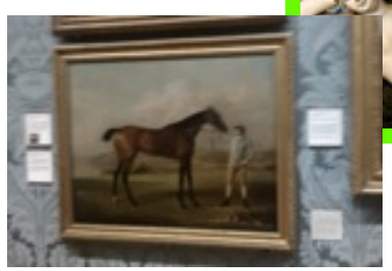
- In the follow up we will focus on **object instance retrieval**

Image retrieval

Used to evaluate features







...

Image retrieval pipeline

Represent images as bags of features

input image

detector

$\{\mathbf{f}_1, \ldots, \mathbf{f}_n\}$

Harris (Laplace)

Hessian (Laplace)

DoG

MSER

Harris Affine

Hessian Affine

....

descriptor

$\{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$

SIFT

LIOP

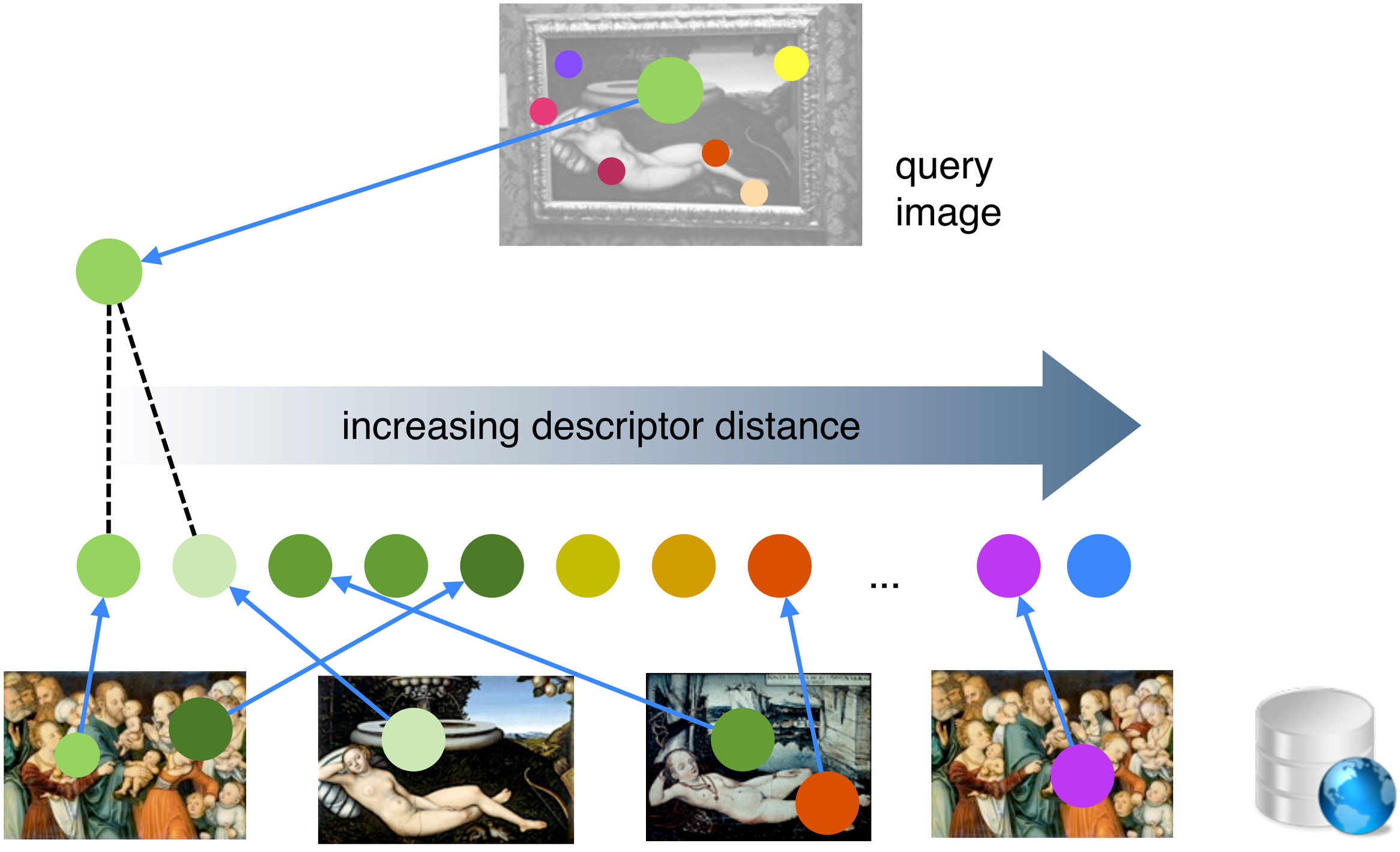
BRIEF

Jets

...

Image retrieval pipeline

Step 1: find neighbours of each query descriptor



H. Jégou, M. Douze, and C. Schmid. Exploiting descriptor distances for precise image search. Technical Report 7656, INRIA, 2011.

Image retrieval pipeline

Step 2: each query descriptor casts a vote for each DB image

vote strength $\max\{\mathbf{d}_k - \mathbf{d}_i, 0\}$

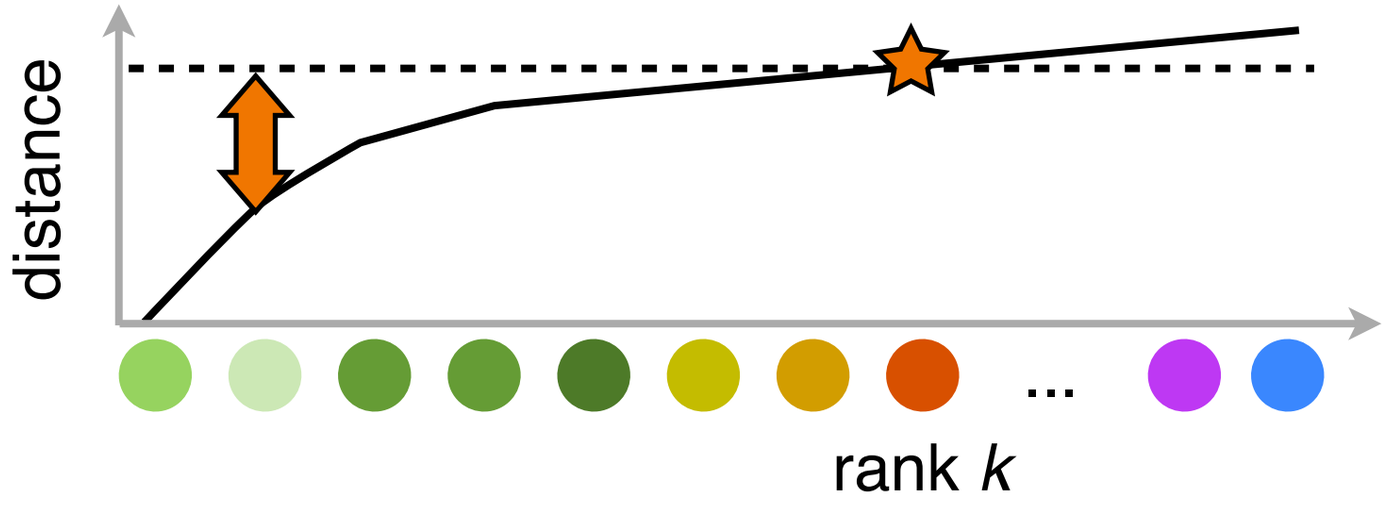
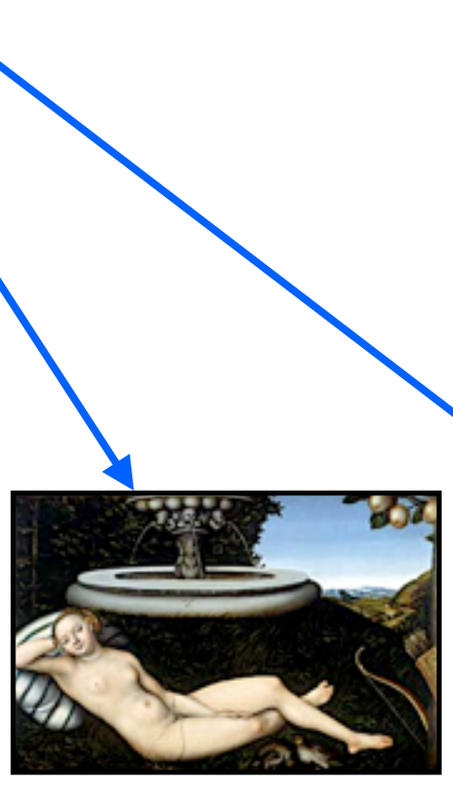
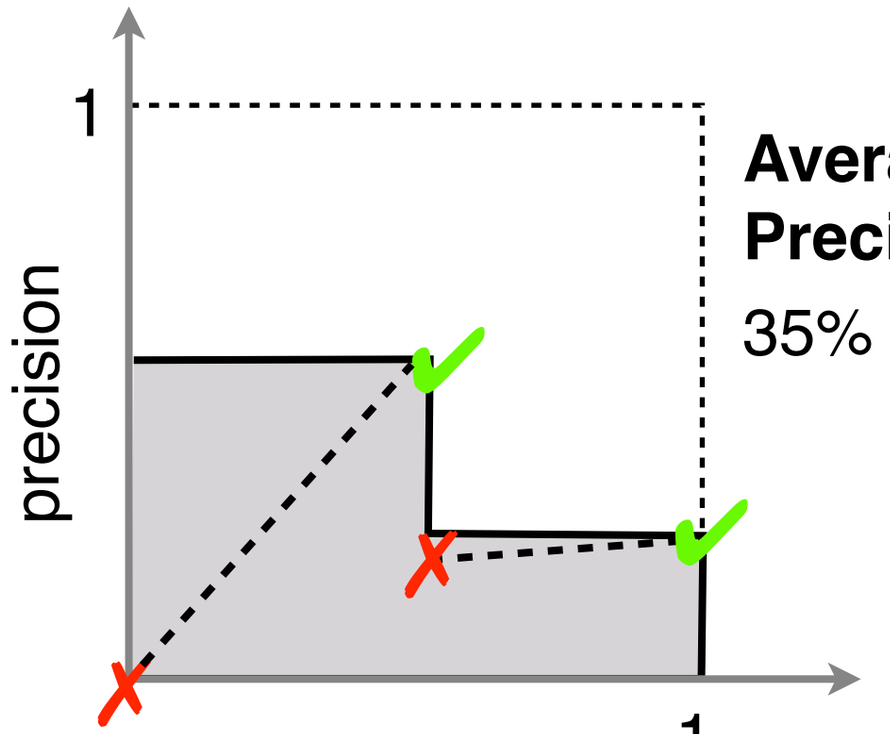


Image retrieval pipeline

Step 3: sort DB images by decreasing total votes

query image





Average Precision (AP)

35%

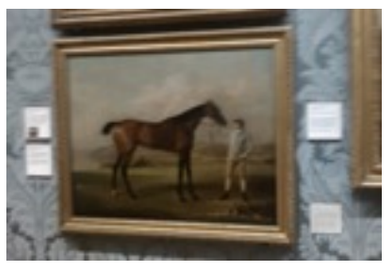
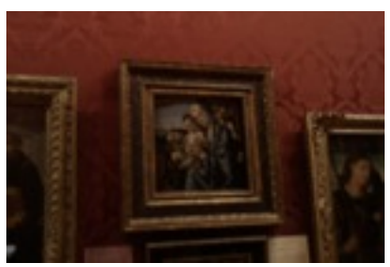
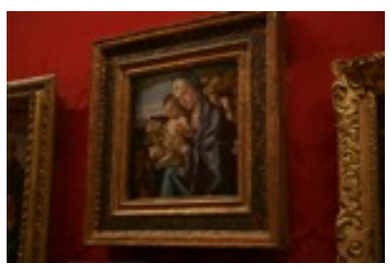
...

✗ ✔ ✗

decreasing total votes

Image retrieval pipeline

Step 4: Overall performance score

query	retrieval results			AP
	 ✗	 ✔	 ✗	35%
	 ✔	 ✗	 ✗	100%
	 ✔	 ✗	 ✔	75%
...	...	...	...	...

Mean Average Precision (mAP) 53%

Oxford 5K data

A retrieval benchmark dataset

Query

Retrieved Images

...

✔ ✗ ✔

- ~ 5K images of Oxford
  - For each of 58 queries
    - about XX matching images
    - about XX confounders images
- Larger datasets are possible, but slow for extensive evaluation
- Relative ranking of features seems to be representative

Indirect evaluation

Repeatability and matching score

Data: affine covariant testbed

Direct evaluation

Image retrieval

Data: oxford 5k

Software

VLBenchmarks

VLBenchmarks

A new easy-to-use benchmarking suite

http://www.vlfeat.org/benchmarks/index.html

- A novel MATLAB framework for feature evaluation
  - **Repeatability and matching scores**
    - VGG affine testbed
  - **Image retrieval**
    - Oxford 5K
- **Goodies**
  - Simple to use MATLAB code
  - Automatically download datasets & run evaluations
  - Backward compatible with published results

VLBenchmarks

Obtaining and installing the code

- **Installation**
  - Download the latest version
  - Unpack the archive
  - Launch MATLAB and type

```
>> install
```

- **Requirements**
  - MATLAB R2008a (7.6)
  - A C compiler (e.g. Visual Studio, GCC, or Xcode)
  - Do not forget to setup MATLAB to use your C compiler

```
mex -setup
```

Example usage


```
import datasets.*;
import localFeatures.*;
import benchmarks.*;

detector = VlFeatSift() ;

dataset = VggAffineDataset('category','graf') ;

benchmark = RepeatabilityBenchmark() ;

repeatability = ...
      benchmark.testFeatureExtractor(detector, ...
                                     dataset.getTransformation(2), ...
                                     dataset.getImagePath(1), ...
                                     dataset.getImagePath(2)) ;
```

choose a detector
(Harris Affine, VGG version)

choose a dataset
(graffiti sequence)

choose test
(detector repeatability)

run the evaluation
(repeatability = 0.66)

Testing on a sequence of images

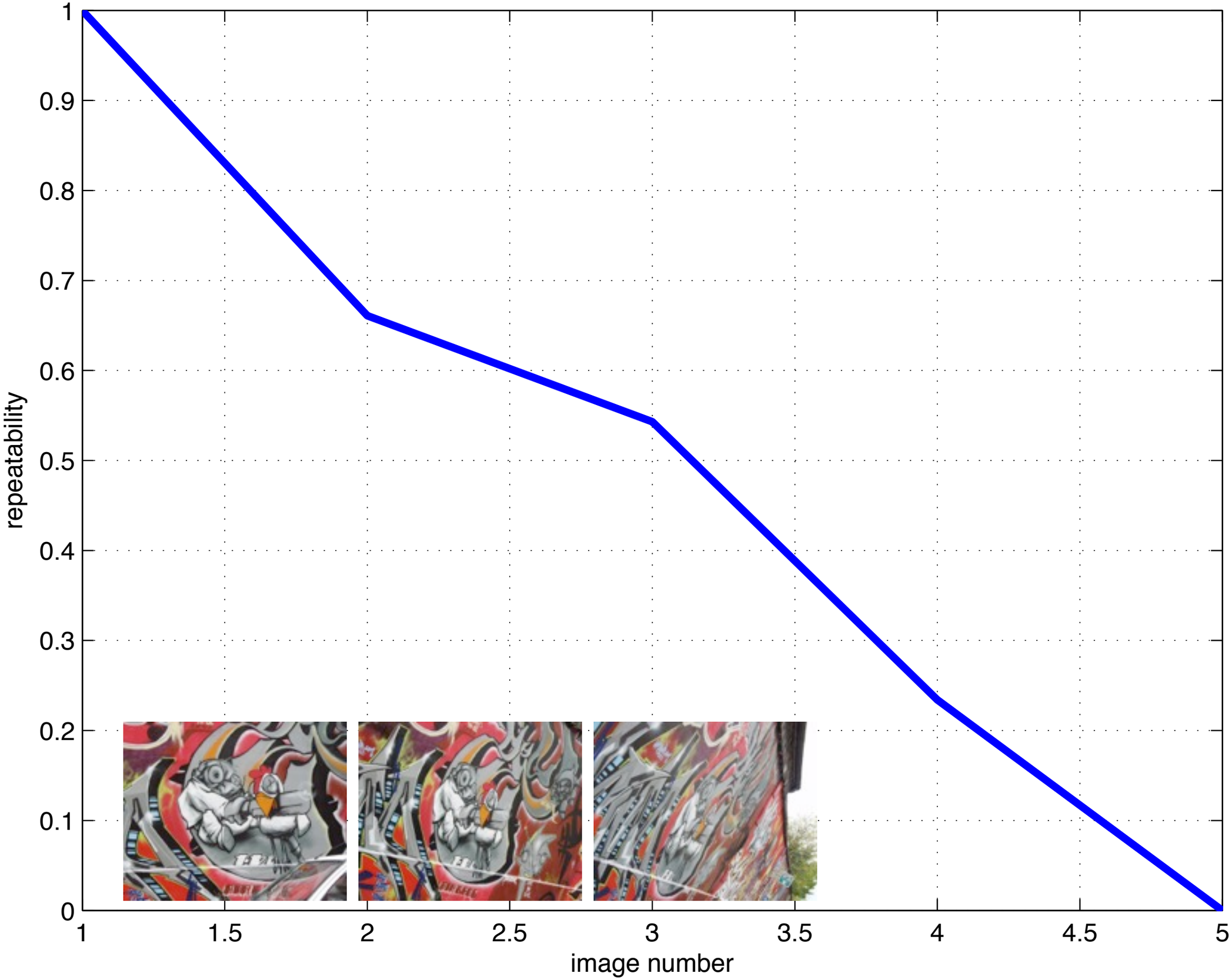

```
import datasets.*; import localFeatures.*; import benchmarks.*;

detector = VlFeatSift() ;
dataset = VggAffineDataset('category','graf') ;
benchmark = RepeatabilityBenchmark() ;

for j = 1:5
  repeatability(j) = ...
        benchmark.testFeatureExtractor(detector, ...
                                       dataset.getTransformation(j), ...
                                       dataset.getImagePath(1), ...
                                       dataset.getImagePath(j)) ;
end

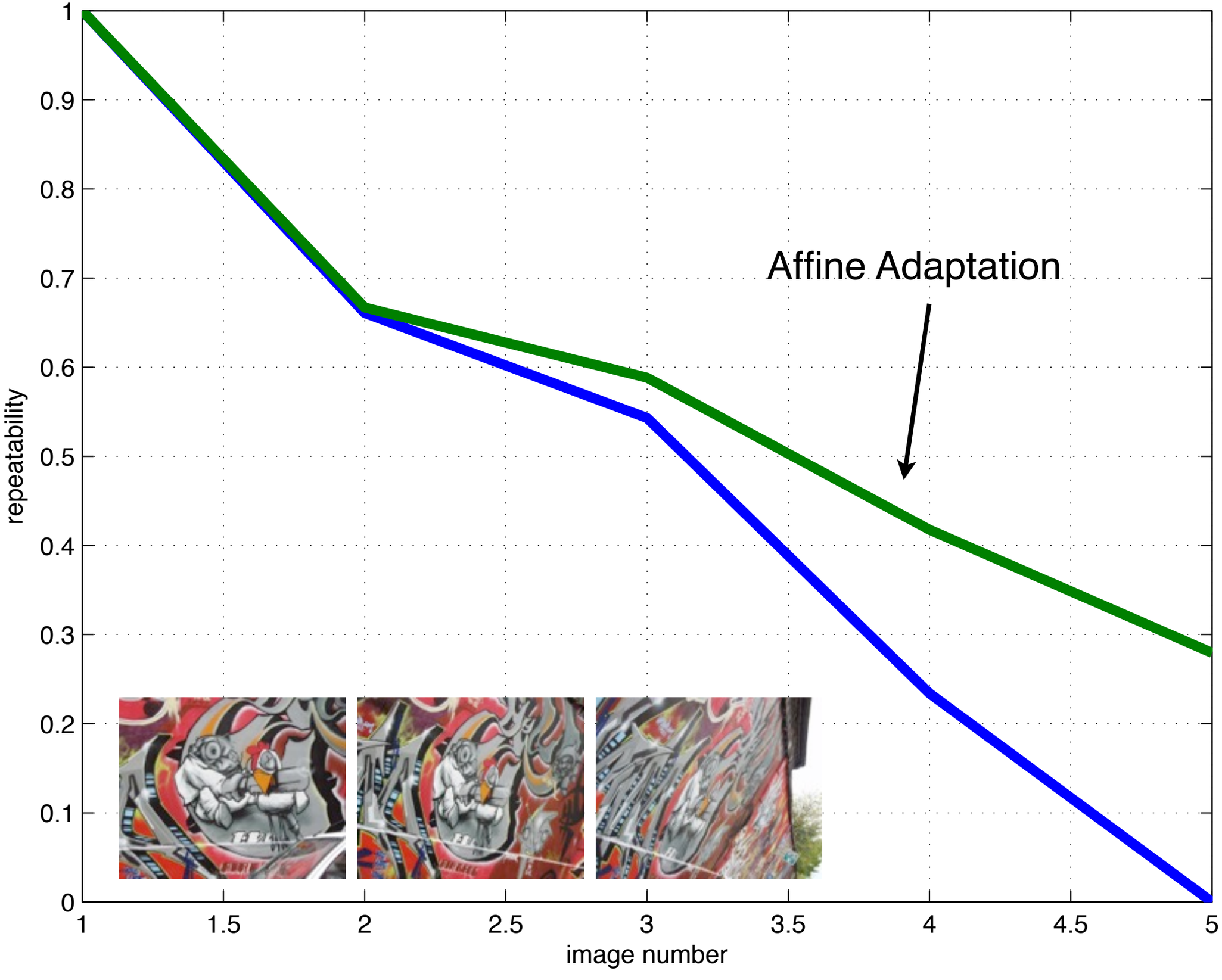
clf ; plot(repeatability, 'linewi
xlabel('image number') ;
ylabel('repeatability') ;
grid on ;
```

Use **parfor** on a cluster!


repeatability
image number
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
1.5
2
2.5
3
3.5
4
4.5
5

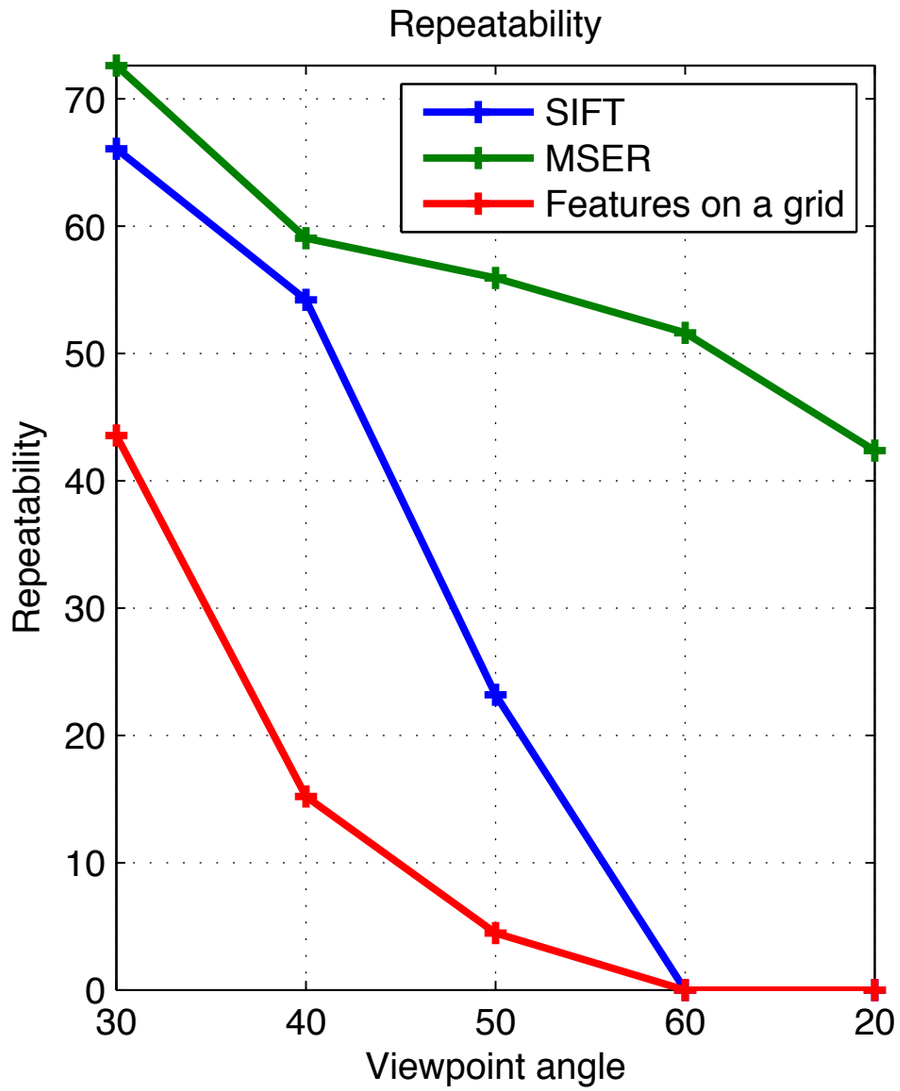
Comparing two features

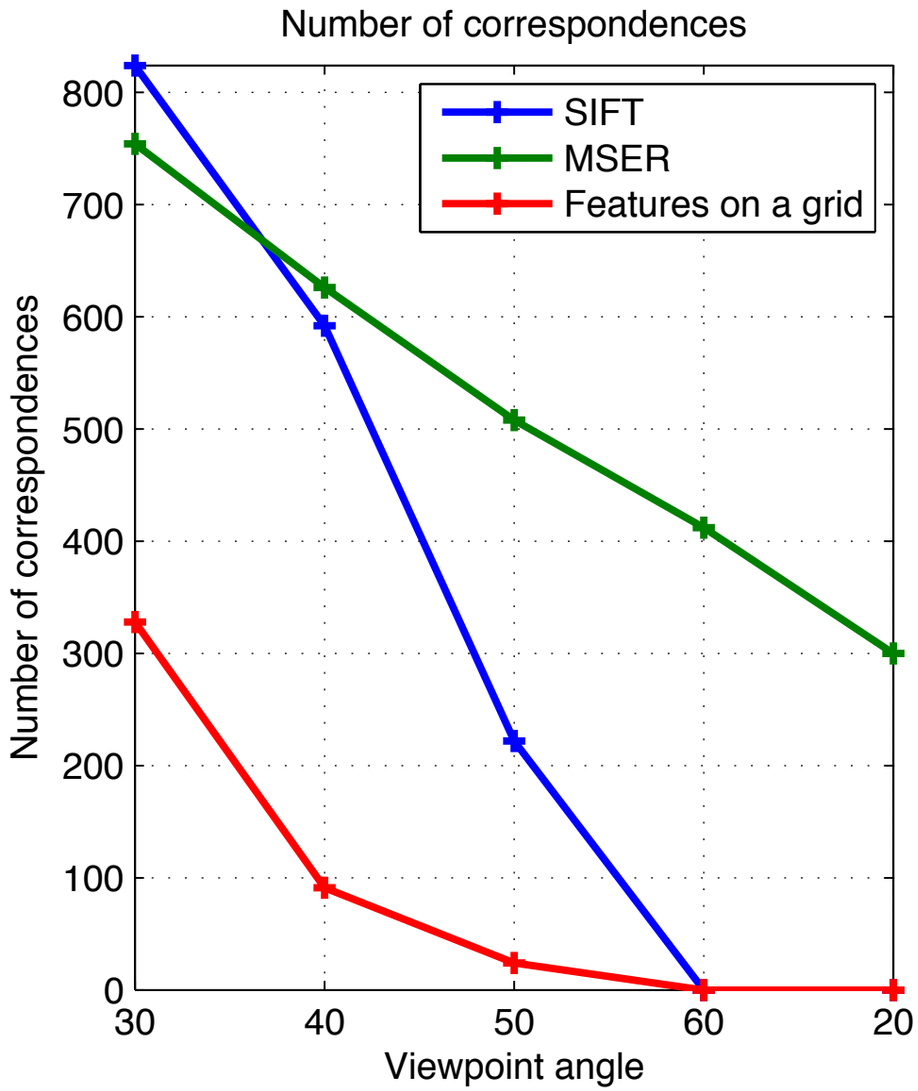

```
import datasets.*; import localFeatures.*; import benchmarks.*;
detectors{1} = VlFeatSift() ;
detectors{2} = VlFeatCovdet('EstimateAffineShape', true) ;
dataset = VggAffineDataset('category','graf') ;
benchmark = RepeatabilityBenchmark() ;
for d = 1:2
  for j = 1:5
    repeatability(j,d) = ...
        benchmark. testFeatureExtractor(detectors{d}, ...
                                        dataset.getTransformation(j), ...
                                        dataset.getImagePath(1), ...
                                        dataset.getImagePath(j)) ;
  end
end
clf ; plot(repeatability, 'linewidth', 4) ;
xlabel('image number') ;
ylabel('repeatability') ;
grid on ;
```


Affine Adaptation
repeatability
image number
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
1.5
2
2.5
3
3.5
4
4.5
5

Example

- **Compare the following features**
  - SIFT, MSER, and features on a grid
  - on the Graffiti sequence
  - for repeatability and number of correspondence





Backward compatible

- Previously published results can be easily reproduced
  - if interested, try the script `reproduceIjcv05.m`

A Comparison of Affine Region Detectors

K. MIKOLAJCZYK
University of Oxford, OX1 3PJ, Oxford, United Kingdom
km@robots.ox.ac.uk

T. TUYTELAARS
University of Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium
tuytelaa@esat.kuleuven.be

C. SCHMID
INRIA, GRAVIR-CNRS, 655, av. de l'Europe, 38330, Montbonnot, France
schmid@inrialpes.fr

A. ZISSERMAN
University of Oxford, OX1 3PJ, Oxford, United Kingdom
az@robots.ox.ac.uk

J. MATAS
Czech Technical University, Karlovo Namesti 13, 121 35, Prague, Czech Republic

Other useful tricks

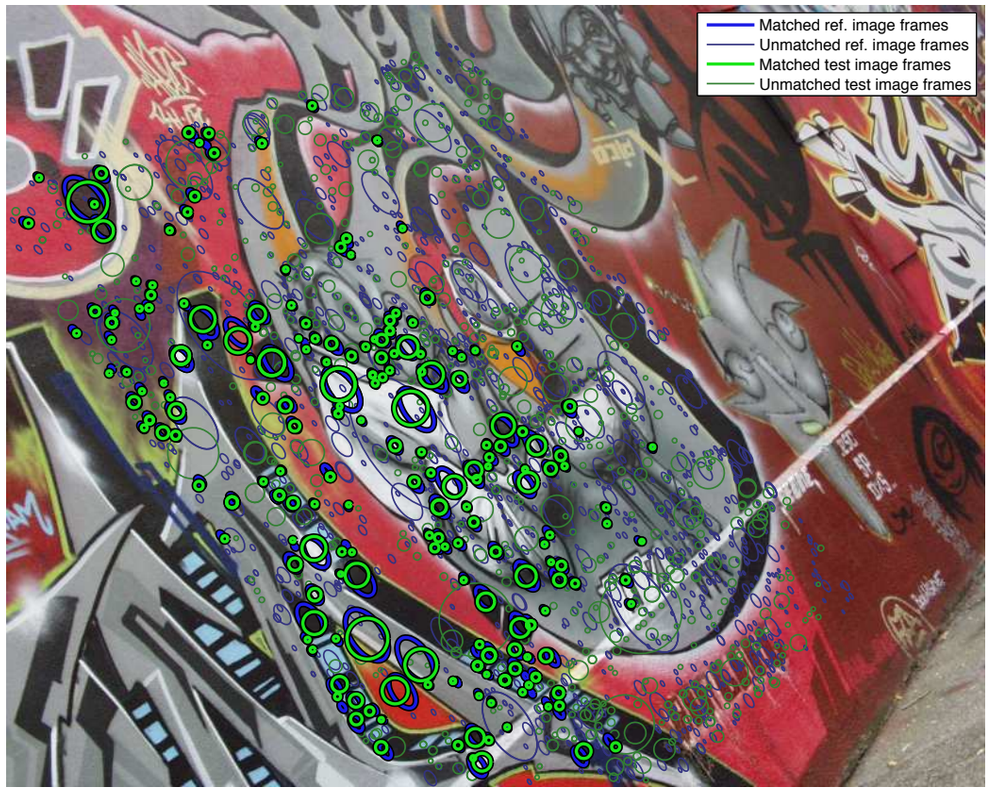
- Compare different parameter settings

```
detectors{1} = VggAffine('Detector','haraff', 'Threshold ', 500) ;
detectors{2} = VggAffine('Detector','haraff', 'Threshold ', 1000) ;
```

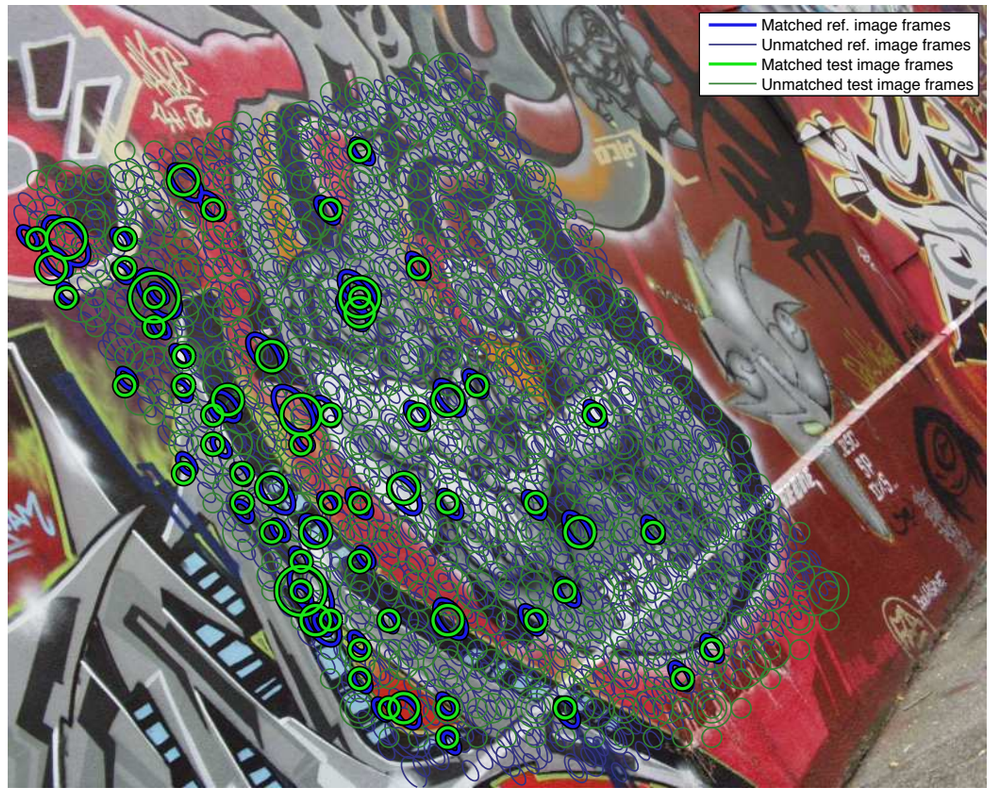
- Visualising matches

```
[~ ~ matches reprojFrames] = benchmark.testFeatureExtractor( ... )
...
benchmarks.helpers.plotFrameMatches(matches, reprojFrames)
```

SIFT Matches with 4 image (VggAffineDataset−graf dataset).



Matches using mean−variance−median descriptor with 4 image (VggAffineDataset−graf dataset).



Other benchmarks

- **Detector matching score**

```
benchmark = RepeatabilityBenchmark('mode','MatchingScore') ;
```

- **Image retrieval**
  - Example: Oxford 5K lite
  - mAP evaluation

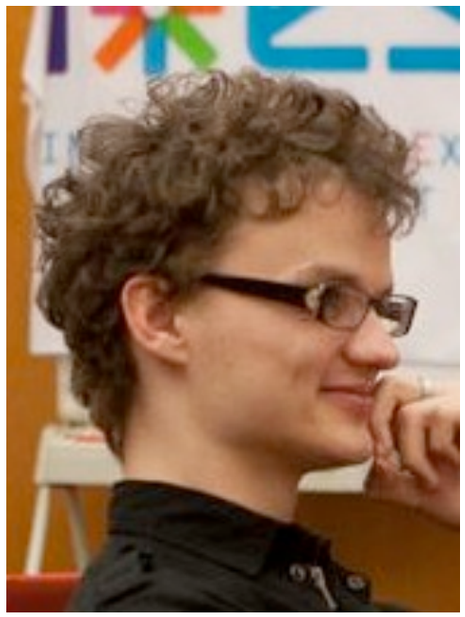
```
dataset = VggRetrievalDataset('Category','oxbuild',
                              'BadImagesNum',100);
benchmark = RetrievalBenchmark() ;
mAP = benchmark.testFeatureExtractor(detectors{d}, dataset);
```

Summary

http://www.vlfeat.org/benchmarks/index.html

- **Benchmarks**
  - Indirect: repeatability and matching score
  - Direct: image retrieval
- **VLBenchmarks**
  - a simple to use MATLAB framework
  - convenient
- **The future**
  - Existing measures have many shortcomings
  - Hopefully better benchmarks will be available soon
  - And they will be added to VLBenchmarks for your convenience

Credits



Karel Lenc

Varun Gulshan

PASCAL2

HARVEST Programme

Krystian Mikolajczyk

Tinne Tuytelaars

Jiri Matas

Cordelia Schmid

Andrew Zisserman

Thank you for coming!

VLFeat

http://www.vlfeat.org/

VLBenchmarks

http://www.vlfeat.org/benchmarks/