SIFT in CUDA

In the Horizon 2020 projects POPART and LADIO

project page: http://www.popartproject.eu http://ladioproject.eu

PopSift repository: https://github.com/alicevision/popsift

S - Scale

I - Invariant

F - Feature

T - Transform

Distinctive Image Features from Scale-Invariant Keypoints

David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

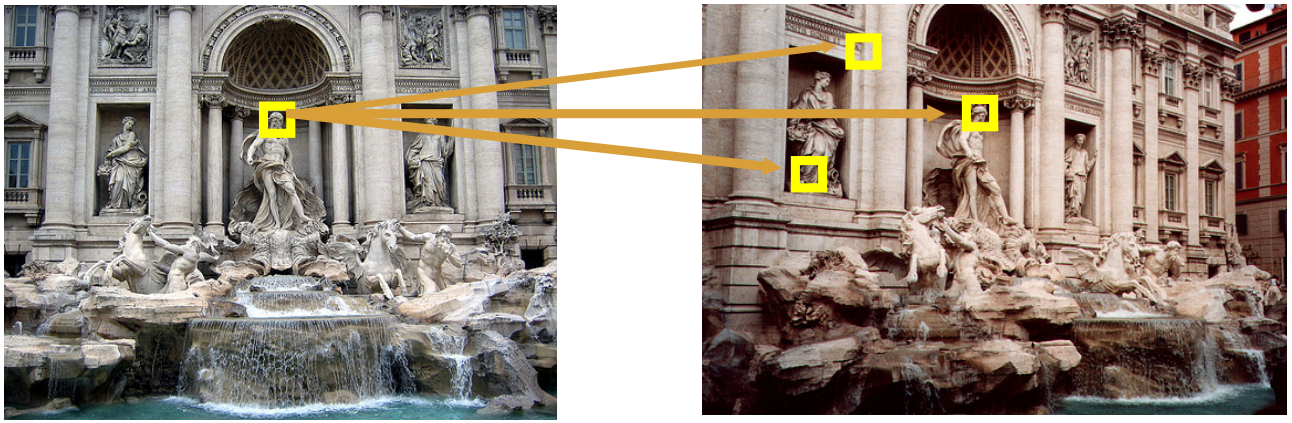
January 5, 2004

Abstract

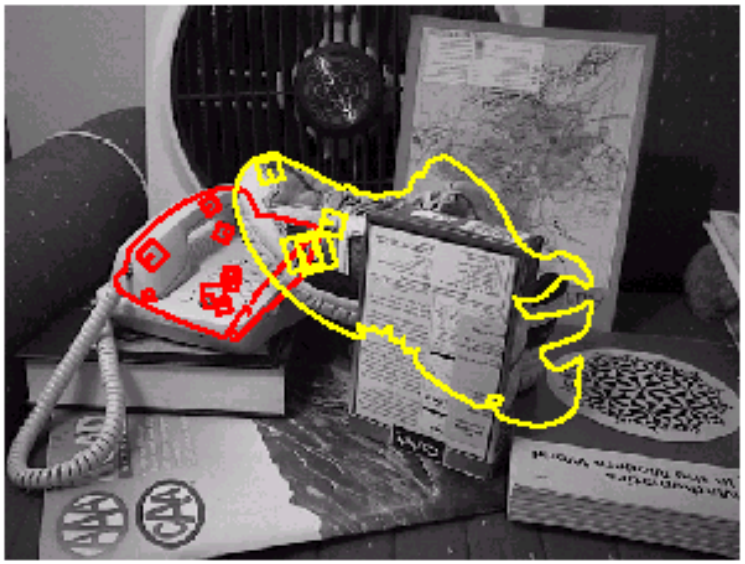
This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

Matching SIFT features

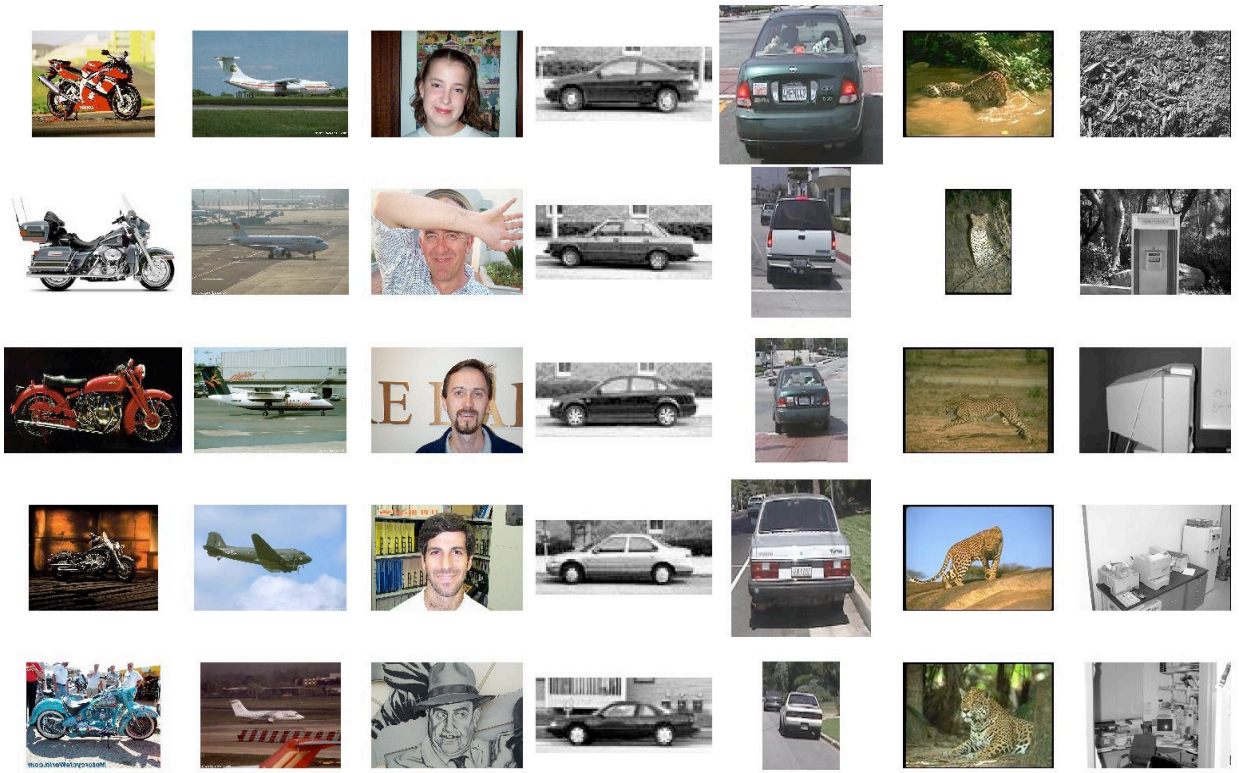
- Given a feature in I_1, how to find the best match in I_2?
  1. Define distance function that compares two descriptors
  2. Test all the features in I_2, find the one with min distance



Object Recognition



Object Categorization



Location recognition

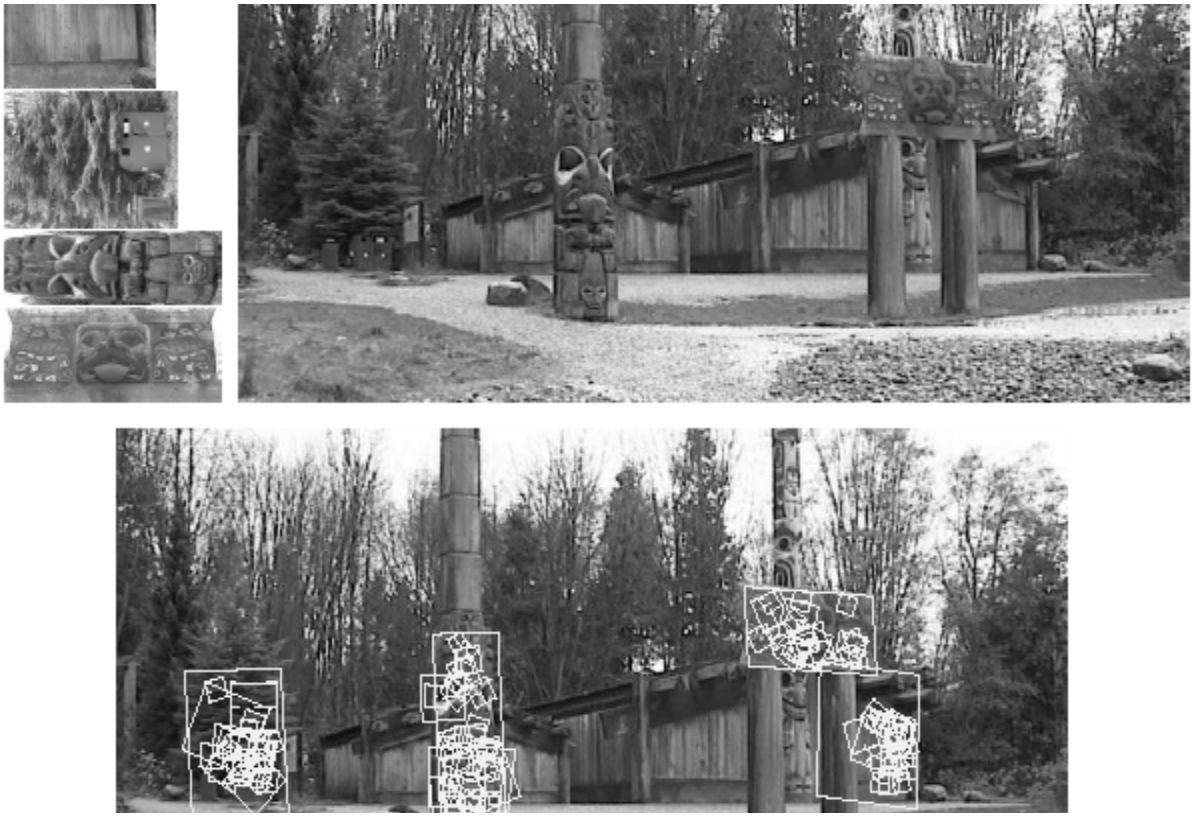


Image retrieval

change in viewing angle

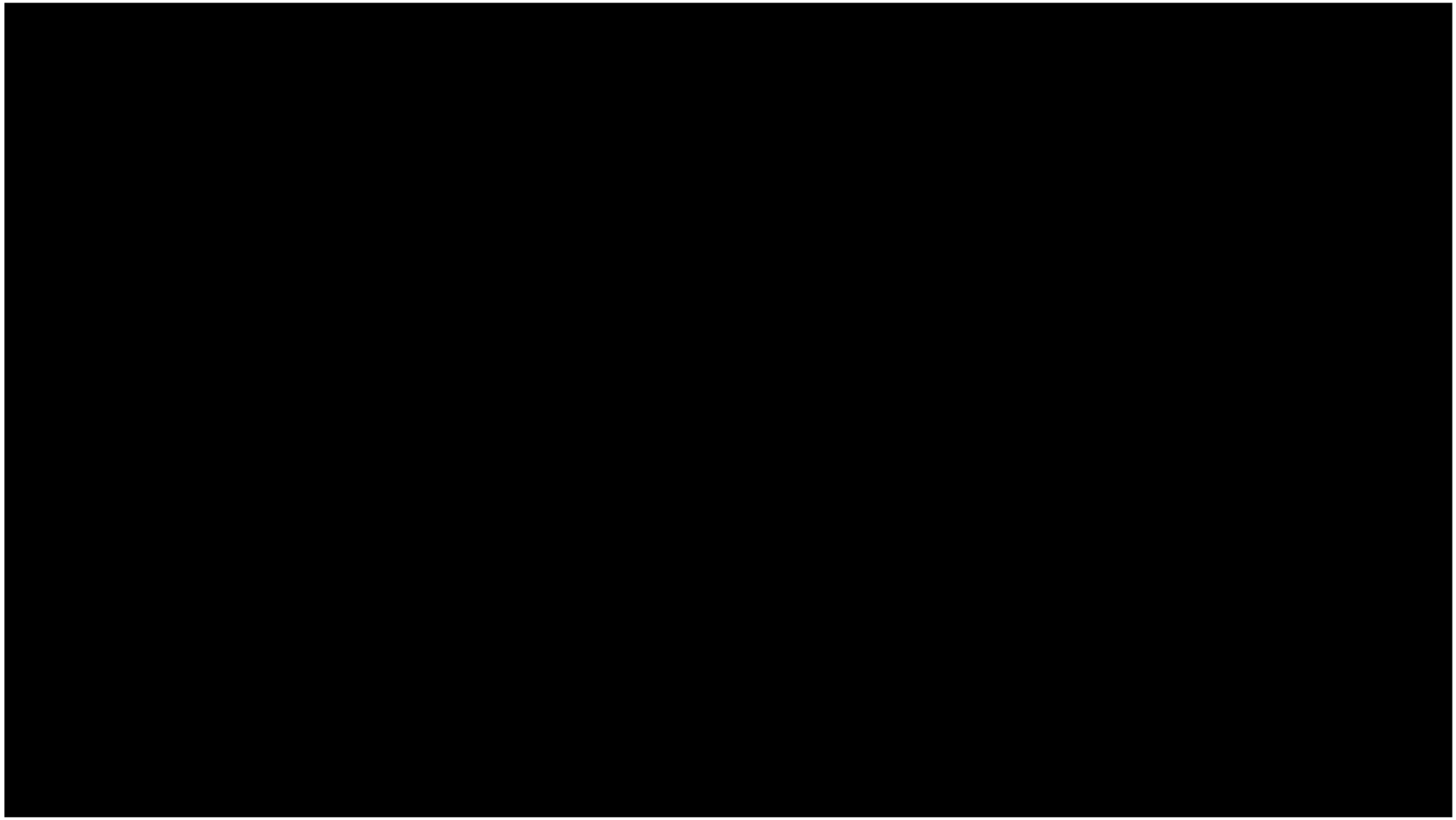
• • •

> 5000 images

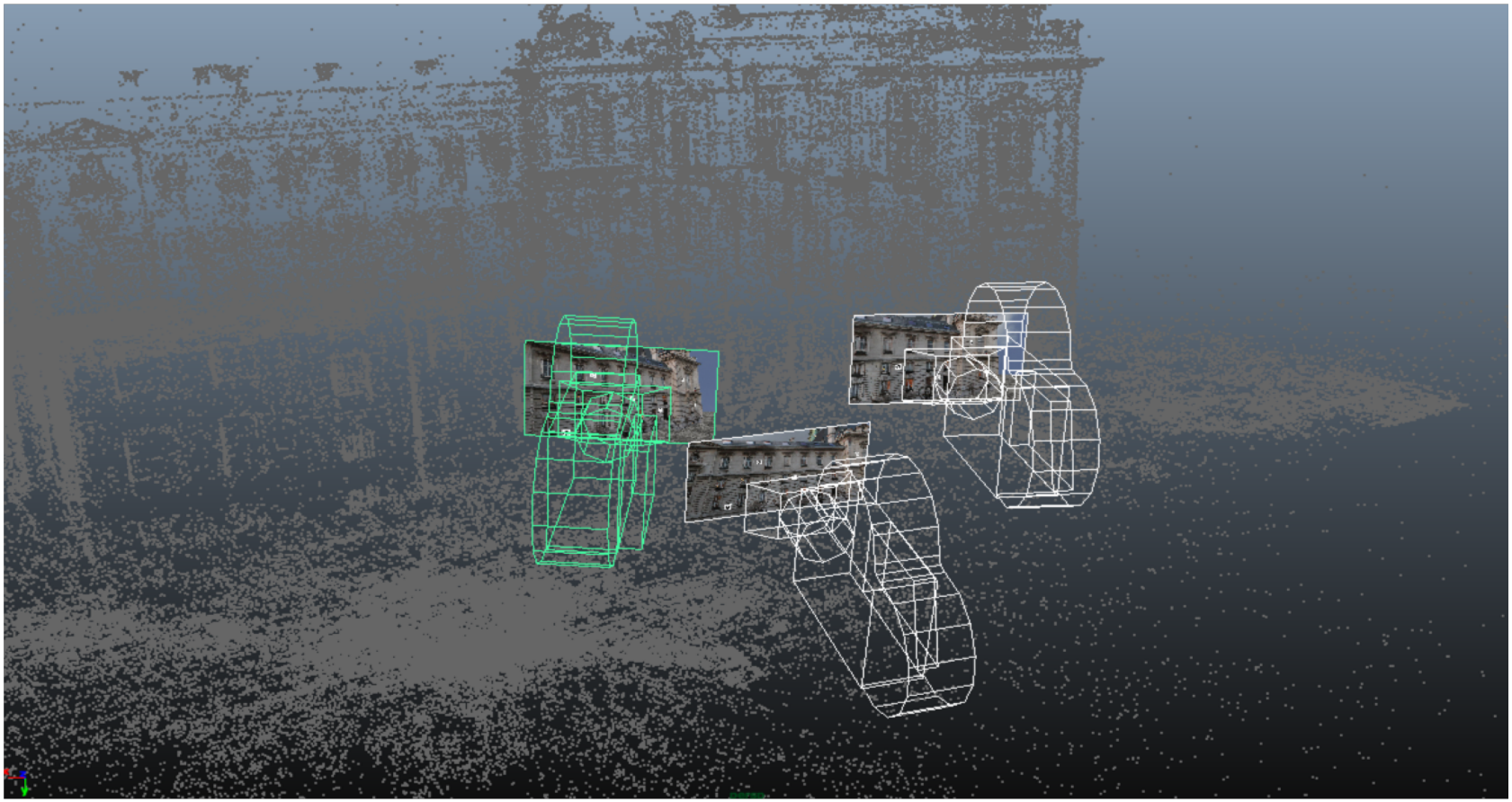
Matches

22 correct matches

3D reconstruction

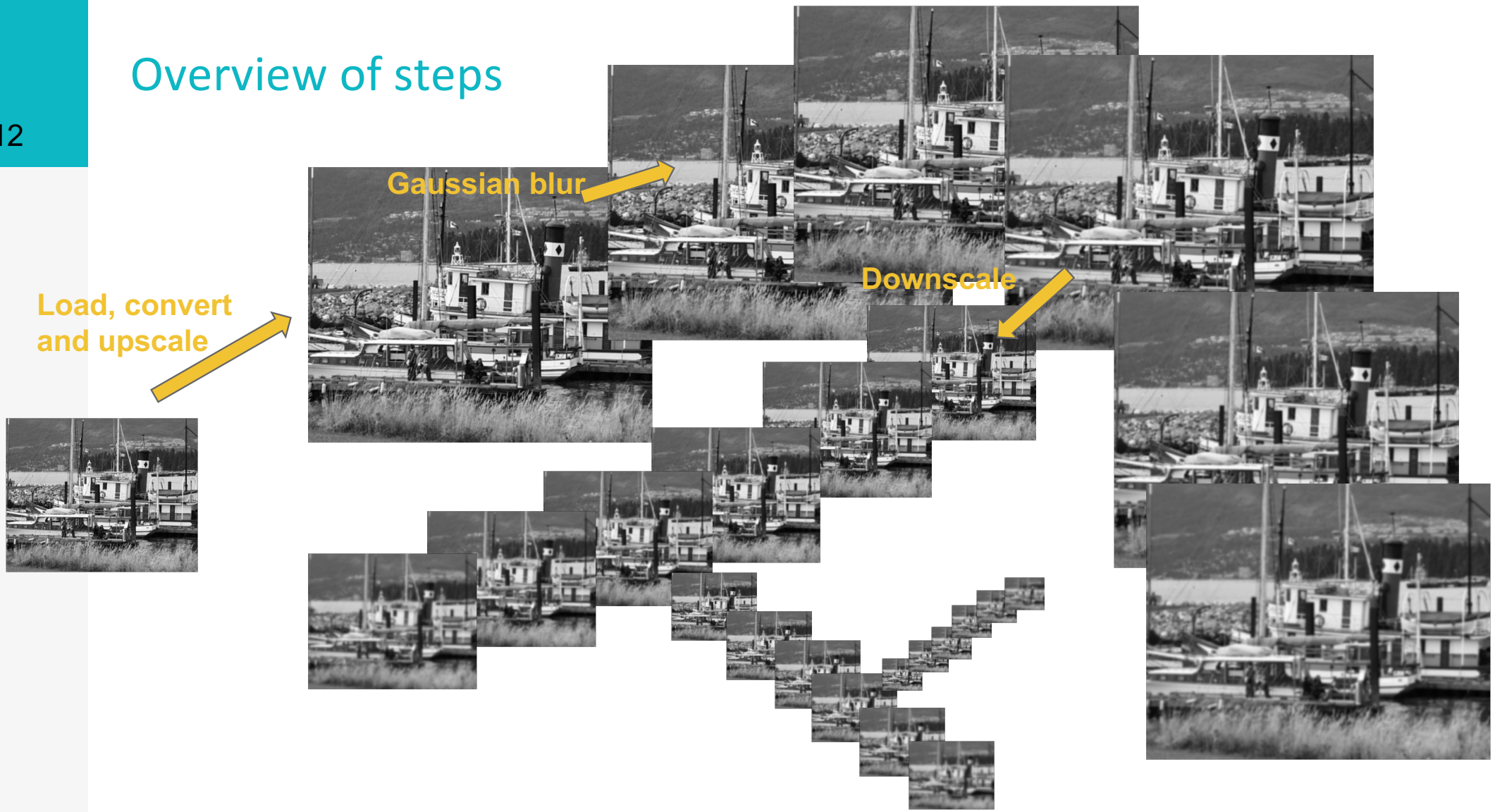


Camera tracking

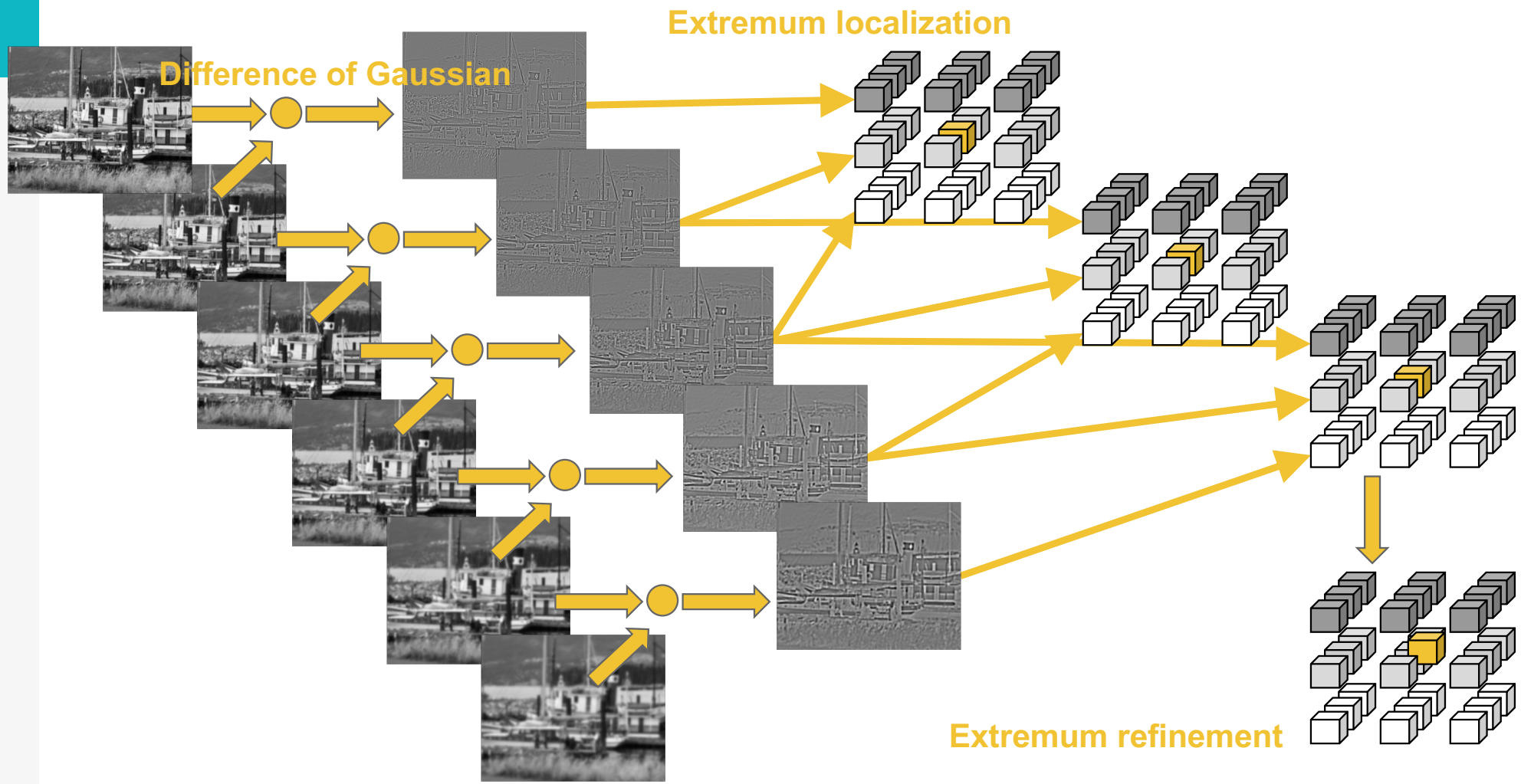


SIFT matching illustration

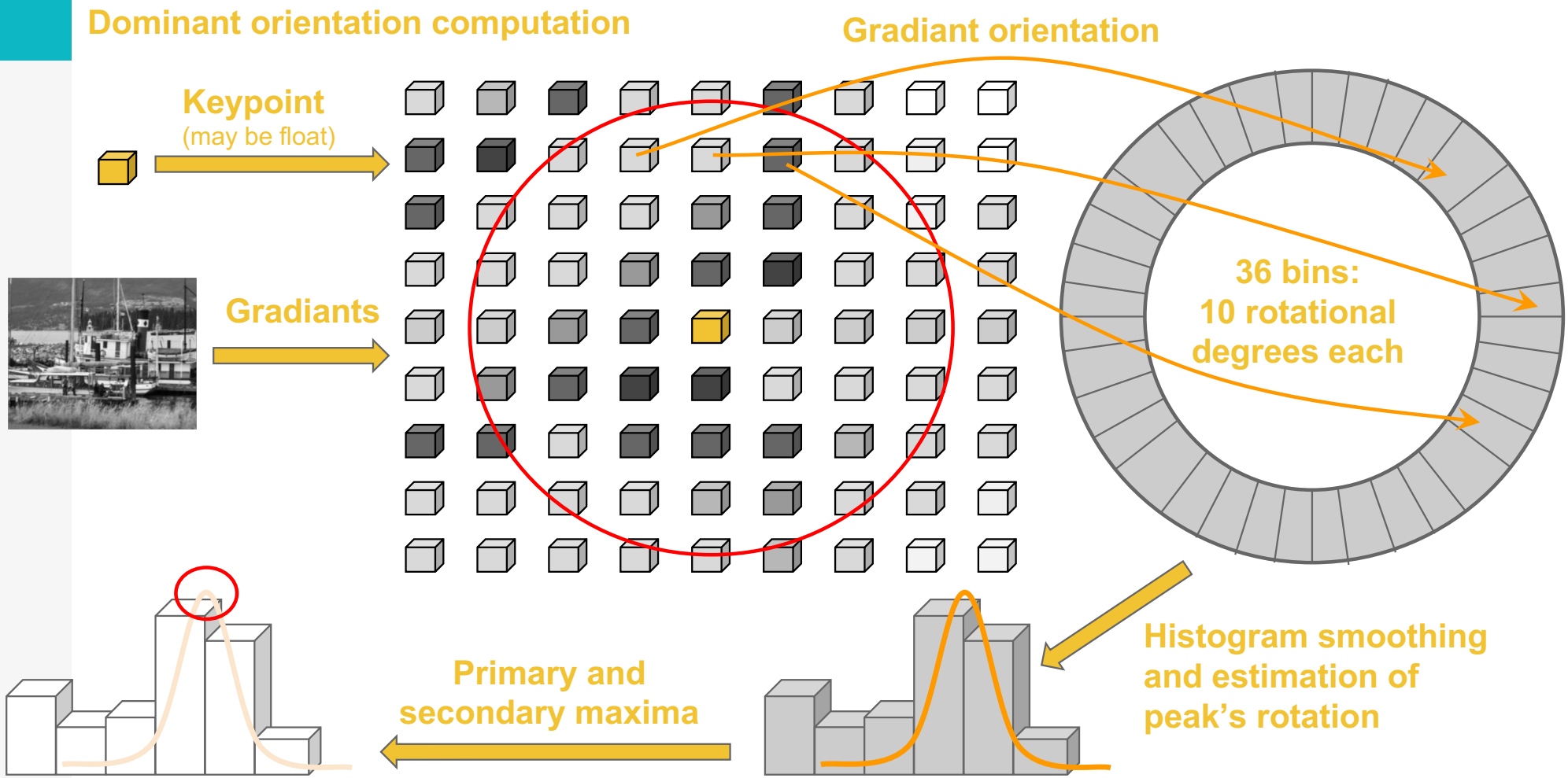
Overview of steps



Overview of steps

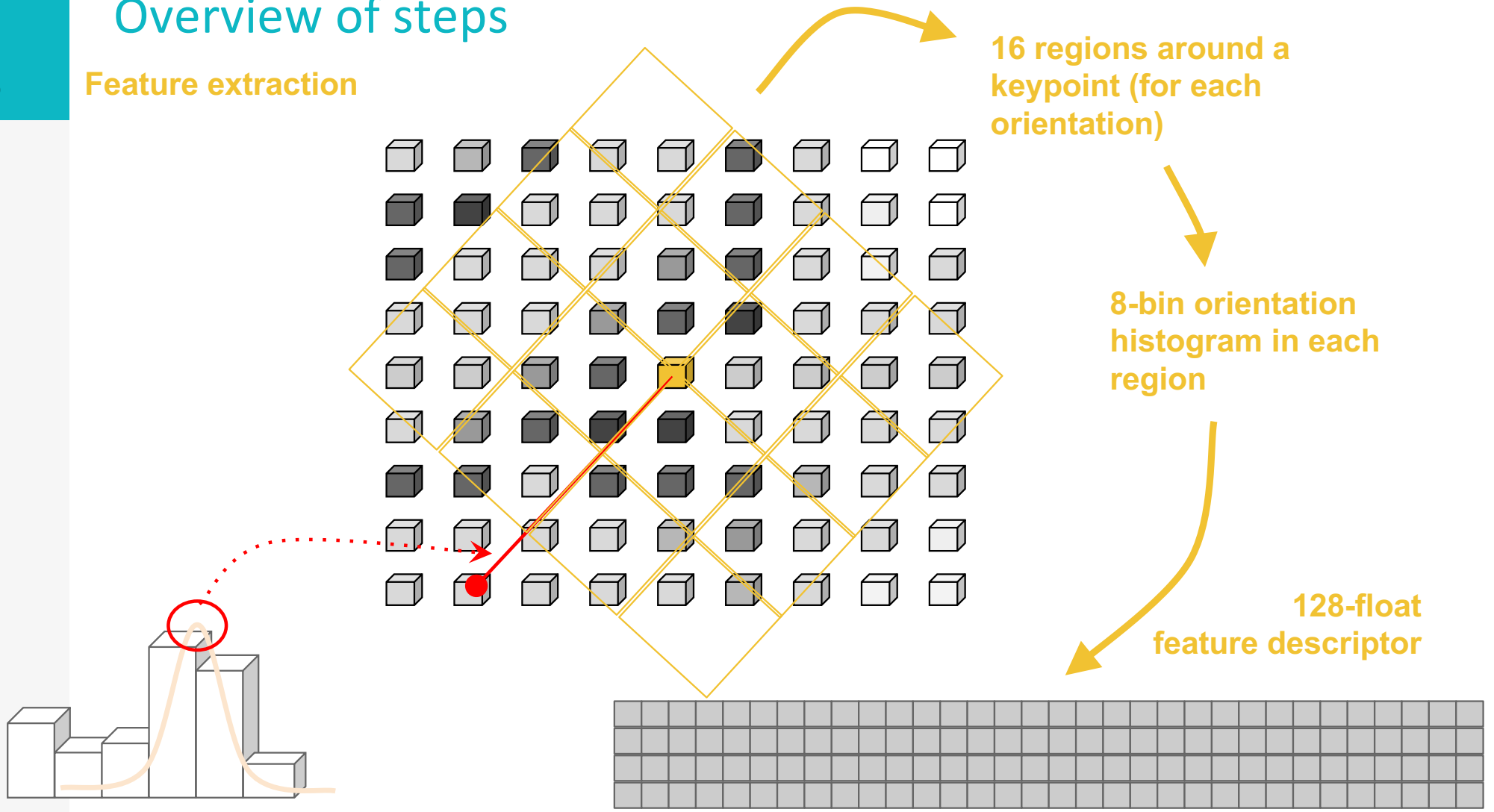


Overview of steps



Overview of steps

Feature extraction



CUDA tuning

Released under MPLv2

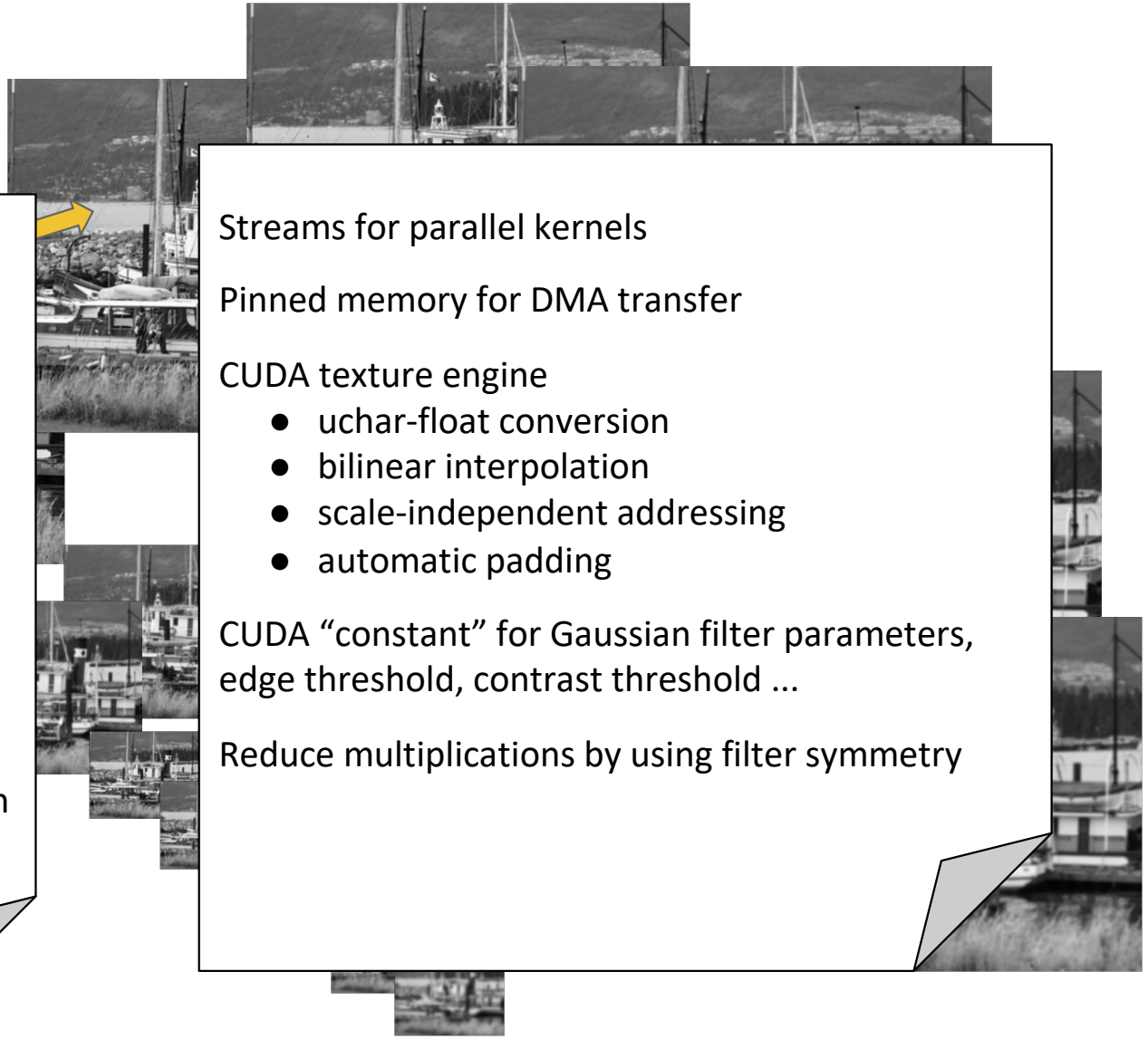
Feature vector output is compatible with VLFeat (drop-in replacement)

Arbitrary input scaling to trade speed for accuracy

Single-stream real-time extraction for 1920x1080 with upscaling on the GTX 980 Ti

Output value scaling by powers of 2

Very similar in terms of extracted and matched features to VLFeat (and better than OpenCV)



Streams for parallel kernels

Pinned memory for DMA transfer

CUDA texture engine

- uchar-float conversion
- bilinear interpolation
- scale-independent addressing
- automatic padding

CUDA “constant” for Gaussian filter parameters, edge threshold, contrast threshold ...

Reduce multiplications by using filter symmetry

#

Create pyramid

Input image: `struct ImageBase` in https://github.com/alicevision/popsift/blob/develop/src/popsift/s_image.h line 110

Pyramid construction in `Pyramid::build_pyramid` https://github.com/alicevision/popsift/blob/develop/src/popsift/s_pyramid_build.cu line 460
case `conf.getGaussMode() == Config::VLFeat_Relative` line 517

for the first image in the first octave, this call `Pyramid::horiz_from_input_image` which is in line 97

Create pyramid

calls `Pyramid::horiz_from_input_image`
starts the kernel

`gauss::normalizedSource::horiz`
with blocks of 128 threads in one dimensions, grid configuration determines that we use one for every pixel in the image!
https://github.com/alicevision/popsift/blob/develop/src/popsift/s_pyramid_build_ra.cu in line 18

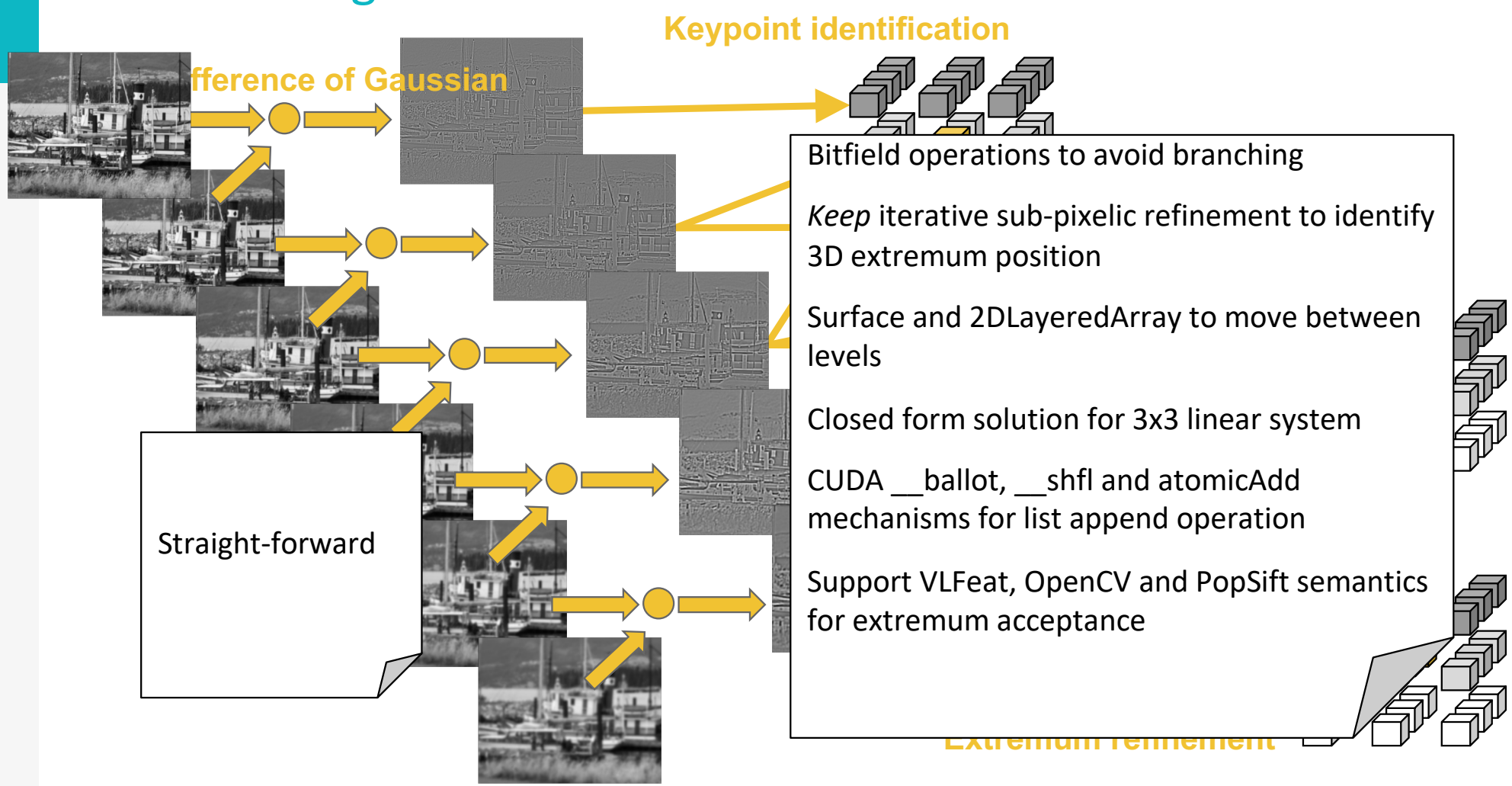
for the all other images, `Pyramid::build_pyramid` calls
`Pyramid::horiz_from_prev_level`
which is in line 250

Create pyramid

calls `Pyramid::horiz_from_prev_level`
starts the kernel

`gauss::absoluteSourceInterpolated::horiz`
with blocks of 128 threads in one dimensions, grid configuration determines that we use one for every pixel in the image!
https://github.com/alicevision/popsift/blob/develop/src/popsift/s_pyramid_build_ai.cu in line 18

CUDA tuning


Keypoint identification
fference of Gaussian
Bitfield operations to avoid branching
Keep iterative sub-pixelic refinement to identify 3D extremum position
Surface and 2DLayeredArray to move between levels
Closed form solution for 3x3 linear system
CUDA __ballot, __shfl and atomicAdd mechanisms for list append operation
Support VLFeat, OpenCV and PopSift semantics for extremum acceptance
Straight-forward

#

Find extrema

https://github.com/alicevision/popsift/blob/develop/src/popsift/s_extrema.cu

`find_extrema_in_dog`: line 512
`find_extrema_in_dog_sub`: line 302
`is_extremum`: line 58

CUDA tuning

Dominant orientation computation

Keypoint

Gradiants

Primary and secondary maxima

Gradiants not pre-computed due to keypoint sparsity

CUDA __shared__ memory for histogram

Support VLFeat, OpenCV and PopSift histogram smoothing methods

Warp shuffle operations to implement no-overhead parallel bitonic sort, finding all accepted orientations in parallel

optional CUDA Dynamic Parallelism for cards with Compute Capability >= 3.5

peak's rotation

#

Find orientation

https://github.com/alicevision/popsift/blob/develop/src/popsift/s_orientation.cu

`Pyramid::orientation`: line 248
`ori_par`: line 61

CUDA tuning

Feature extraction

Obviously, speed depends on the scene

Correlated with number of extrema

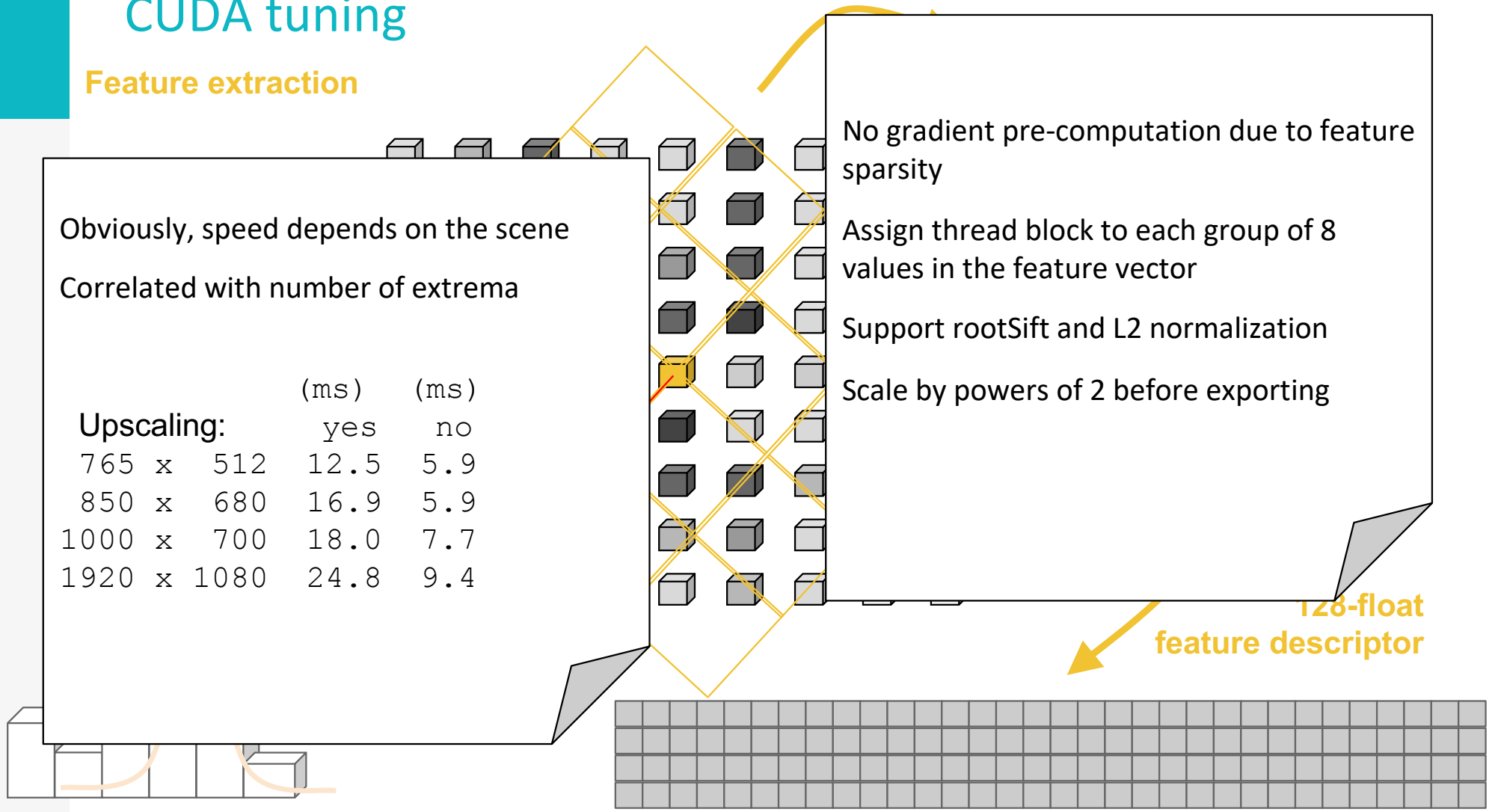
Upscaling:	(ms) yes	(ms) no
765 x 512	12.5	5.9
850 x 680	16.9	5.9
1000 x 700	18.0	7.7
1920 x 1080	24.8	9.4

No gradient pre-computation due to feature sparsity

Assign thread block to each group of 8 values in the feature vector

Support rootSift and L2 normalization

Scale by powers of 2 before exporting



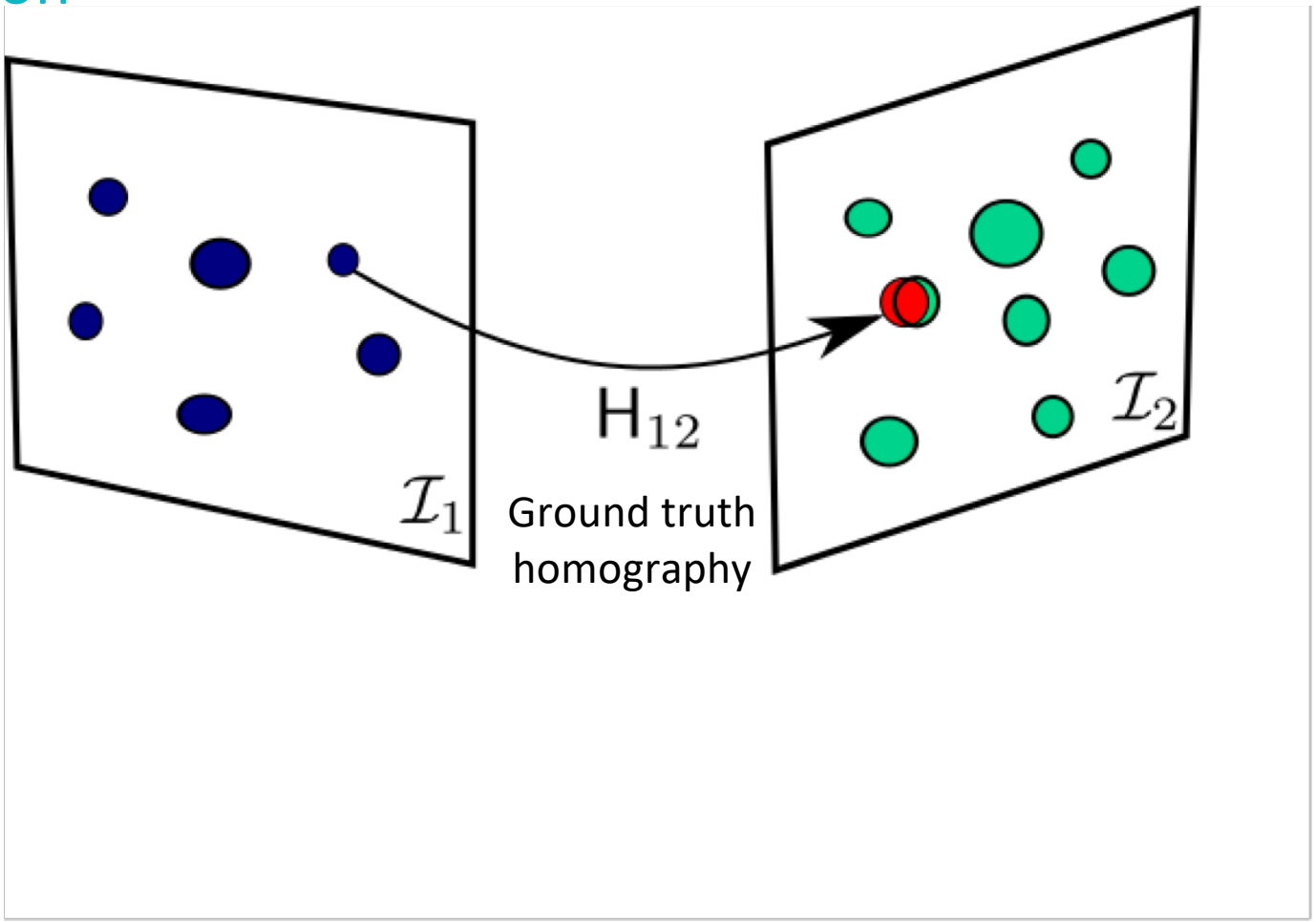
#

Compute descriptors

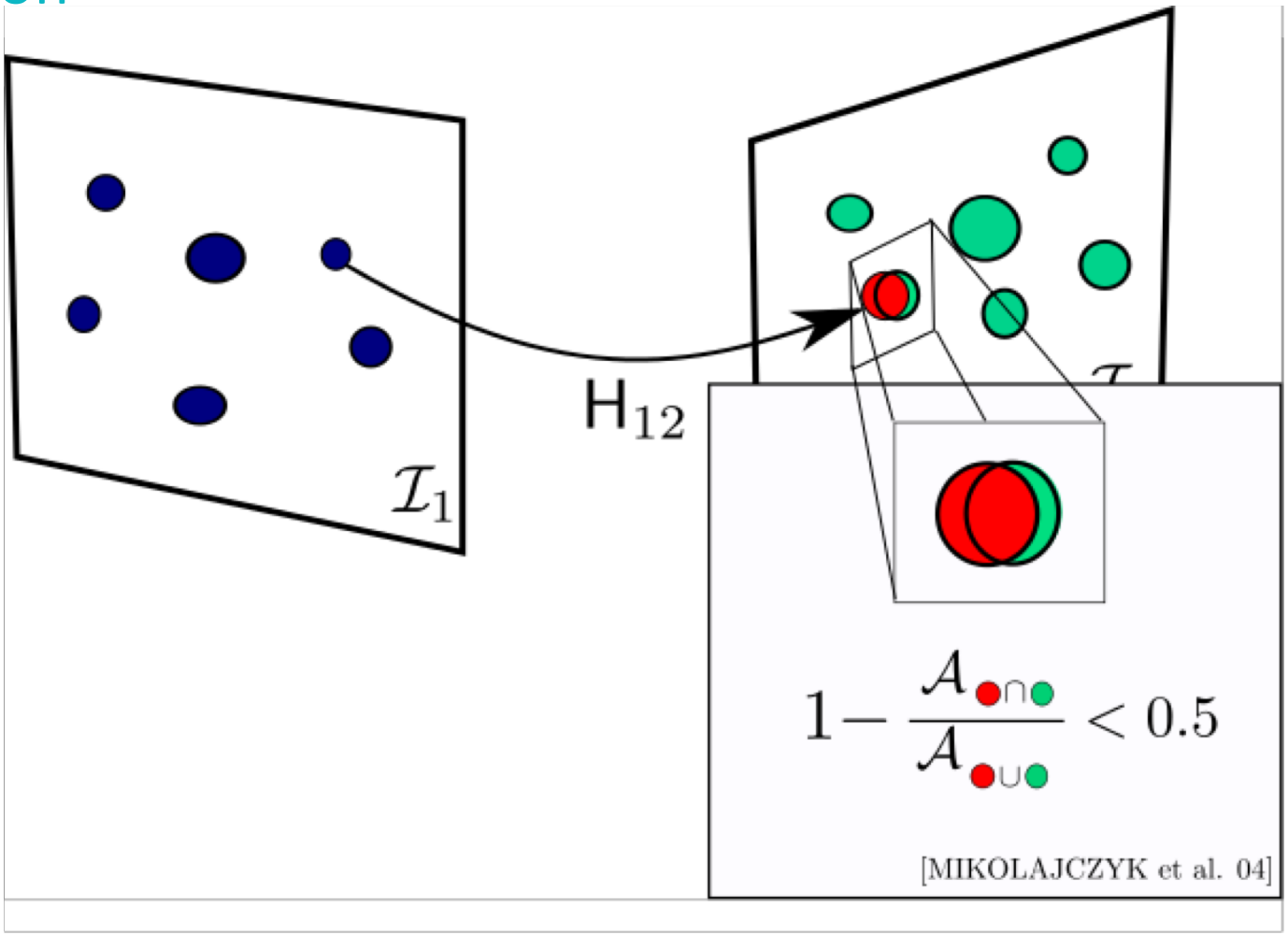
https://github.com/alicevision/popsift/blob/develop/src/popsift/s_desc_igrid.cu

`ext_desc_igrid`: line 61

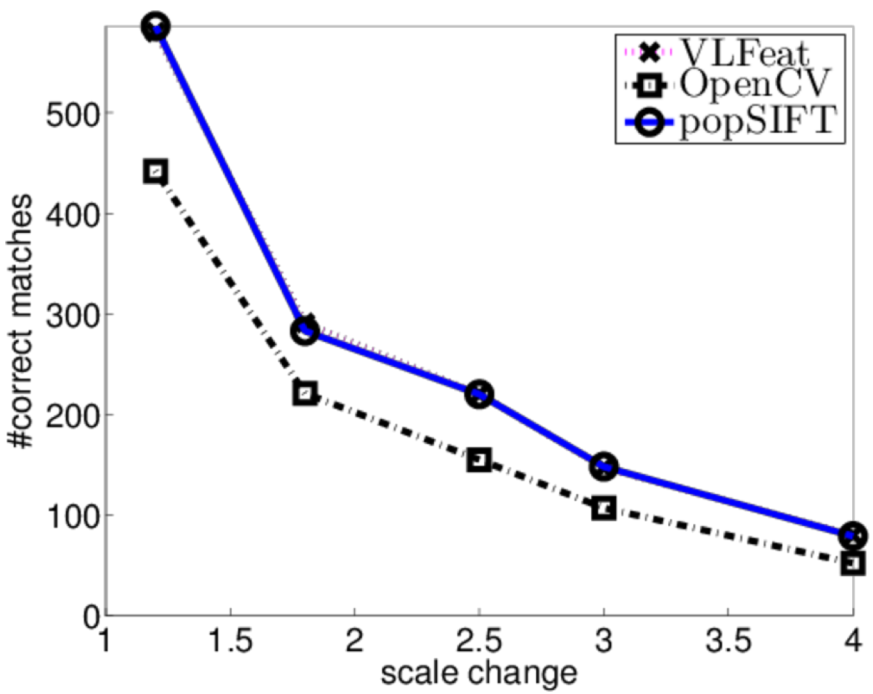
Evaluation

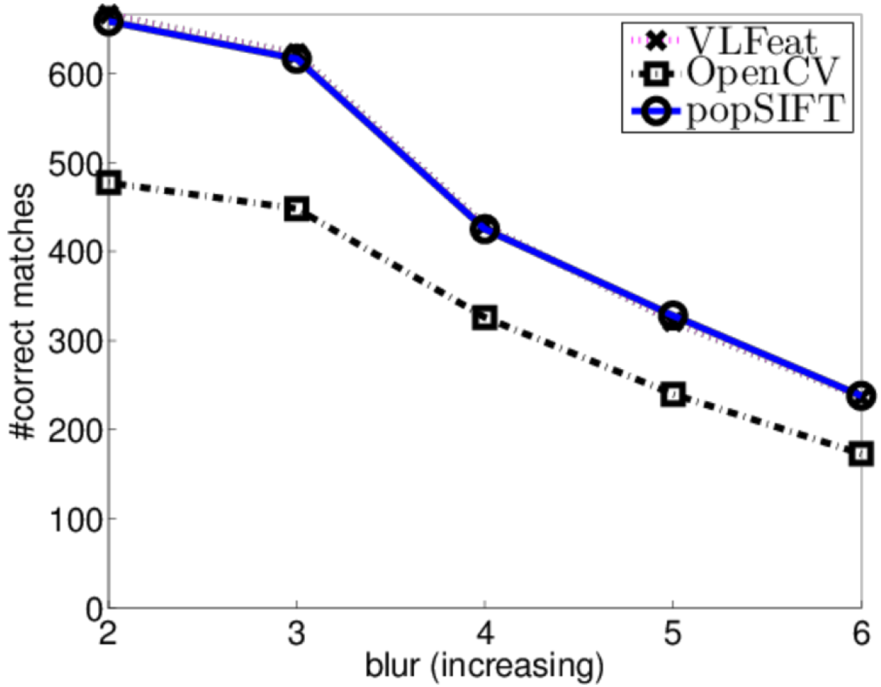


Evaluation

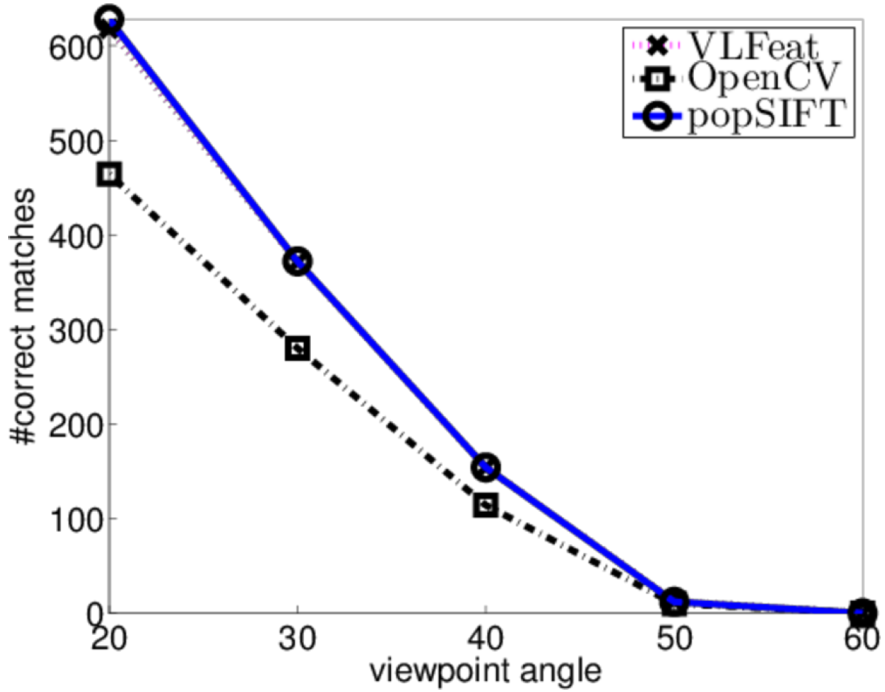

$\mathcal{I}_1$
H_{12}
$1 - \frac{\mathcal{A}_{\bullet\cap\bullet}}{\mathcal{A}_{\bullet\cup\bullet}} < 0.5$
[MIKOLAJCZYK et al. 04]

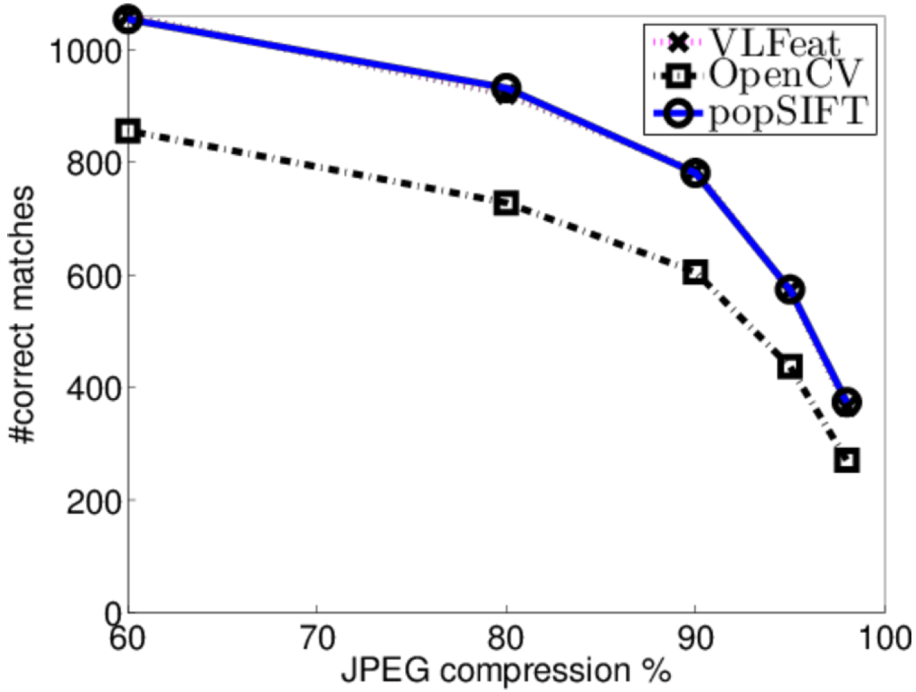
Evaluation


VLFeat
OpenCV
popSIFT
#correct matches
scale change
0
100
200
300
400
500
1
1.5
2
2.5
3
3.5
4


VLFeat
OpenCV
popSIFT
#correct matches
blur (increasing)
0
100
200
300
400
500
600
2
3
4
5
6

Evaluation


VLFeat
OpenCV
popSIFT
#correct matches
viewpoint angle
0
100
200
300
400
500
600
20
30
40
50
60


VLFeat
OpenCV
popSIFT
#correct matches
JPEG compression %
0
200
400
600
800
1000
60
70
80
90
100