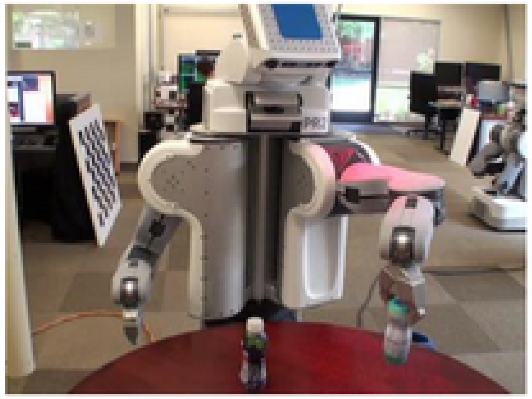
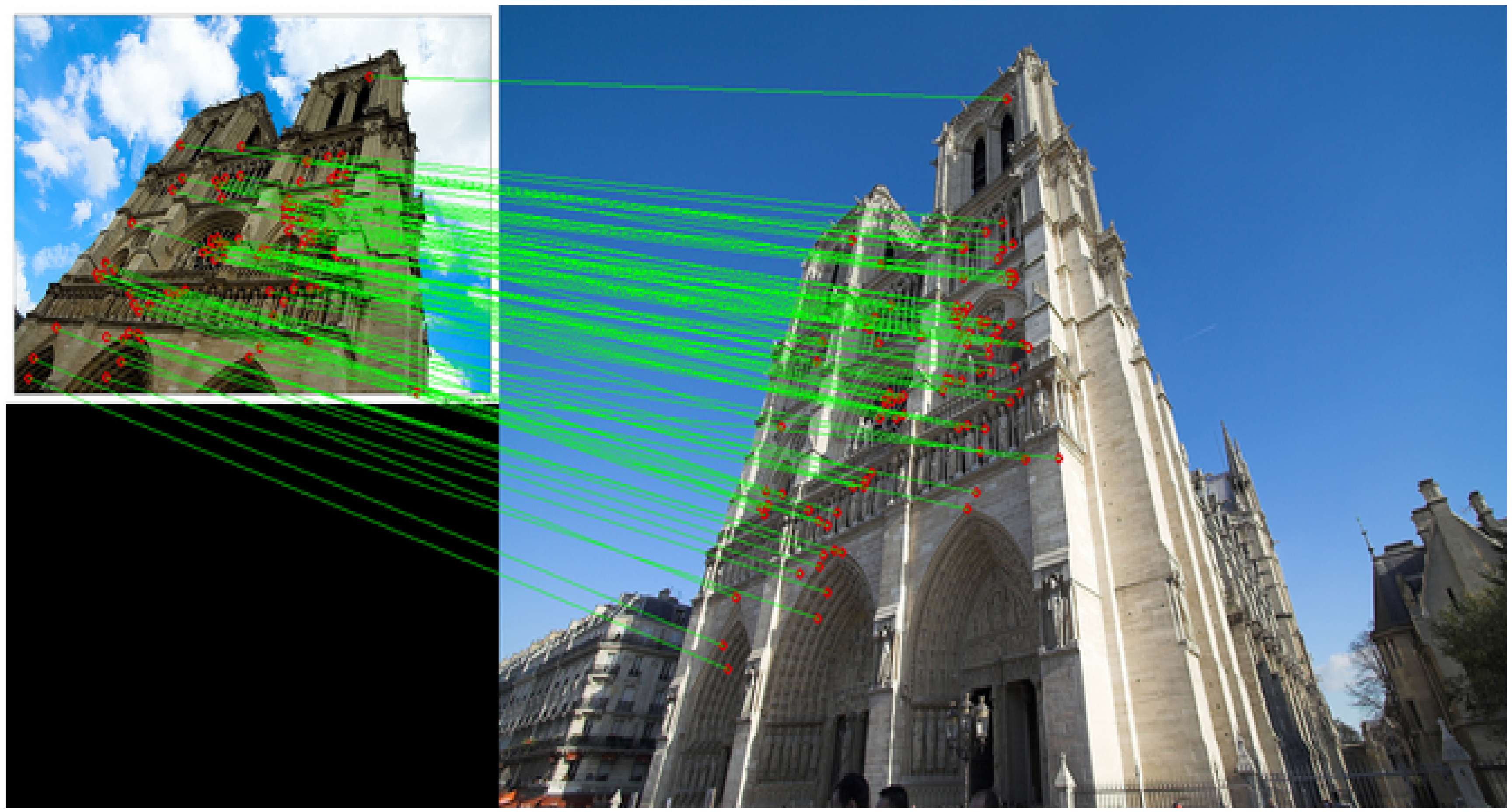


COMP 152: Probabilistic Robotics for Human-Robot Interaction

Instructor: Jivko Sinapov

www.cs.tufts.edu/~jsinapov

Image Features and Optical Flow



Research Article Presentation

- Sign-up for research article presentation

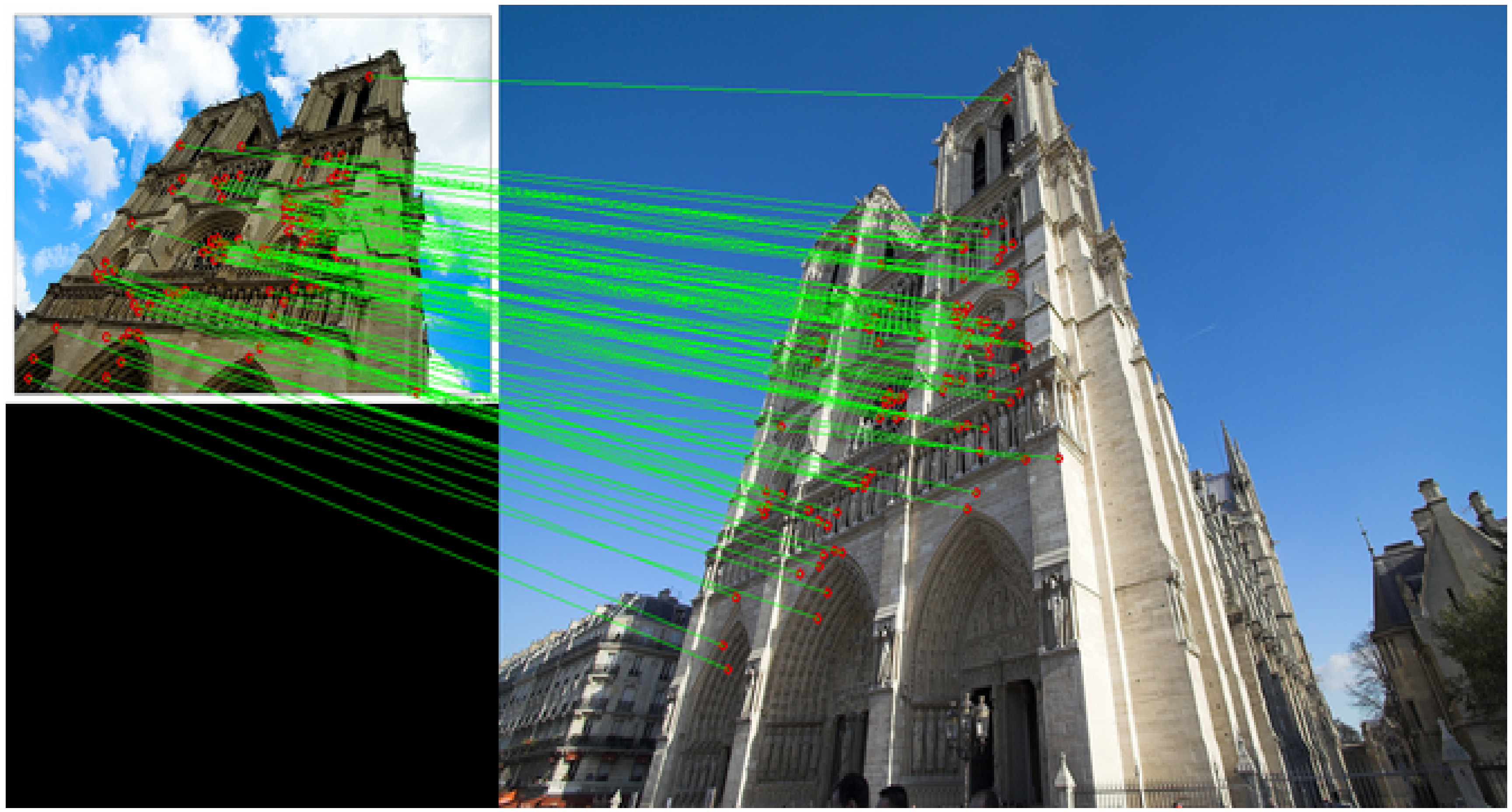
Reading Assignment

Khandelwal, P., Zhang, S., Sinapov, J., Leonetti, M., Thomason, J., Yang, F., Gori, I., Svetlik, M., Khante, P., Lifschitz, V., Aggarwal, J.K., Mooney, R., and Stone, P. (2017)

BWIBots: A platform for bridging the gap between AI and Human-Robot Interaction research

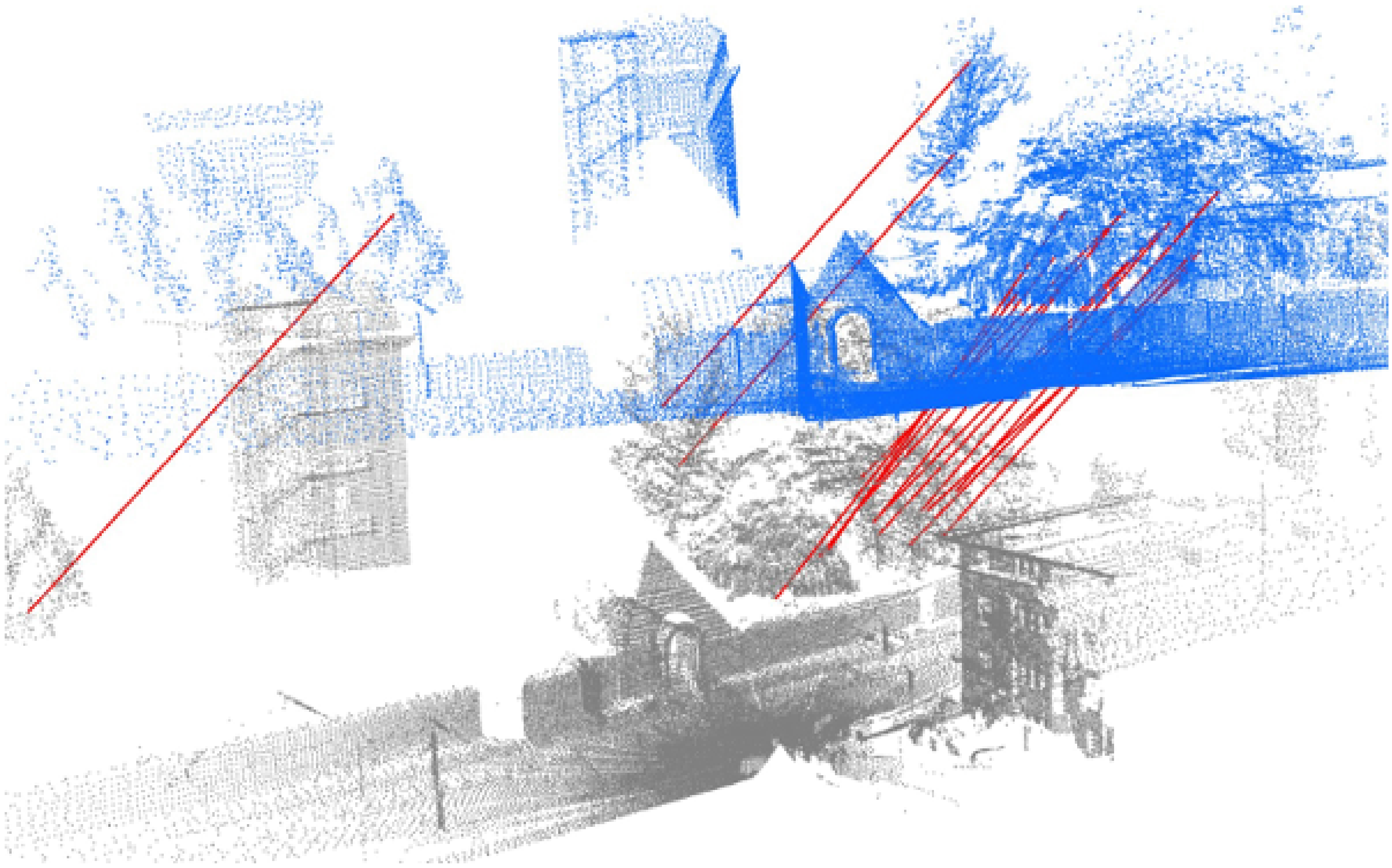
International Journal of Robotics Research, Vol. 36, No.5-7, pp. 635-659, 2017.

Visual Registration and Recognition

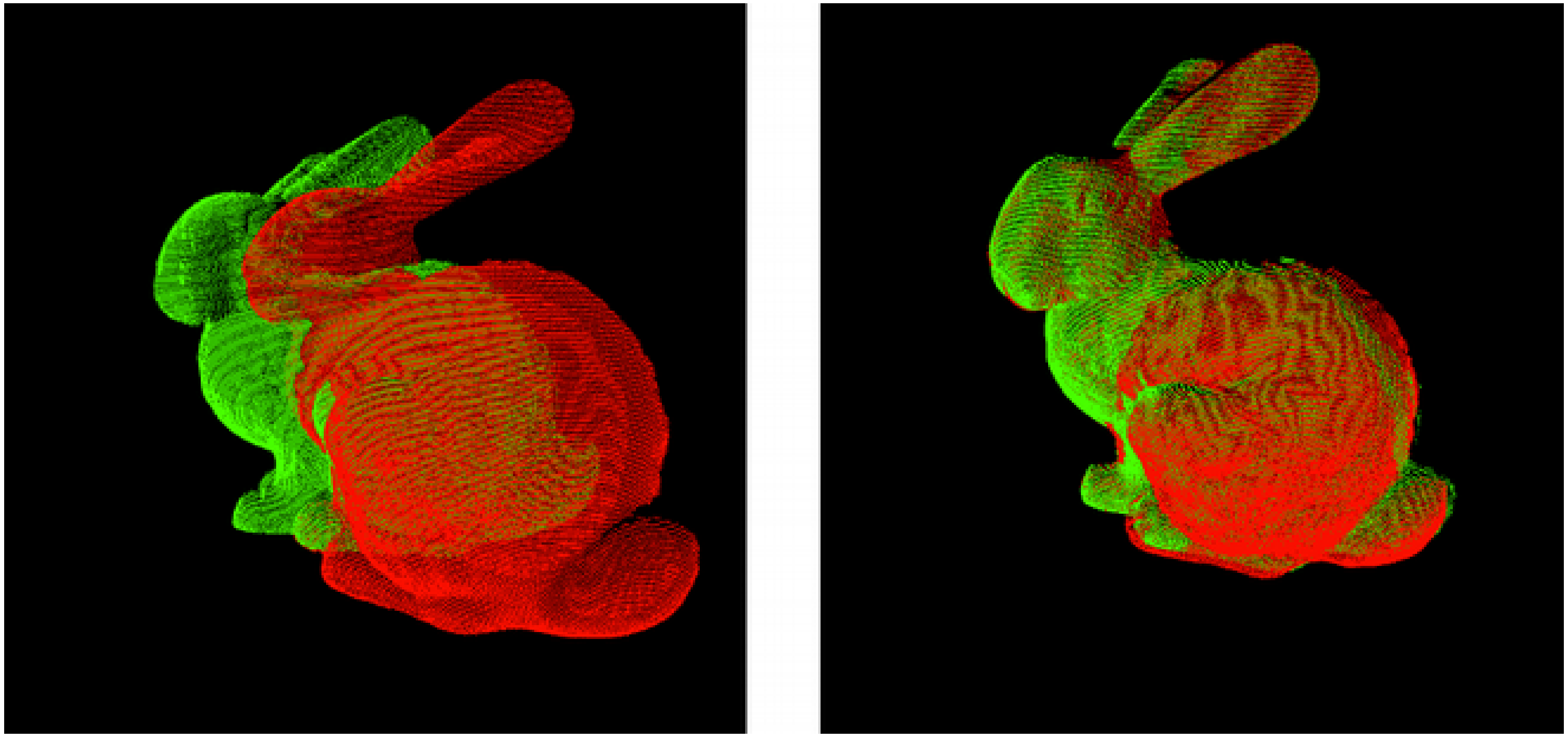


Visual Registration and Recognition

Registration

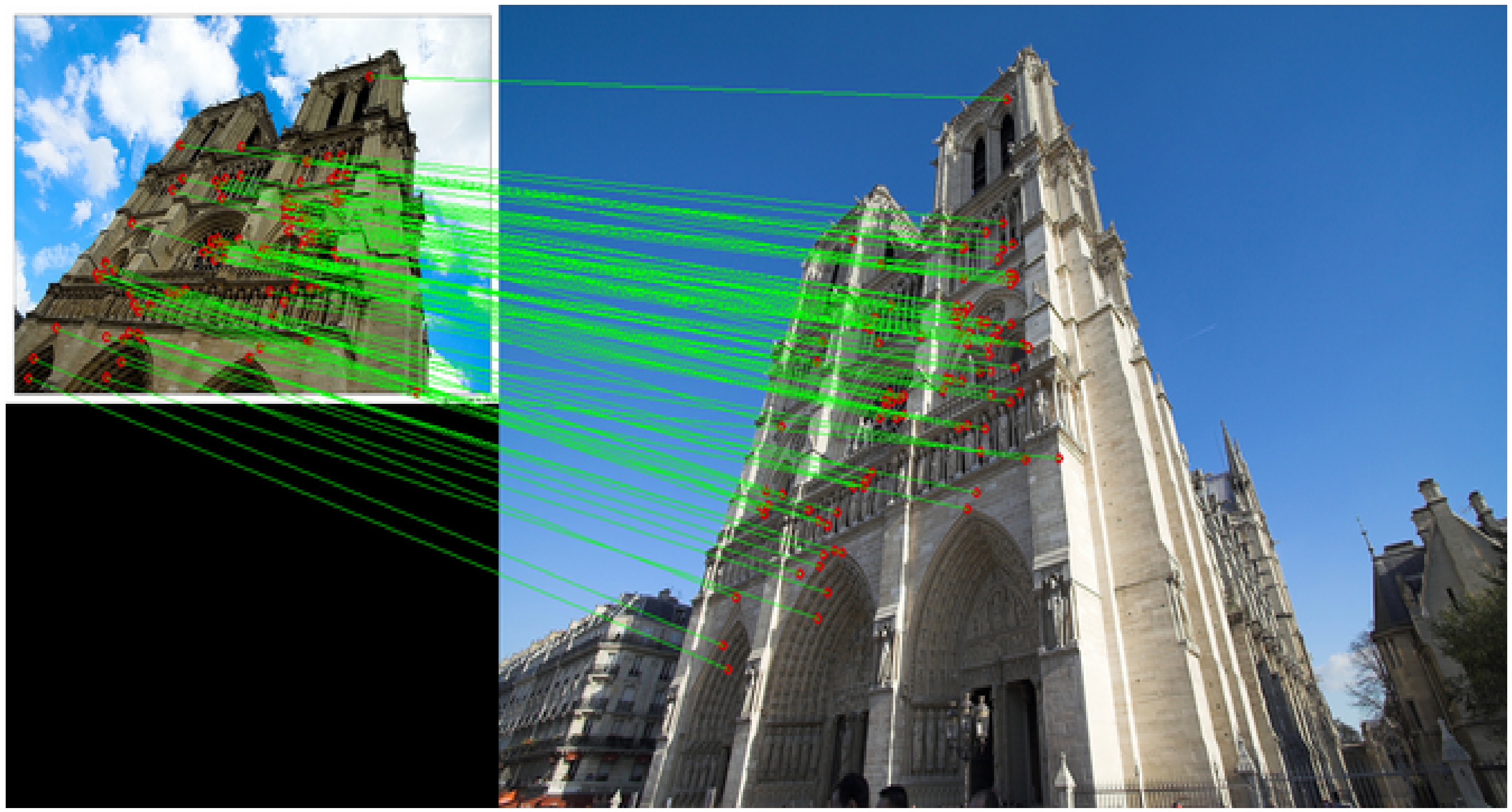


Registration

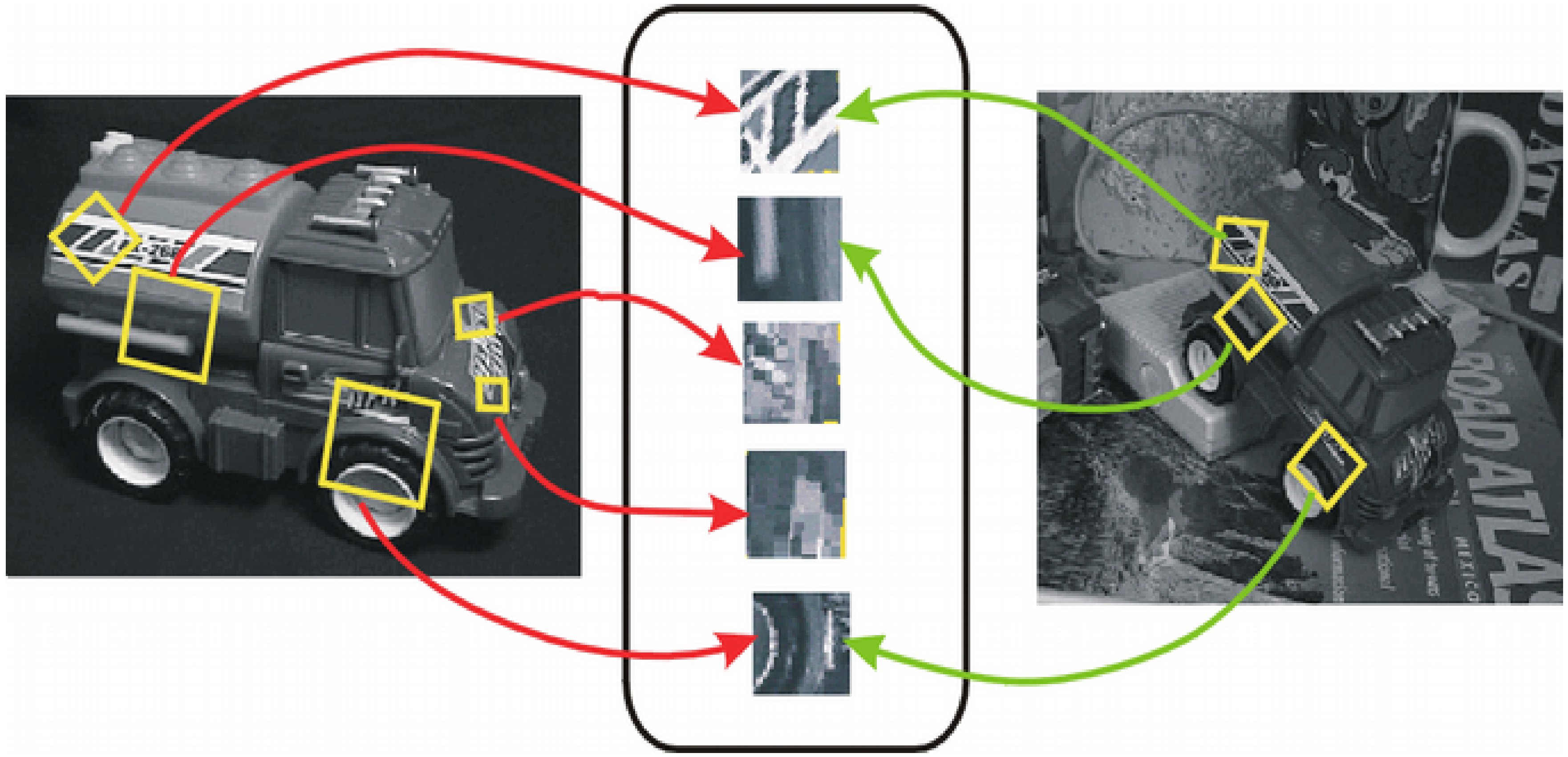


[http://vihari.github.io/personal_website/images/3dregistration.png]

Registration



Interest Point Registration

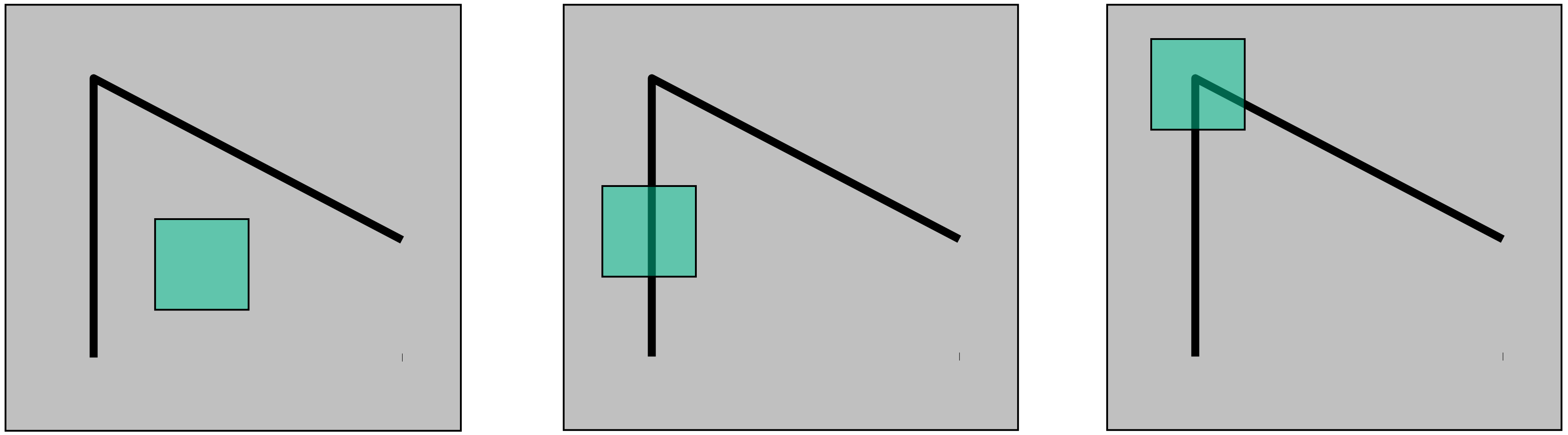


Interest Point Detection

- Look for image regions that are unusual
  - Leads to unambiguous matches in other images
  - How do we define unusual?

Suppose we only consider a small window of pixels

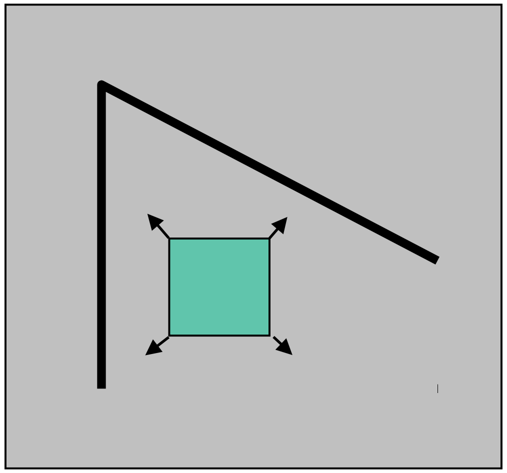
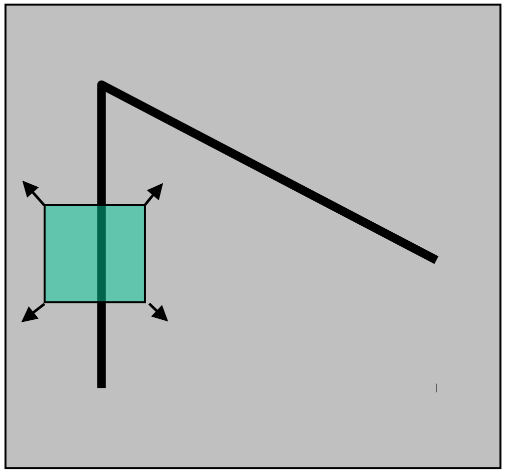
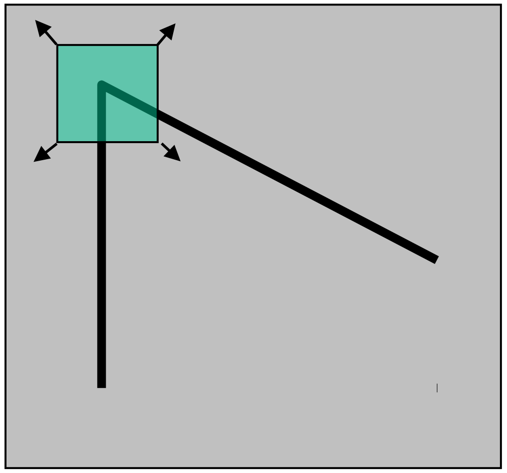
- What defines whether a feature is a good or bad candidate?



Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Local measure of feature uniqueness

- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*

"flat" region:
no change in all directions

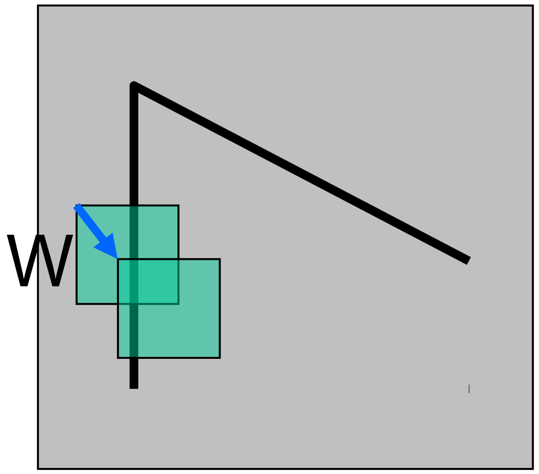
"edge":
no change along the edge direction

"corner":
significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Consider shifting the window W by (u,v)

- how do the pixels in **W** change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" of *E(u,v)*:



Harris Detector: Mathematics

Change of intensity for the shift $[u,v]$:

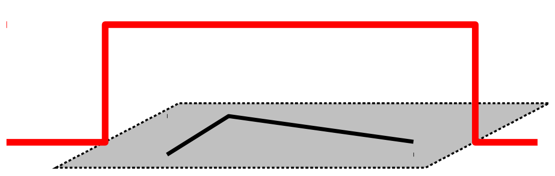
$$E(u,v) = \underset{x,y}{\text{�}} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

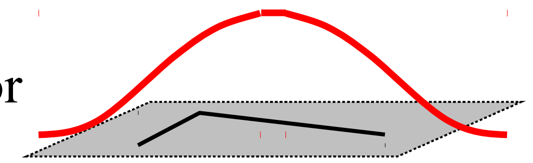
Intensity

Window function $w(x,y) =$

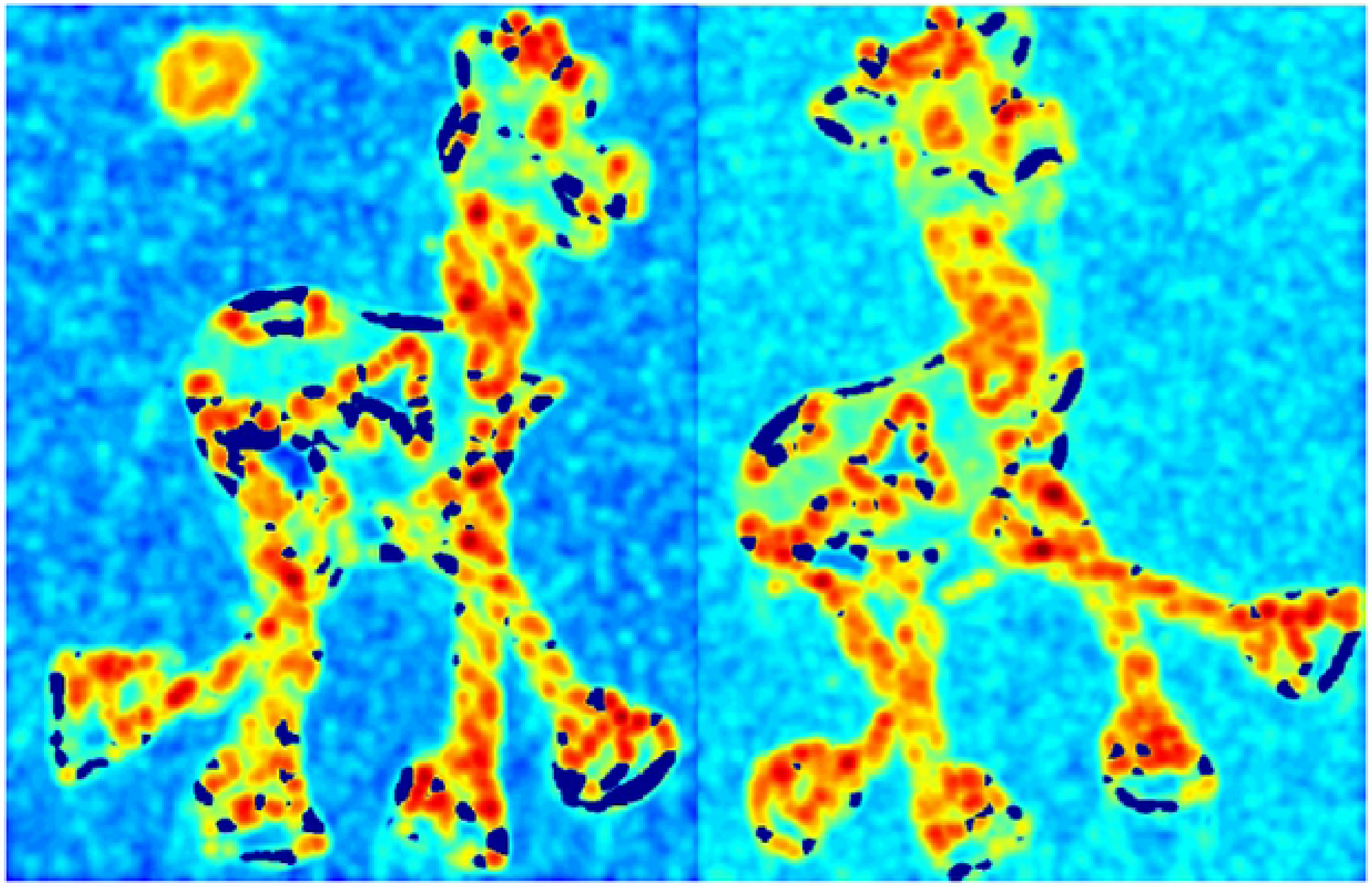


1 in window, 0 outside

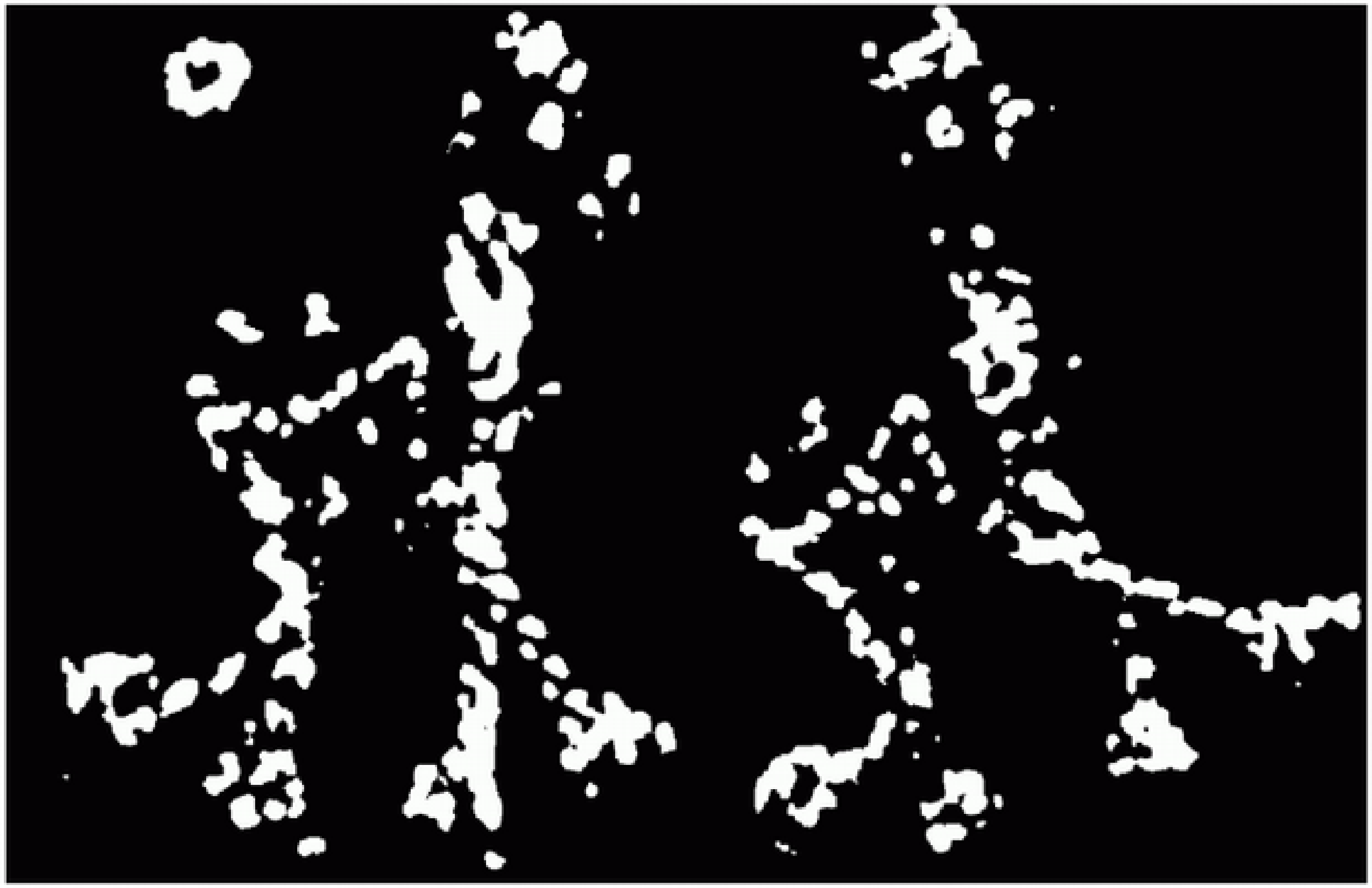
or



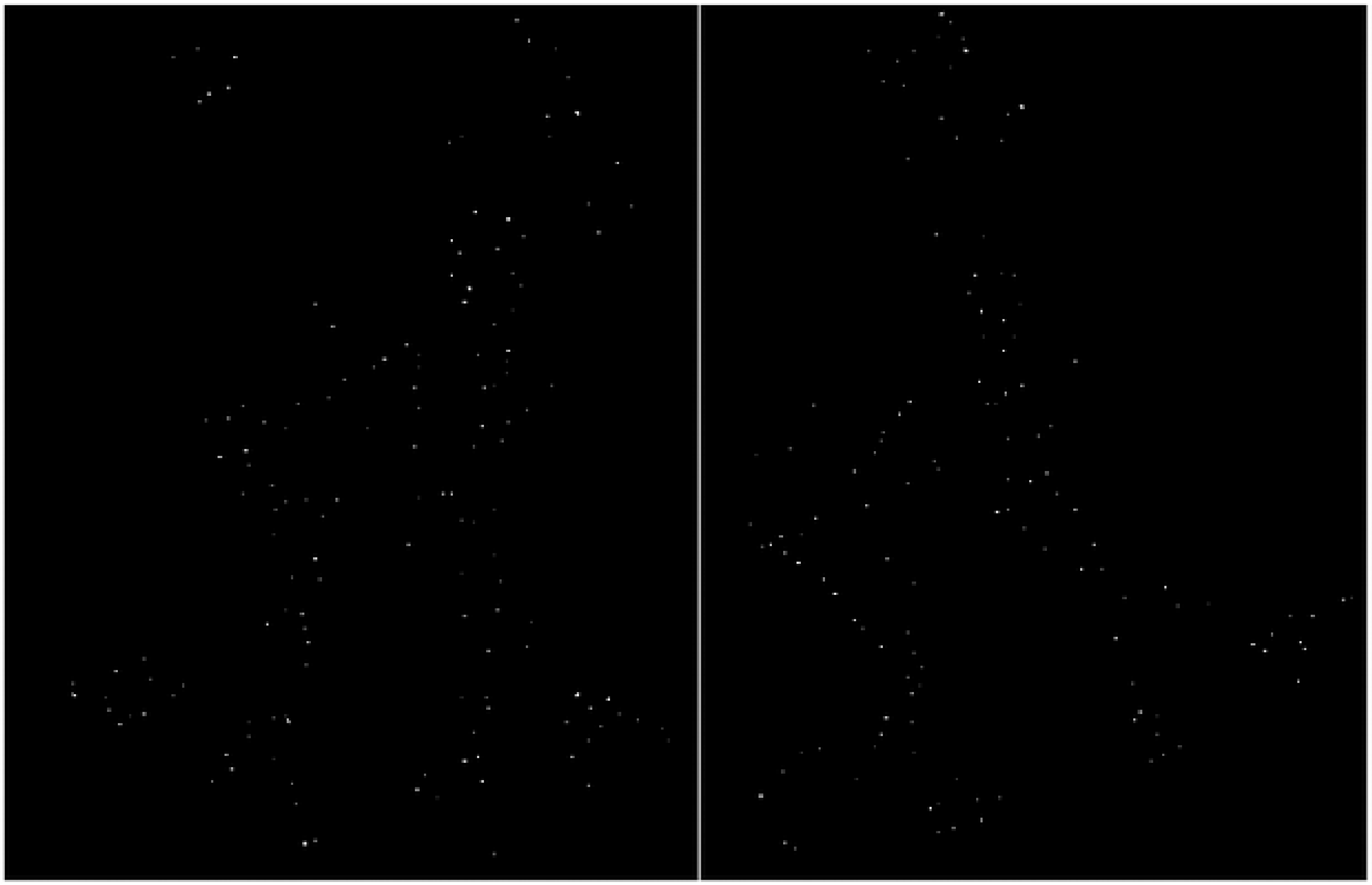
Gaussian

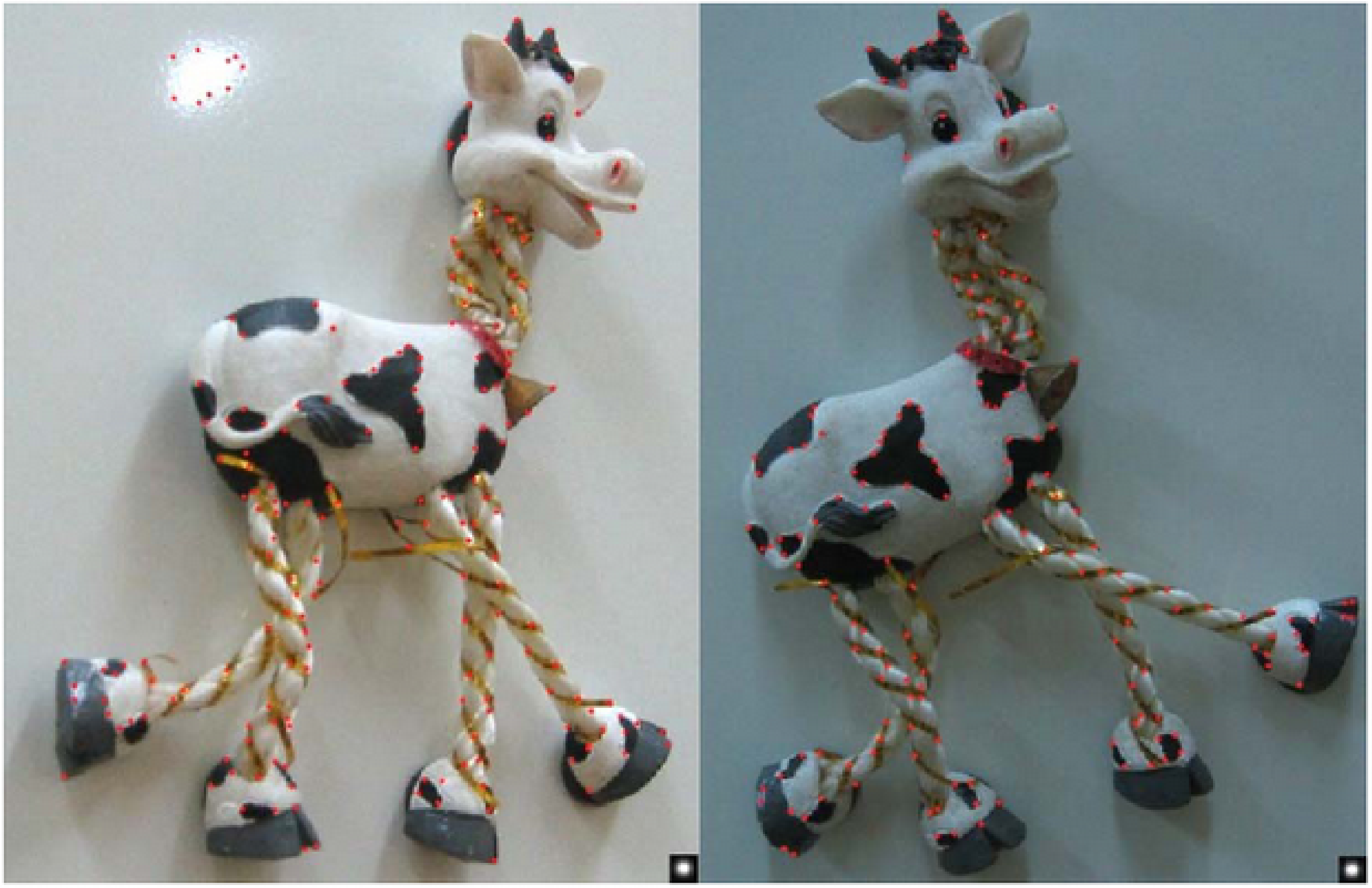


Eliminate small responses.



Find local maxima of the remaining.

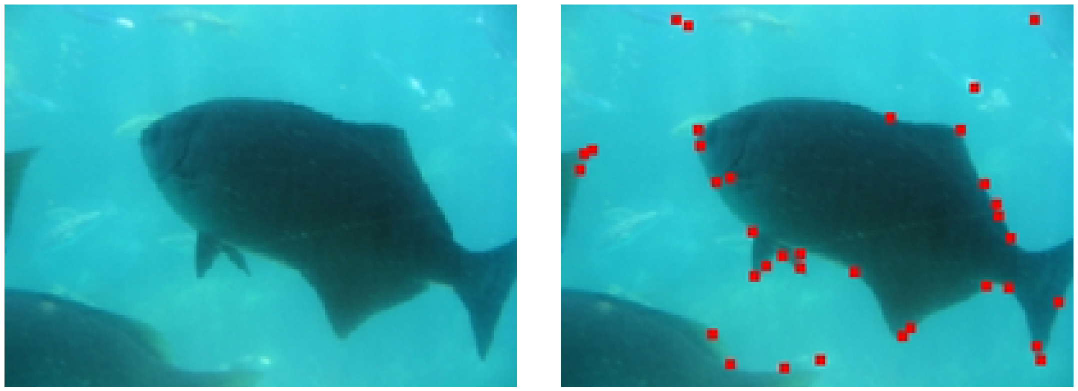




OpenCV: finding features

cv::cornerHarris(...)

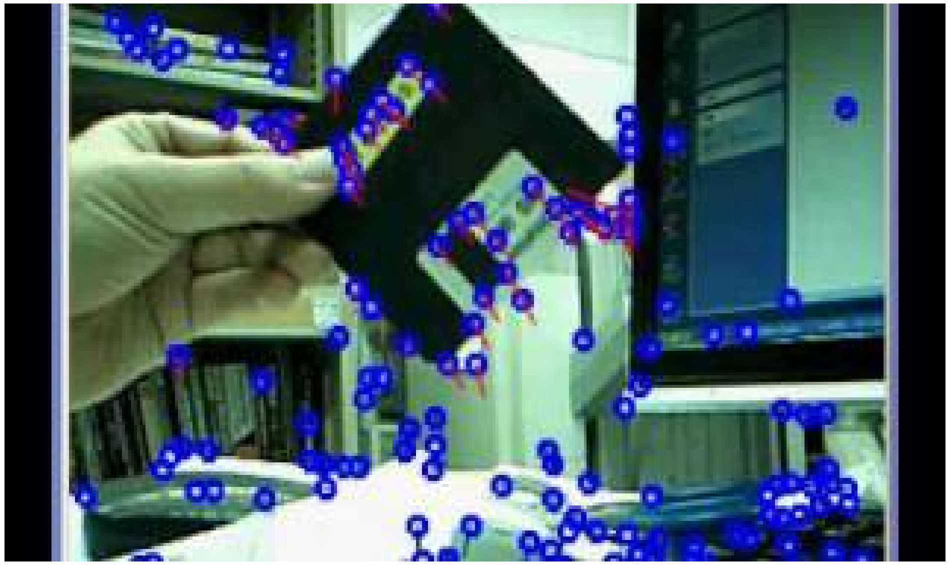
http://docs.opencv.org/2.4/doc/tutorials/features2d/trackingmotion/harris_detector/harris_detector.html

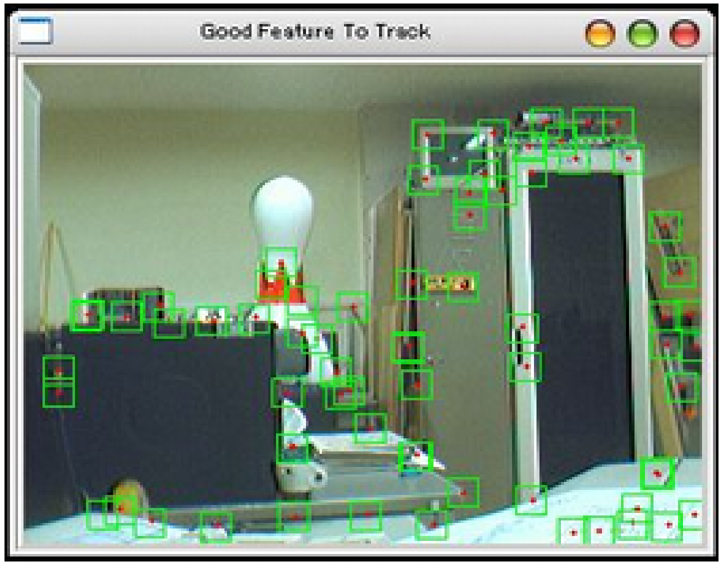


OpenCV: finding features

Shi and Tomasi '94: cv::goodFeaturesToTrack(...)

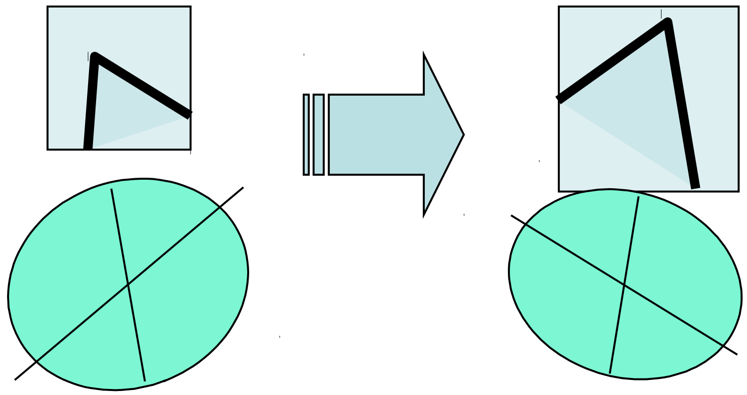
http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html





Harris Detector: Some Properties

- Rotation invariance

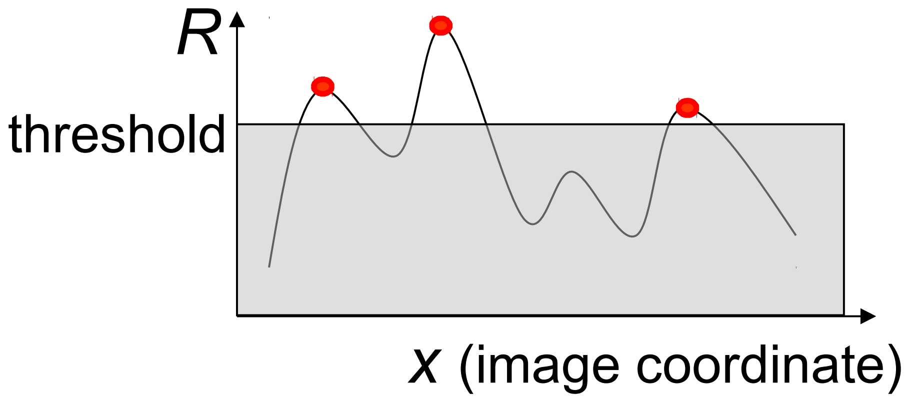


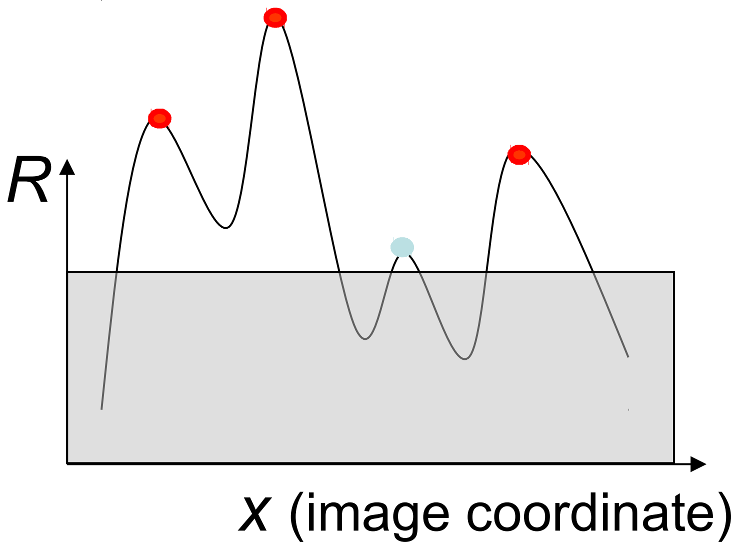
Ellipse rotates but its shape remains the same

Corner response R is invariant to image rotation

Harris Detector: Some Properties

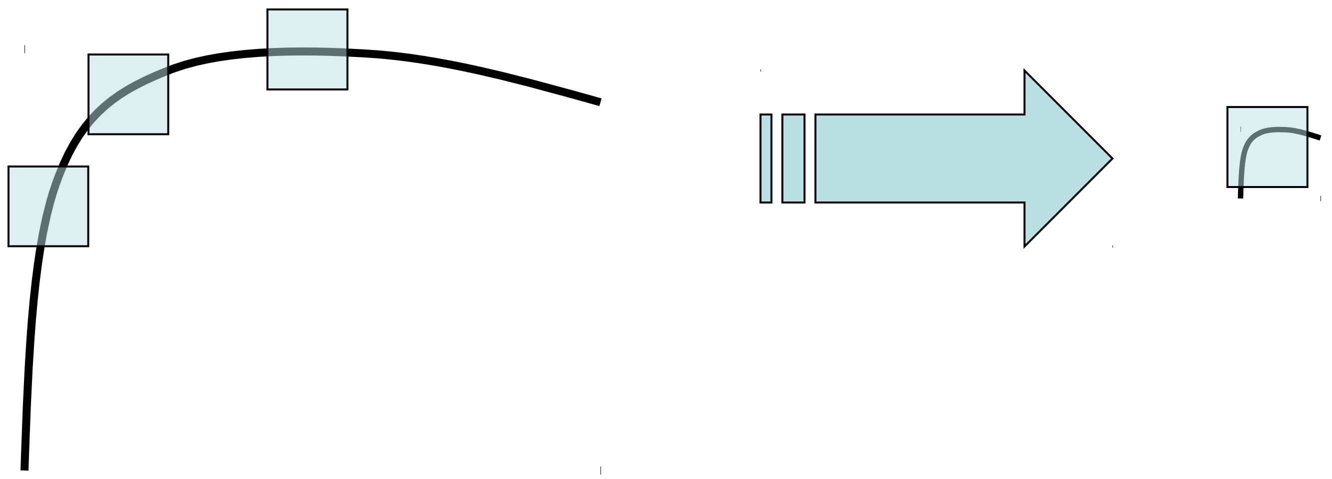
- Partial invariance to *affine intensity* change
  - ✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
  - ✓ Intensity scale: $I \rightarrow a\,I$





Harris Detector: Some Properties

- But: non-invariant to *image scale*!

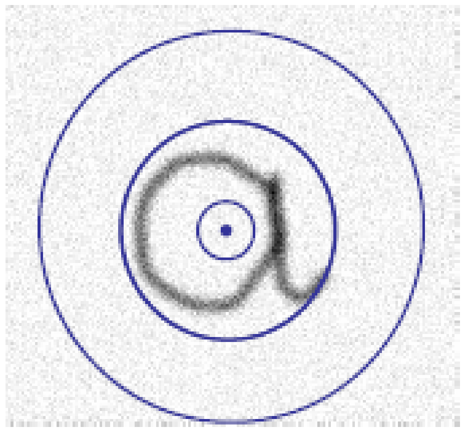


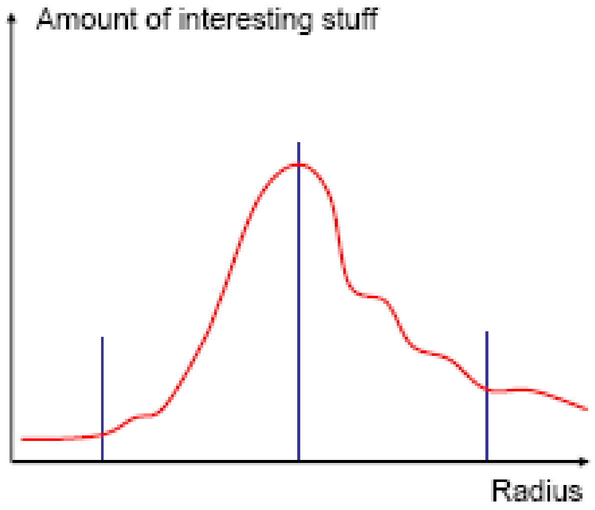
All points will be classified as edges

Corner !

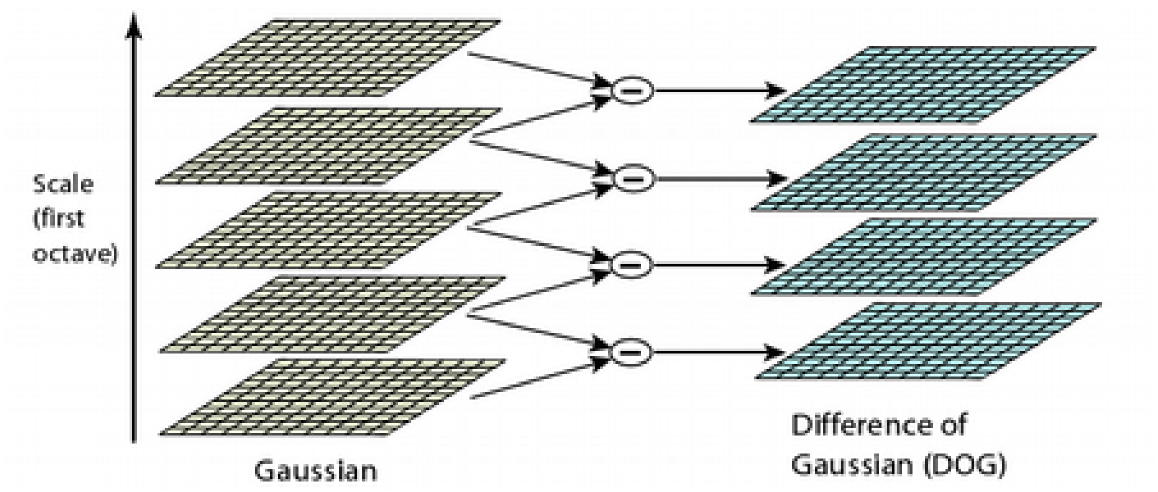
Achieving Scale Invariance

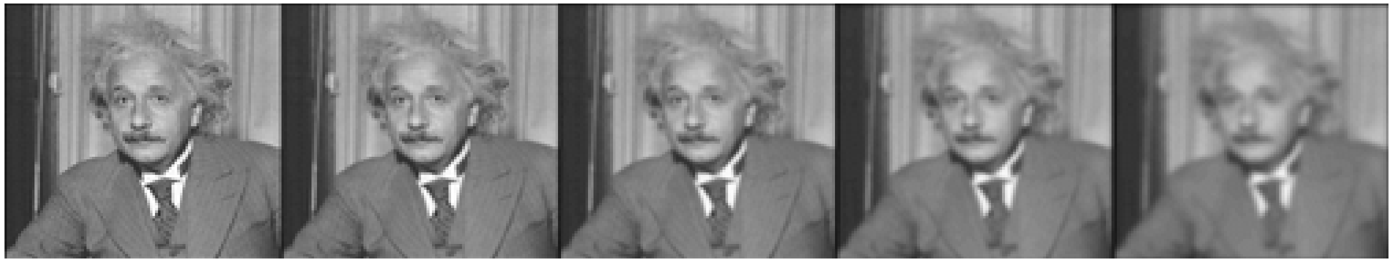
- How do we choose scale?

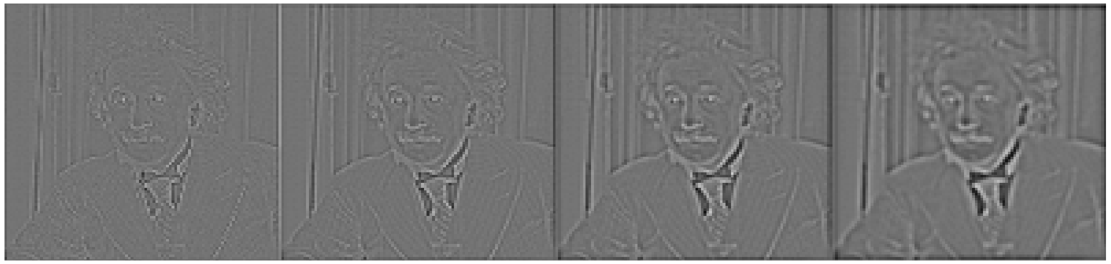




Difference-of-Gaussians



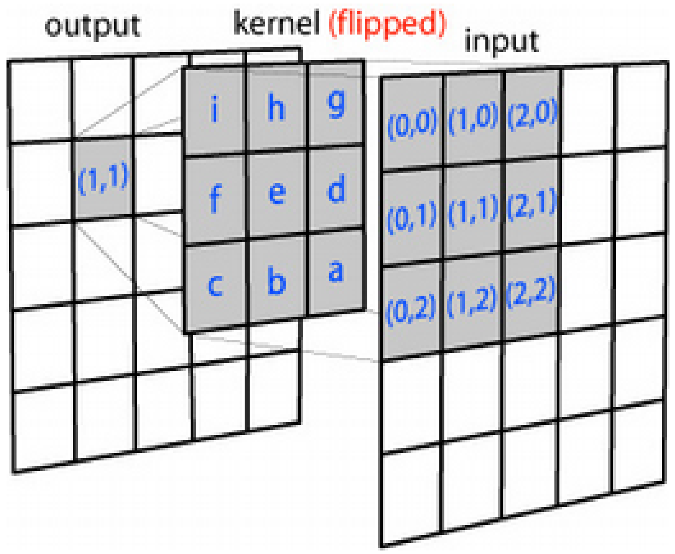




Gaussian Blur

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3x3 Gaussian Kernel



Computation of the Output Image

Gaussian Blur Kernels

1/16

1	2	1
2	4	2
1	2	1

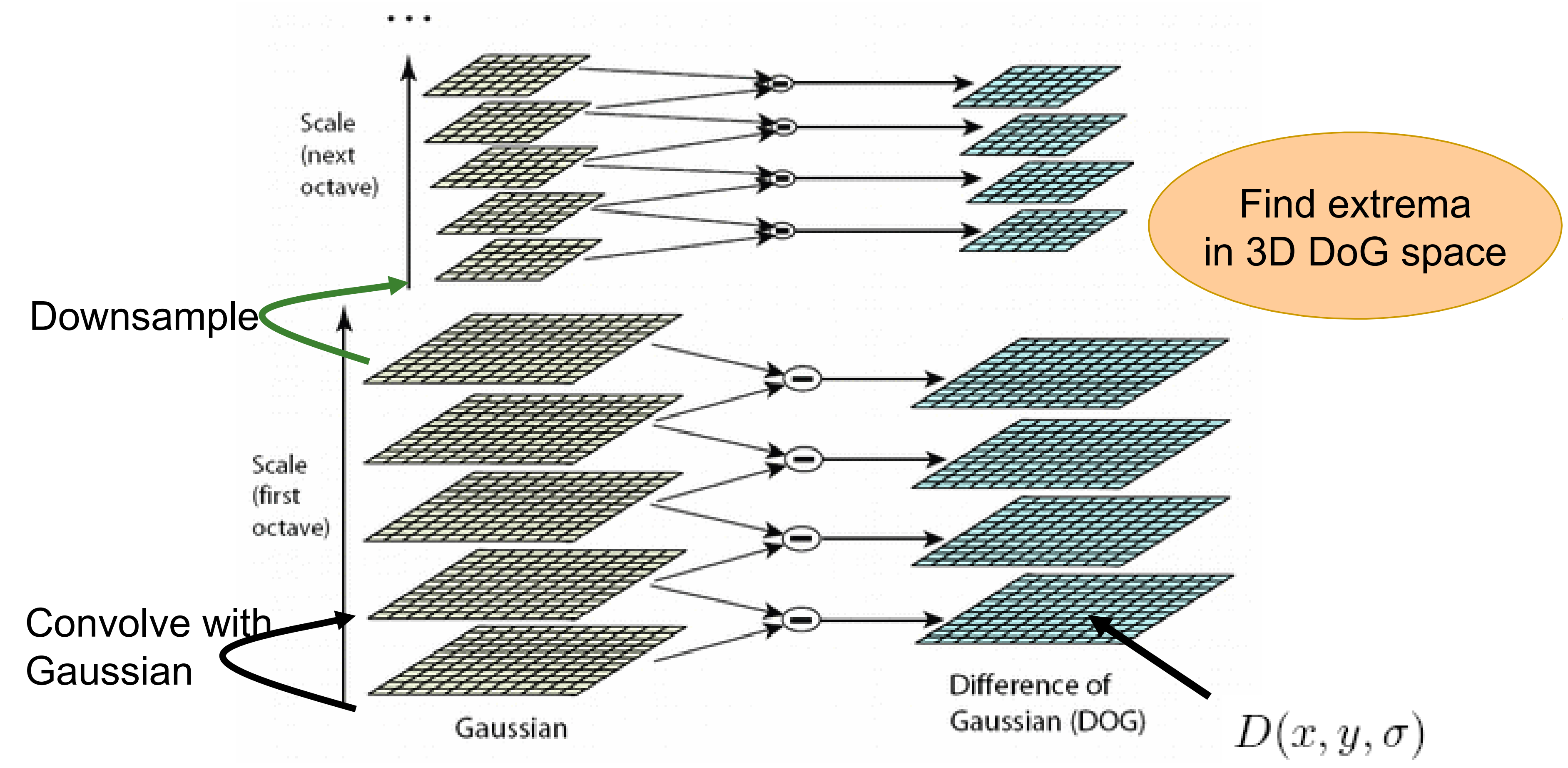
1/273

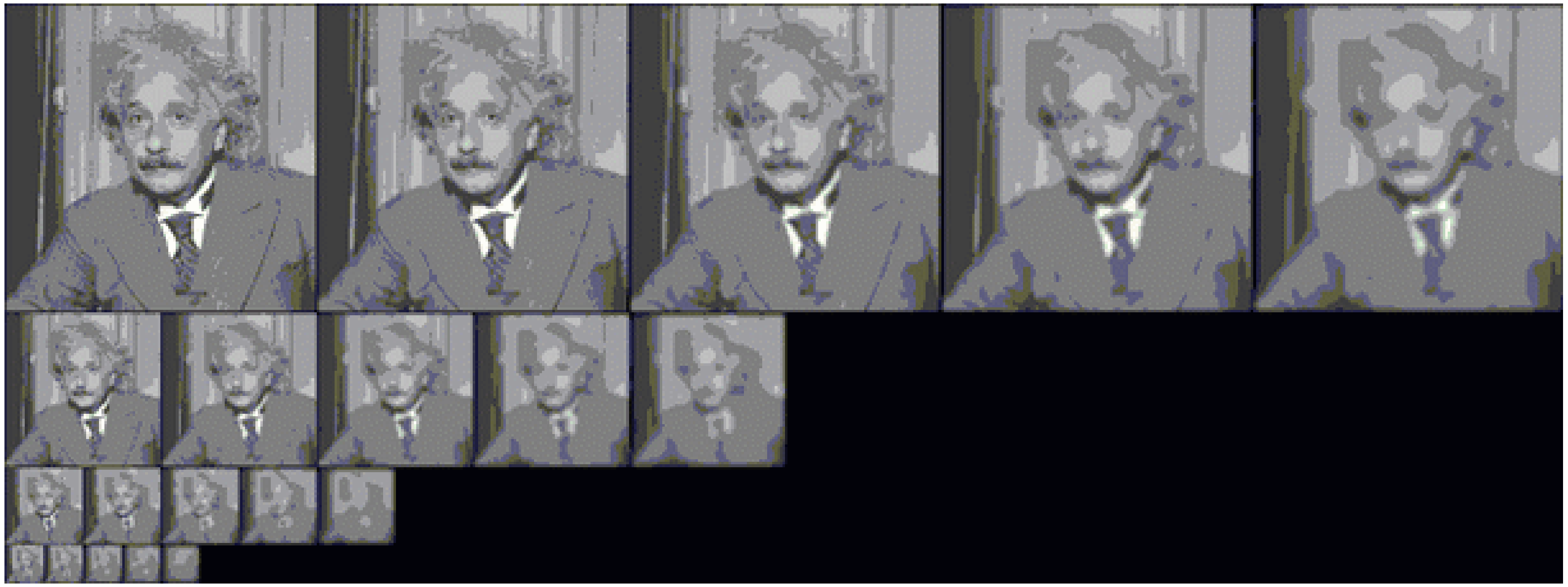
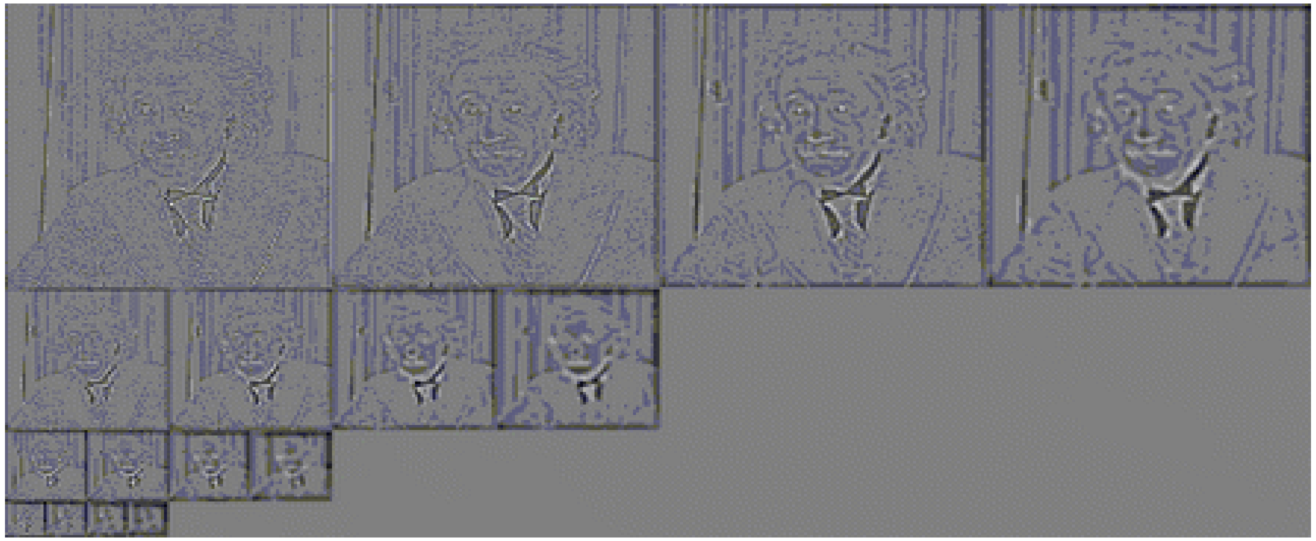
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

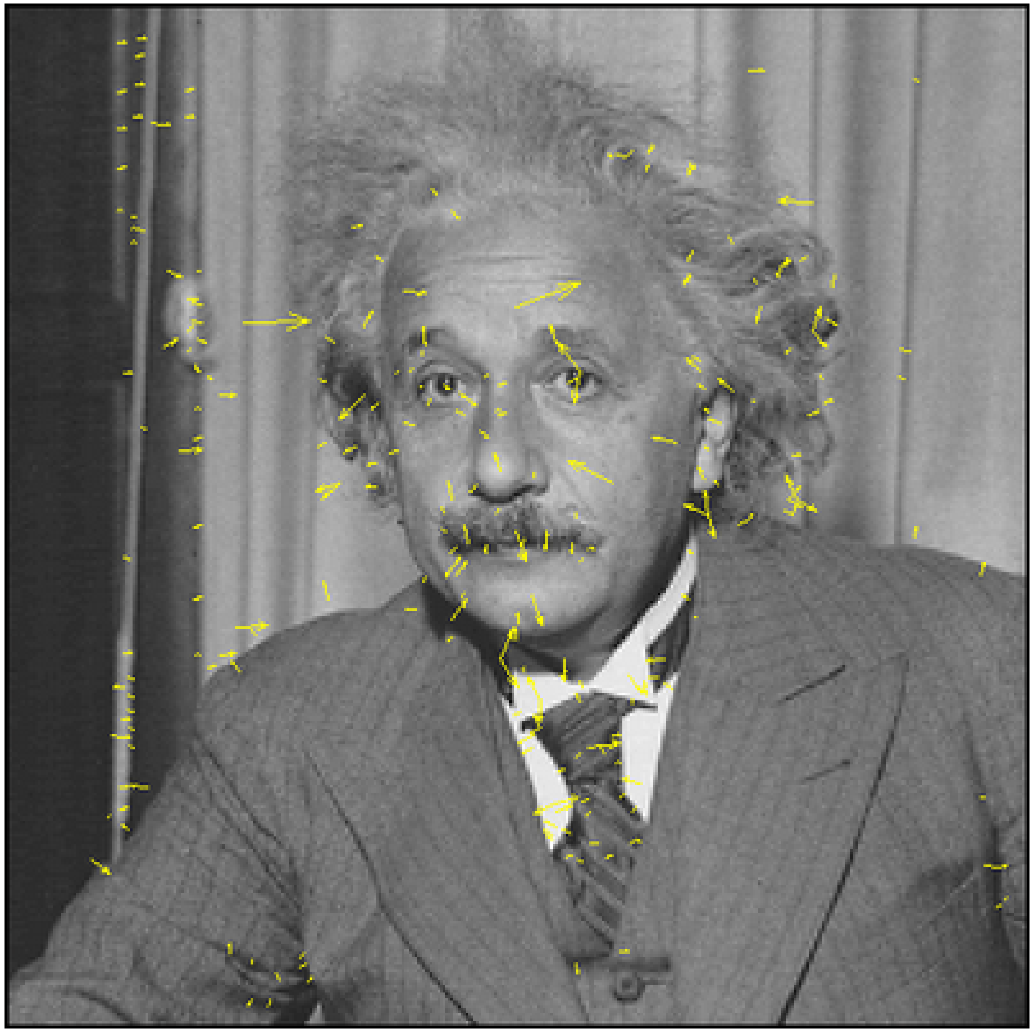
0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

Finding Keypoints – Scale, Location



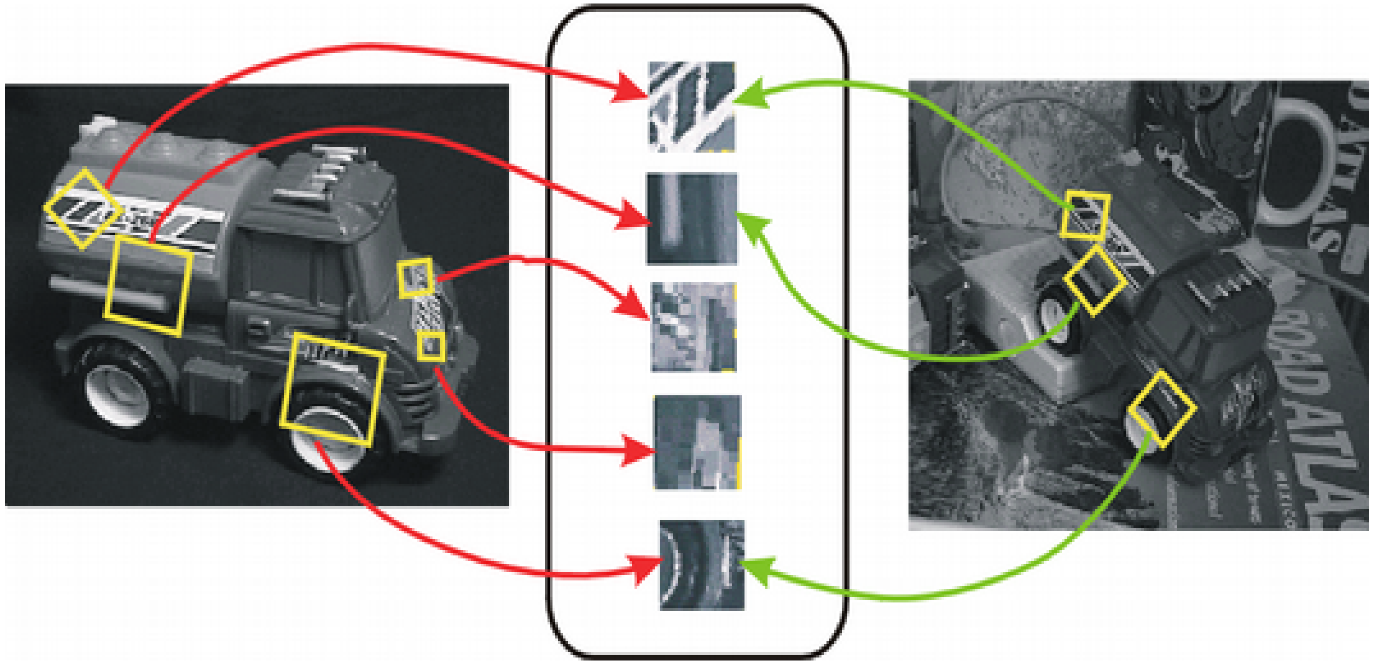



SIFT descriptor



Interest Point Descriptors

- Now that we can find interest points, how do we compare them?



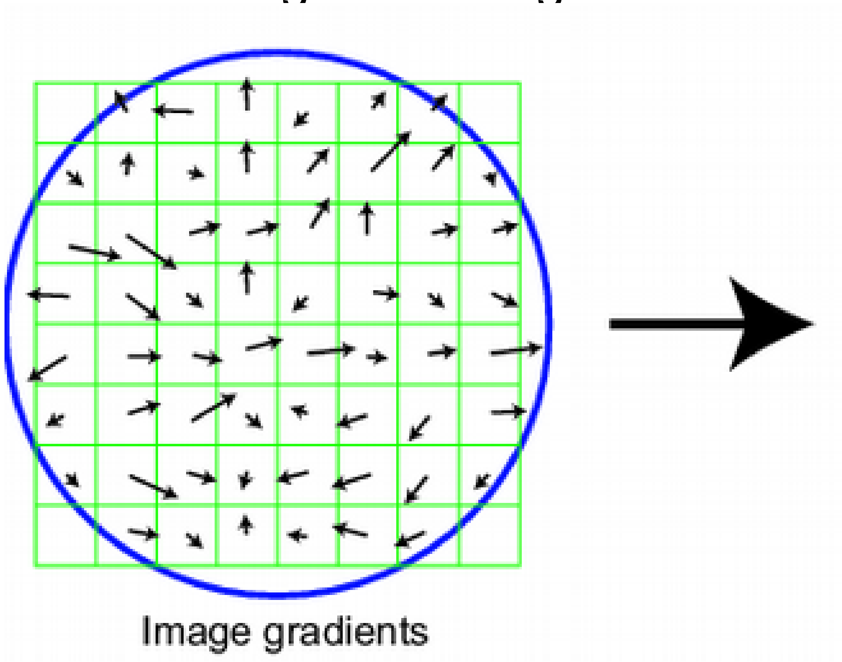
Interest Point Descriptors

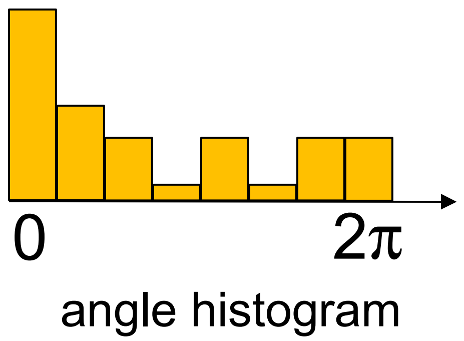
- Now that we can find interest points, how do we compare them?
- Answer: compute a numerical feature descriptor describing the orientation, and scale of the interest point

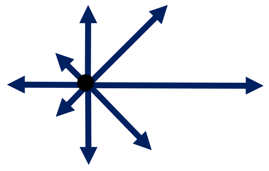
SIFT descriptor

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Create histogram of edge orientations





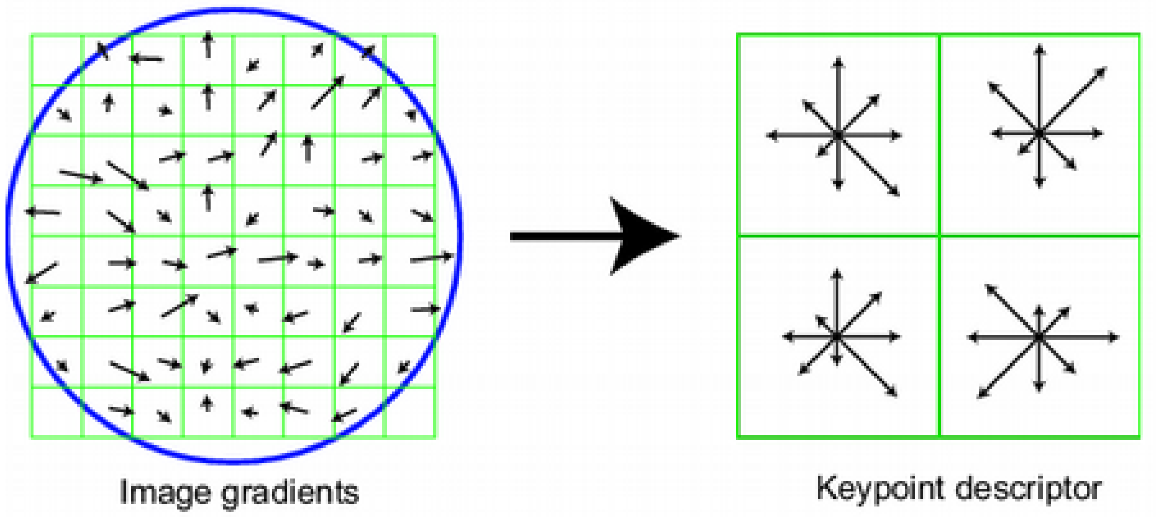


Adapted from slide by David Lowe

SIFT descriptor

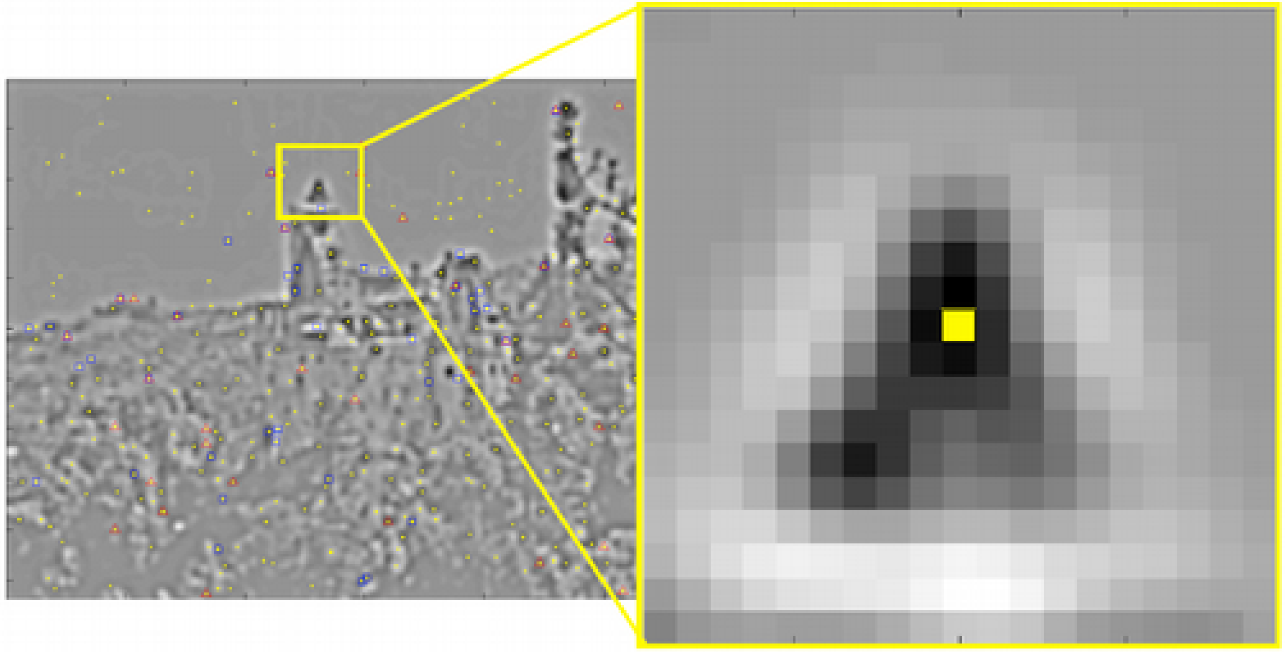
Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



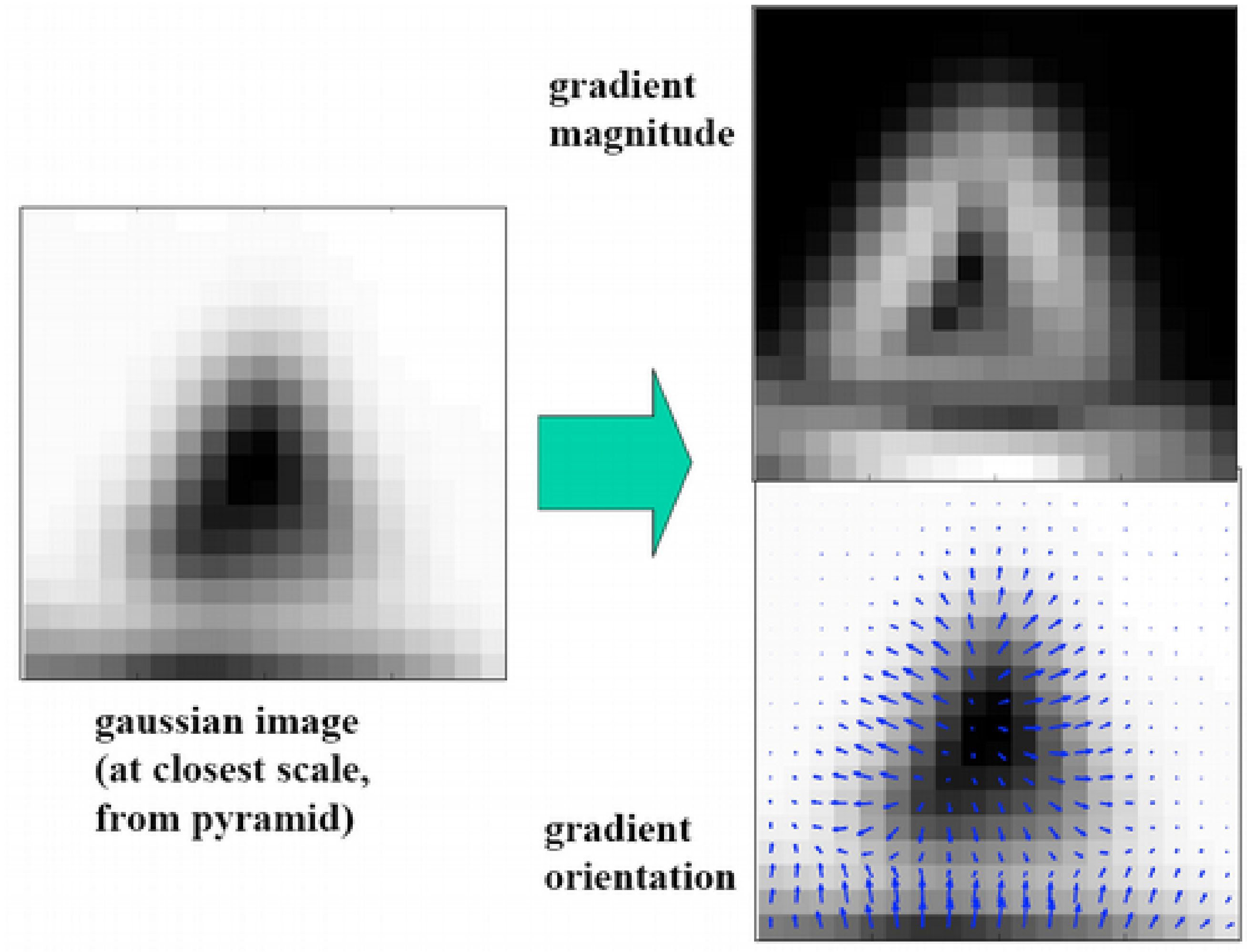
Adapted from slide by David Lowe

SIFT descriptor

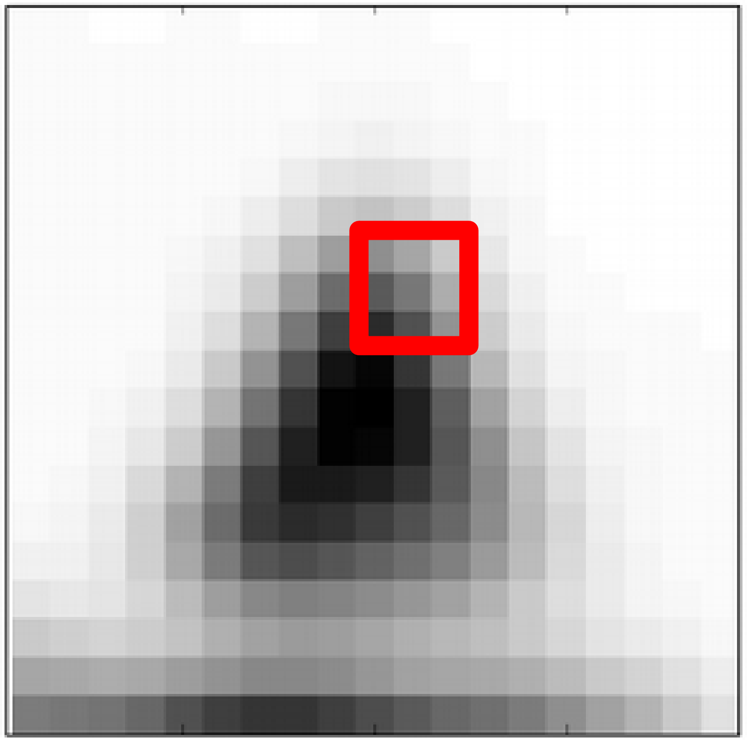


- Keypoint location = extrema location
- Keypoint scale is scale of the DOG image

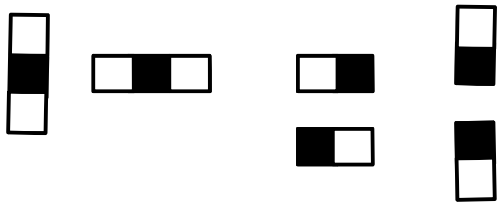
SIFT descriptor

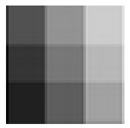


Computing Angle of Gradient



Angle and magnitude of gradient are computed using 1 and 2-side edge filters:

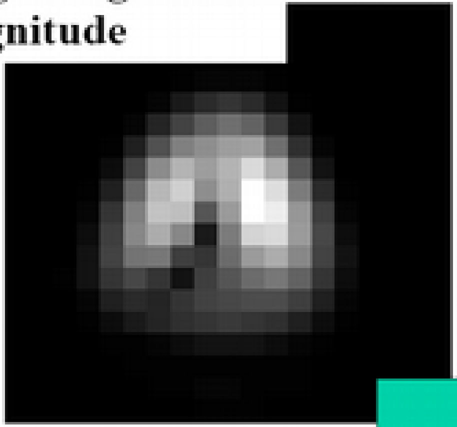




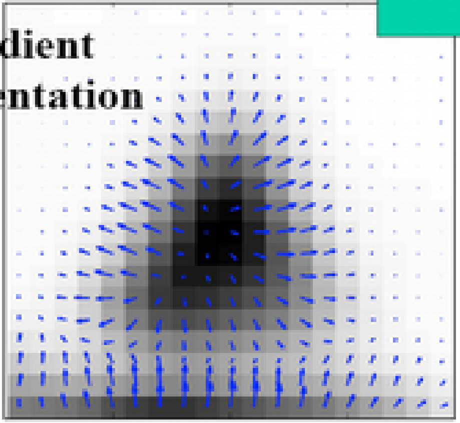
Patch

SIFT descriptor

weighted gradient magnitude

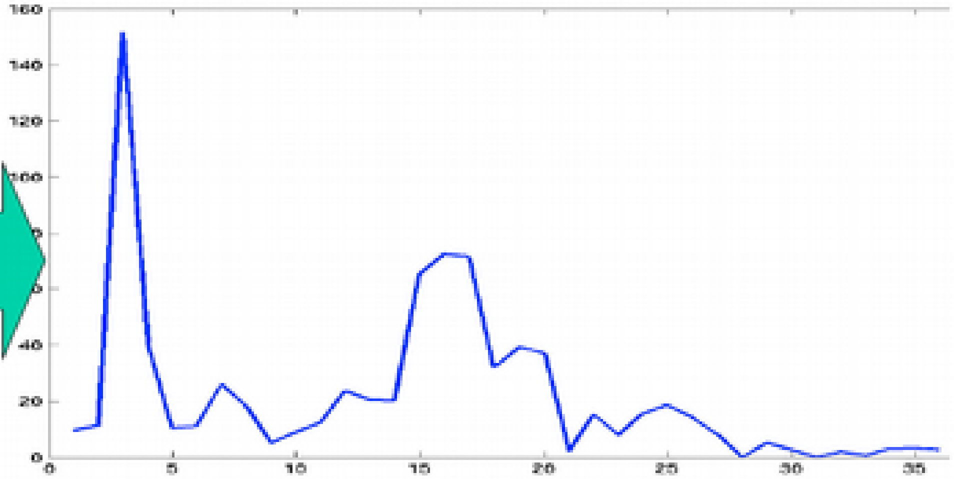


gradient orientation



weighted orientation histogram.

Each bucket contains sum of weighted gradient magnitudes corresponding to angles that fall within that bucket.



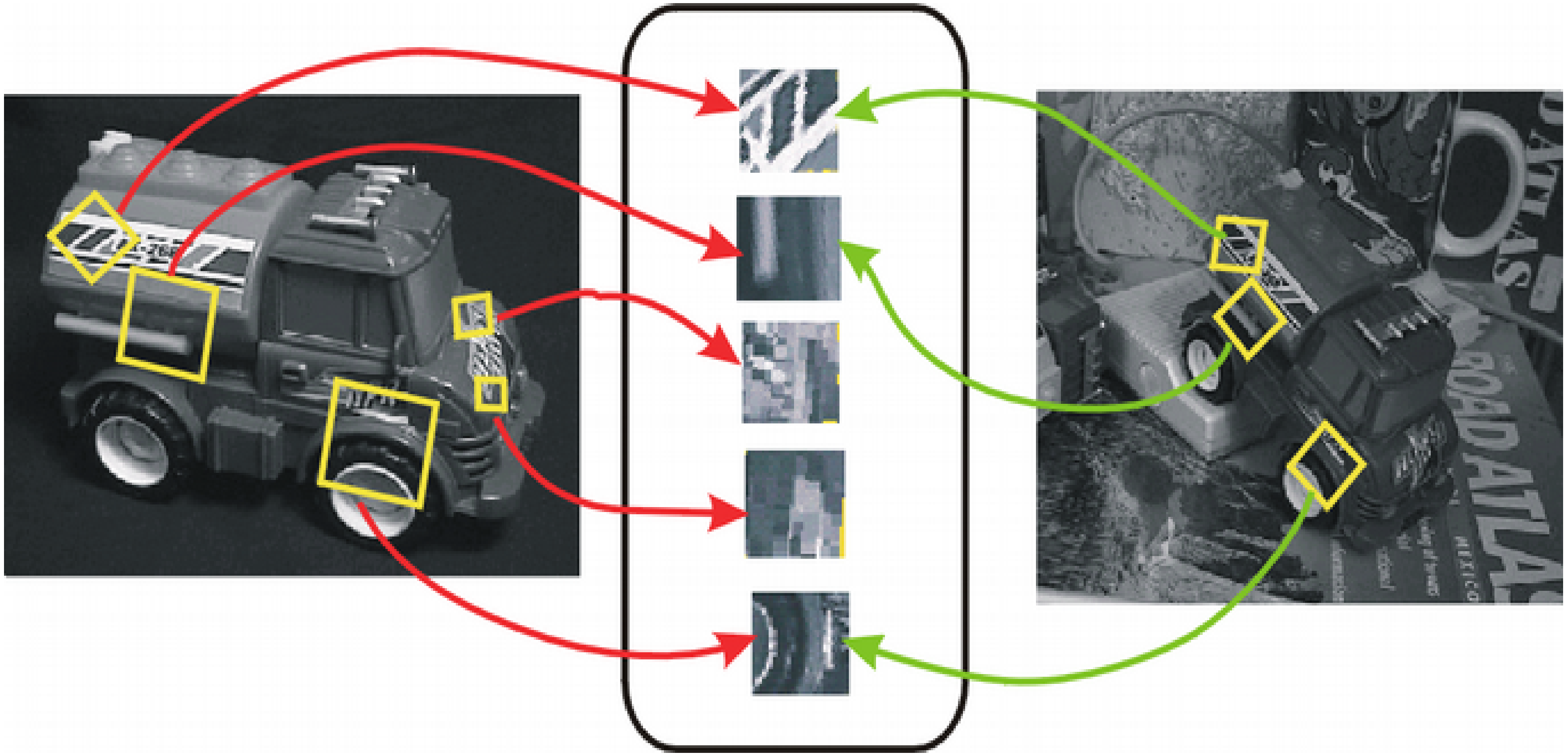
36 buckets

10 degree range of angles in each bucket, i.e.

0 <=ang<10 : bucket 1

10<=ang<20 : bucket 2

20<=ang<30 : bucket 3 …

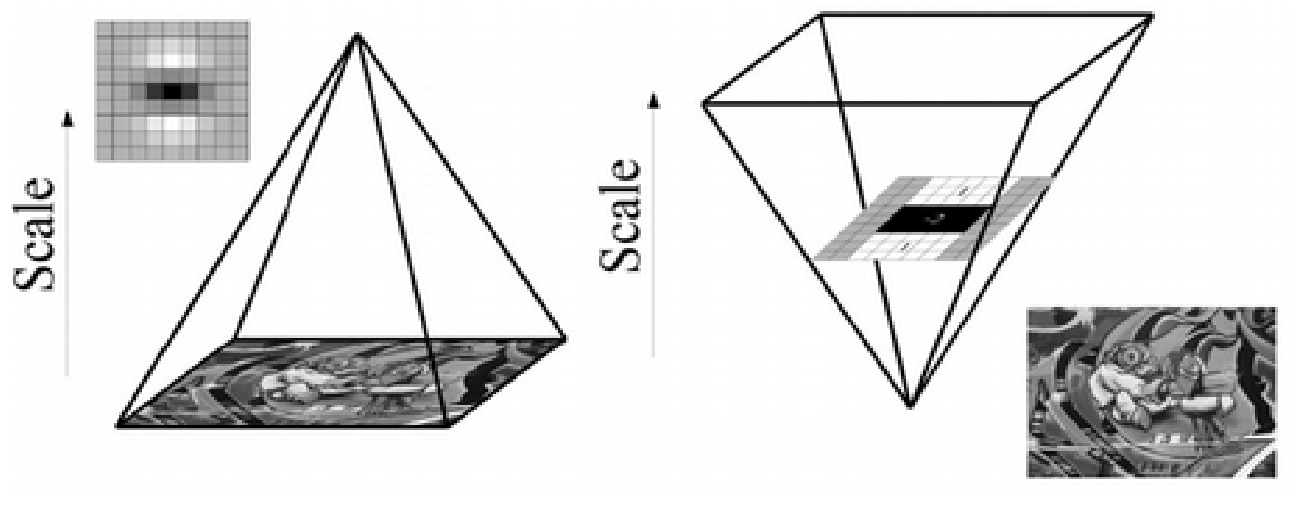


The problem with SIFT…

- Slow…
- Copyrighted!
  - Alternatives: SURF
  - OpenSURF:
    - http://opensurf1.googlecode.com/files/OpenSURF.pdf
  - Included in OpenCV 2.0+
  - OpenCV Tutorial:
    - http://achuwilson.wordpress.com/2011/08/05/object-detection-using-surf-in-opencv-part-1/

SURF

- Achieves quicker computation by scaling the filter rather than the image:



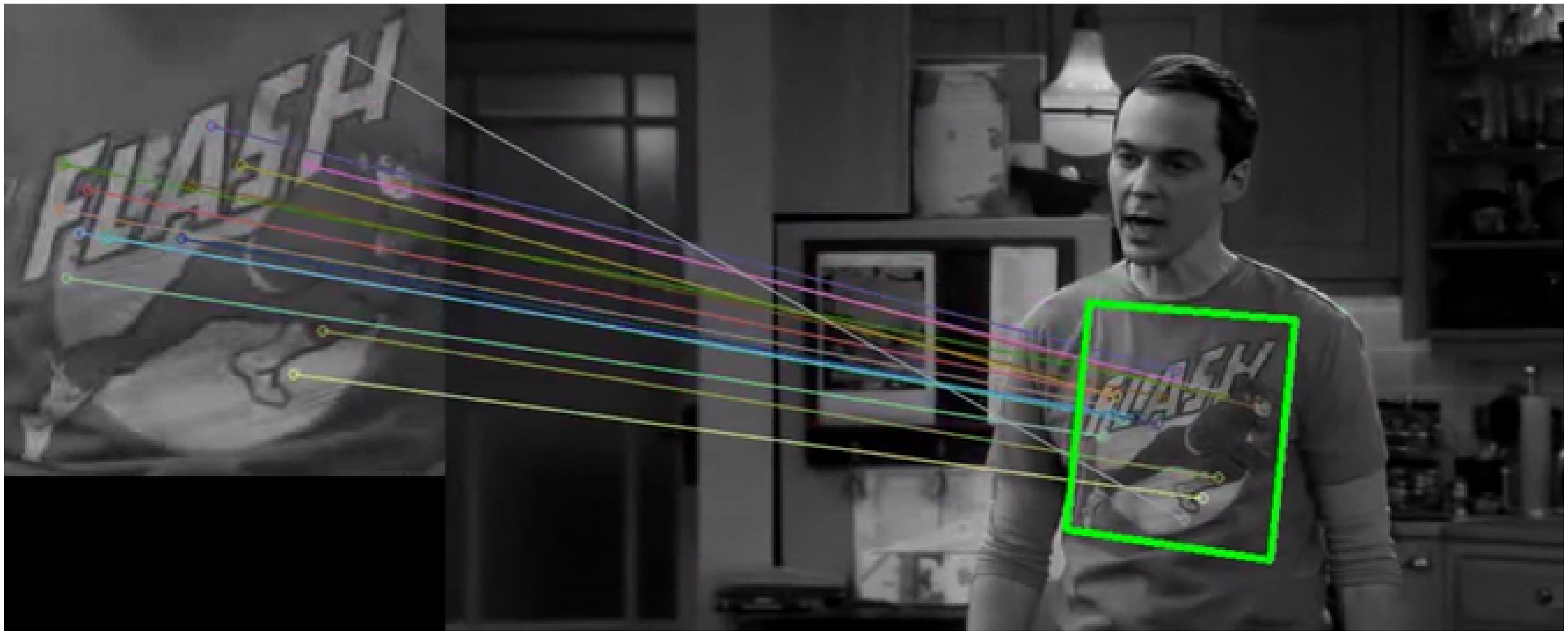
SIFT **SURF**

To summarize…

- Feature detectors:
  - Find interest points in image (e.g., using difference of Gaussians, Harris corner detection, etc.)
- Feature descriptors
  - Each detected feature can be represented by a numerical descriptor encoding orientation, scale, etc.

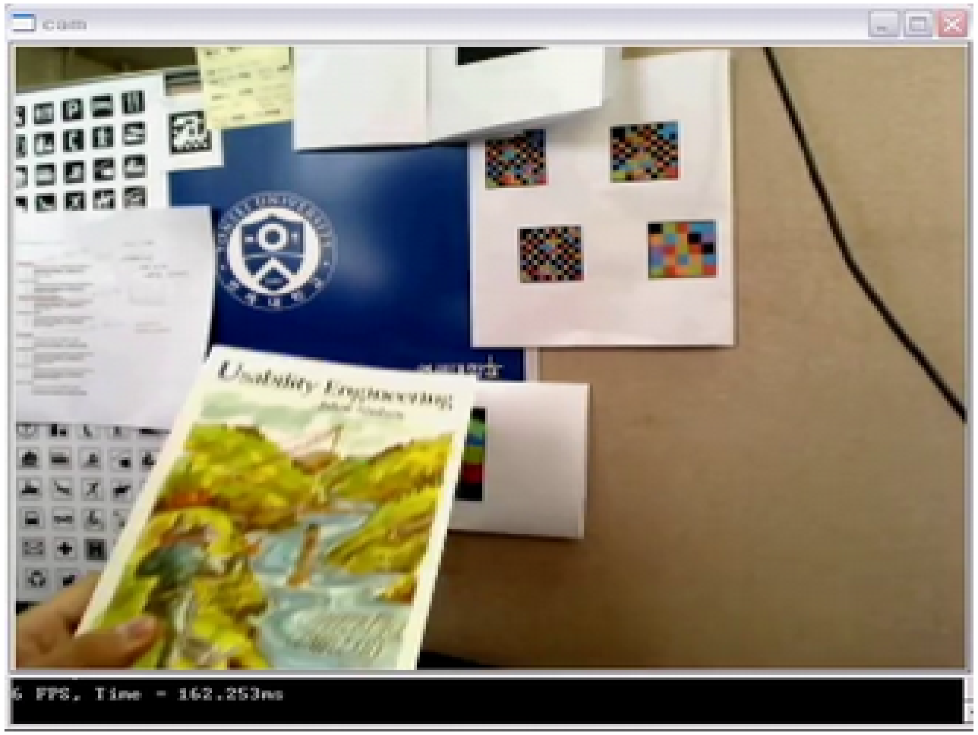
Applications

Object Detection



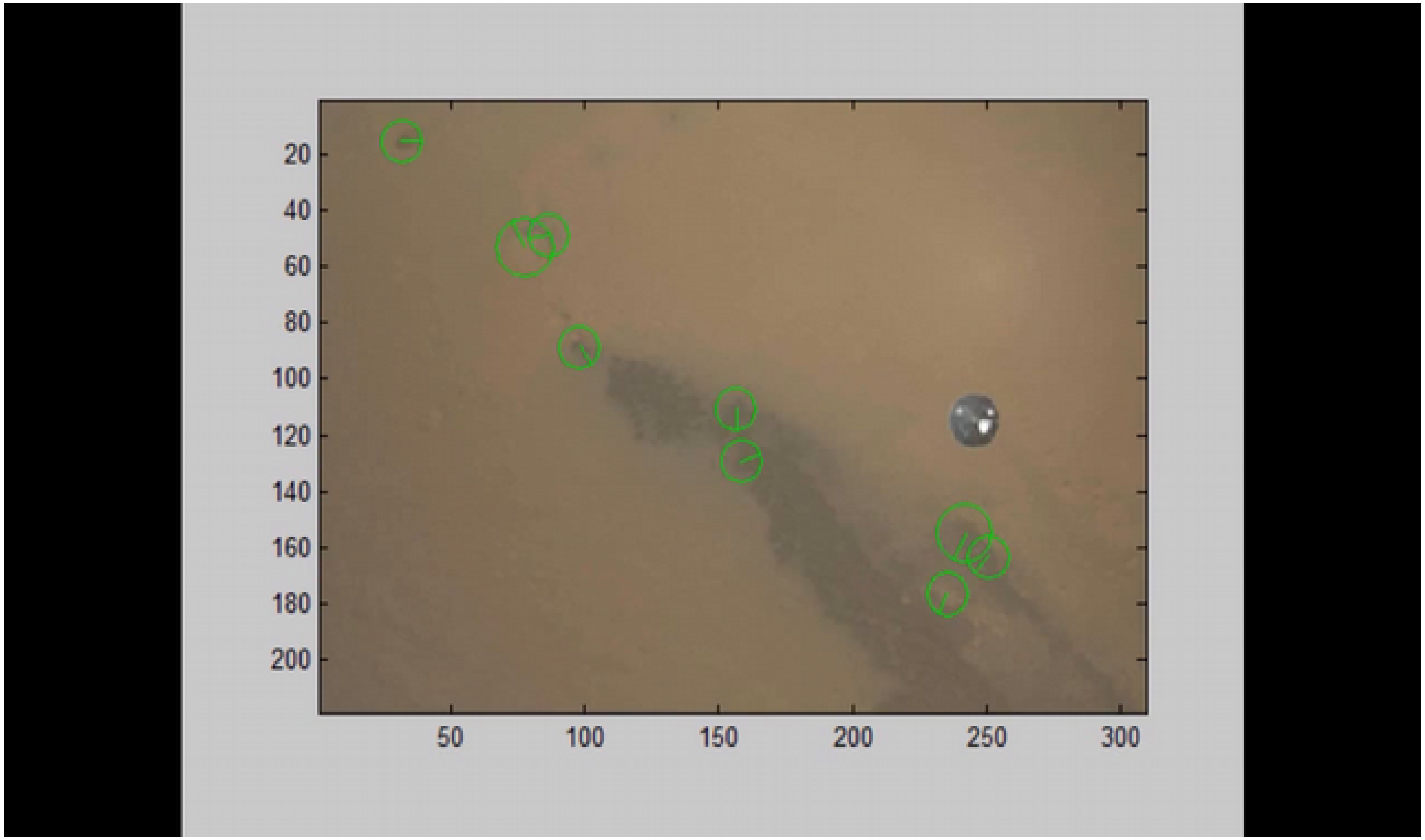
[https://www.youtube.com/watch?v=zQSFzmzR-is]

Marker-less tracking



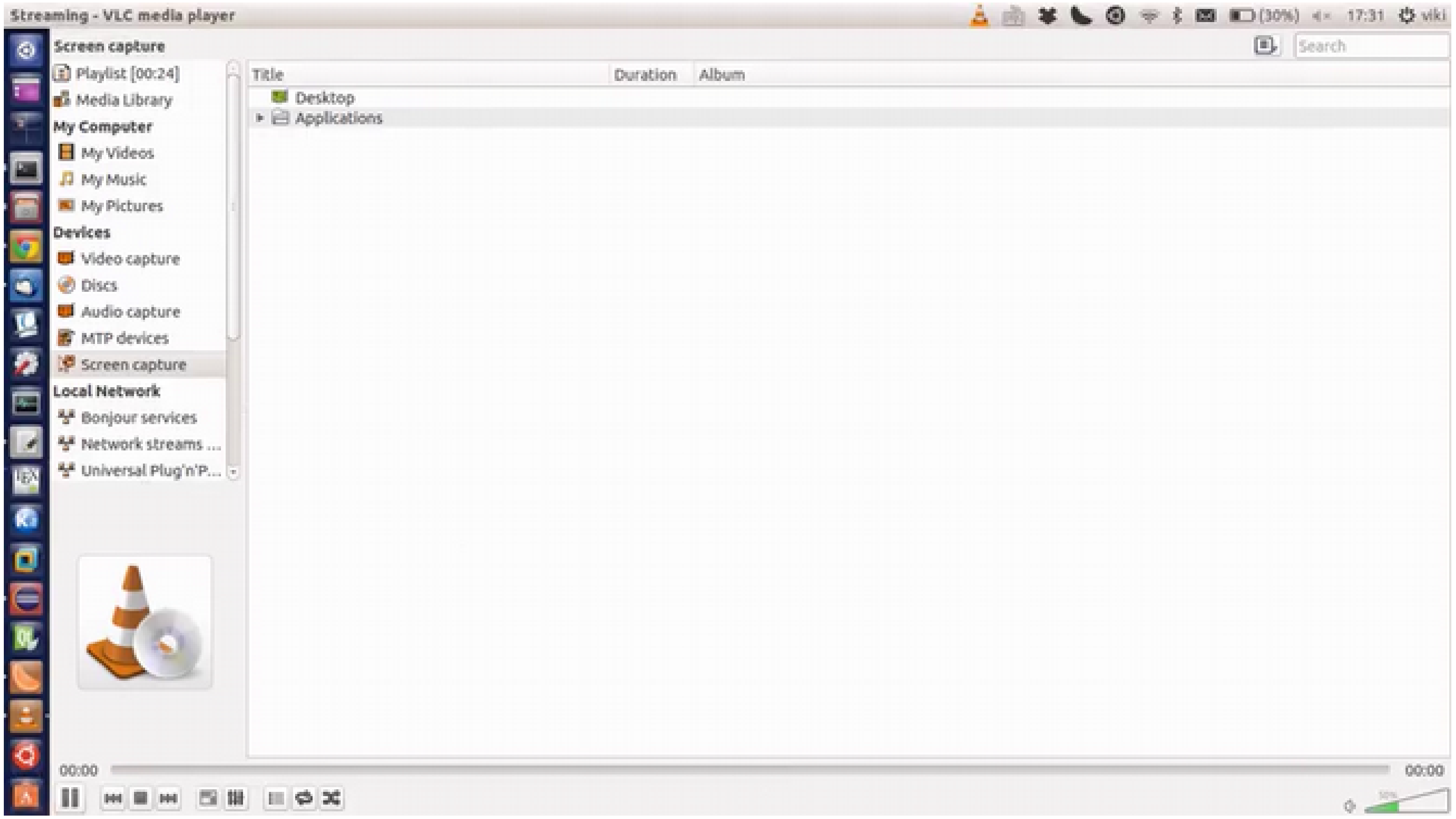
https://www.youtube.com/watch?v=caFHvamMUTw

SURF feature tracking during Curiosity Landing



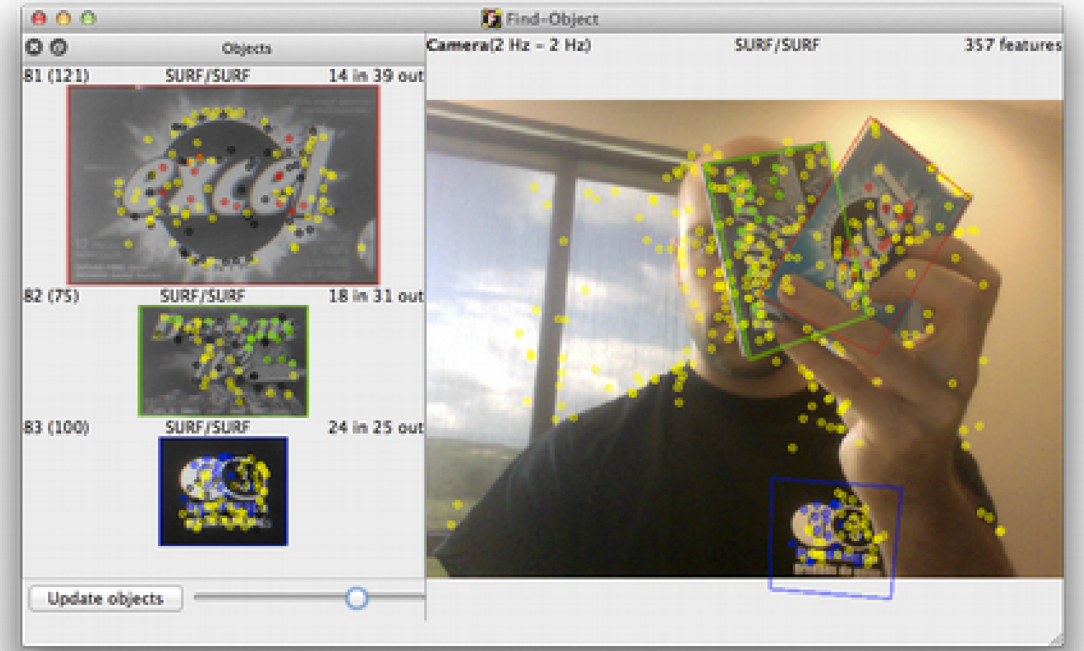
https://www.youtube.com/watch?v=Dgz0U4iWW_E

Visual-odometry



https://www.youtube.com/watch?v=kNy7ruXnWTo

Image Registration in ROS

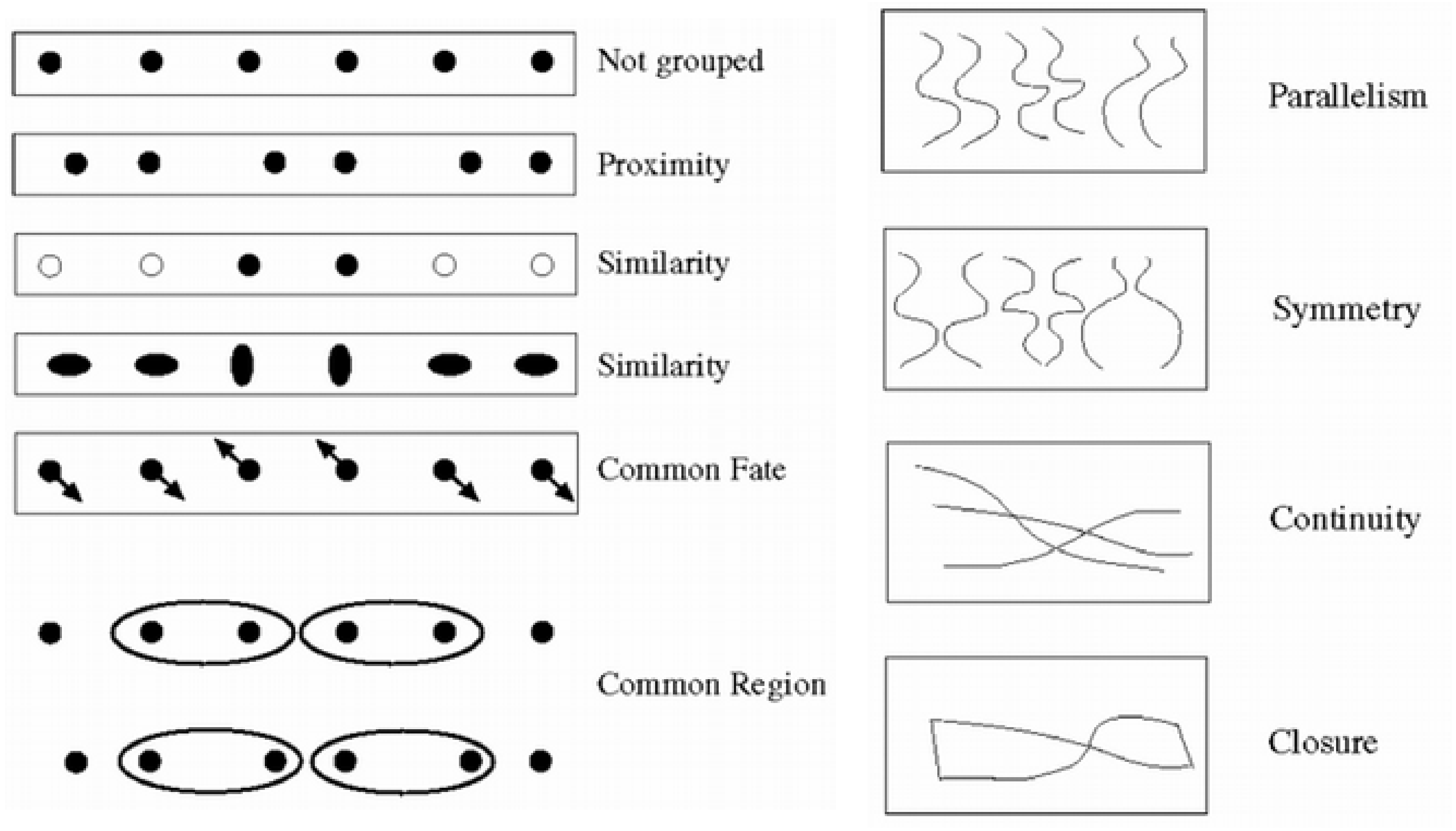


http://wiki.ros.org/find_object_2d

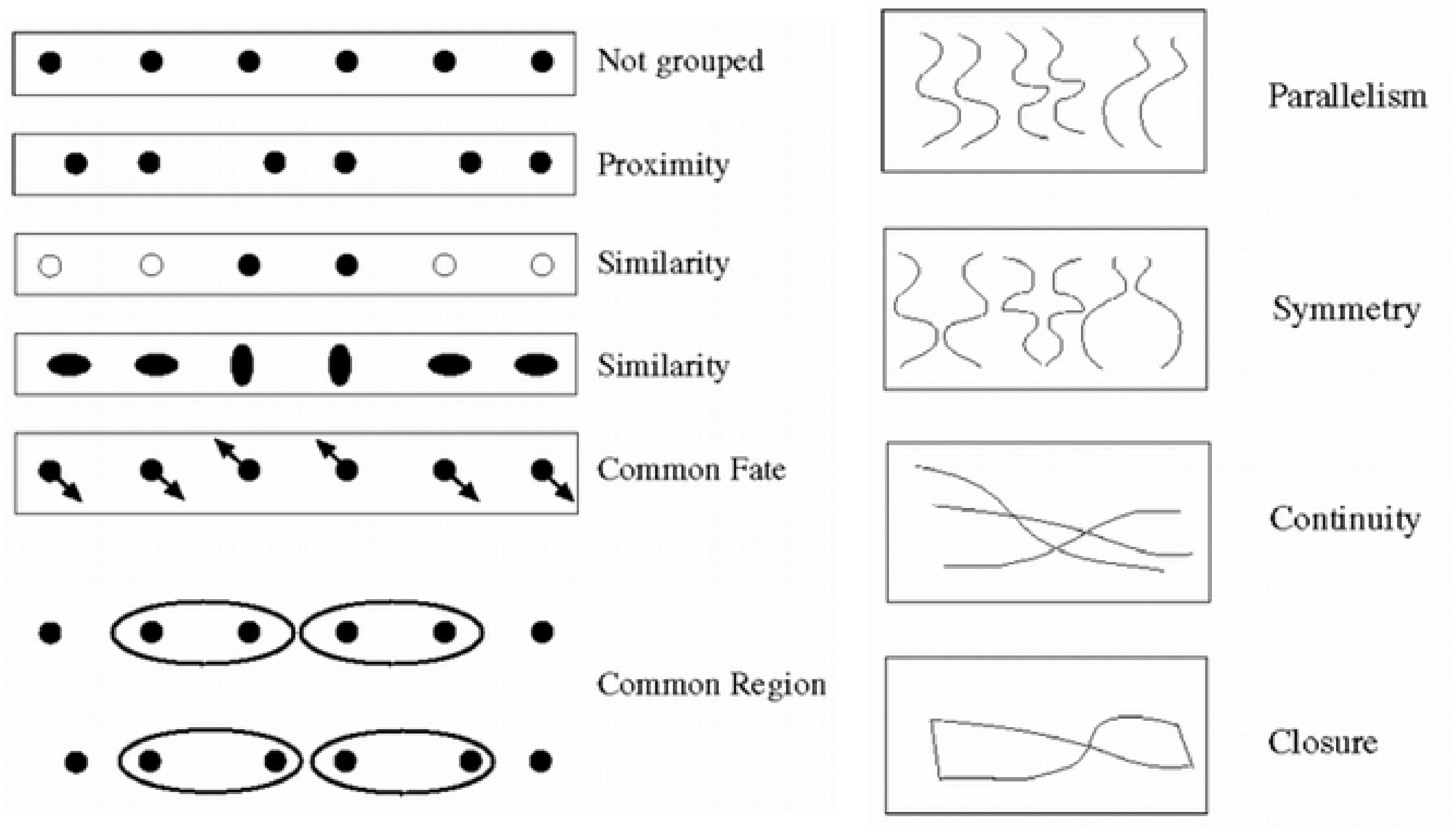
Optical Flow

- Interest key points and feature descriptors are great but suffer from one limitation:
  - They ignore time

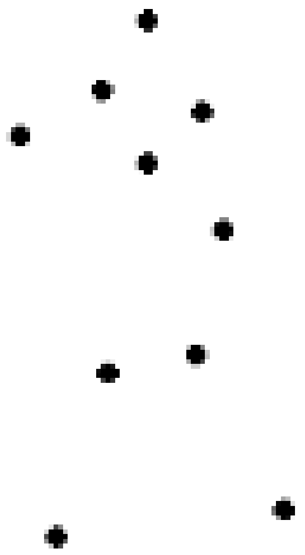
Why Optical Flow?



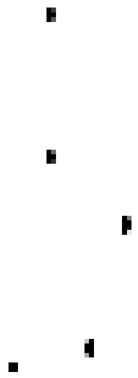
Why Optical Flow?



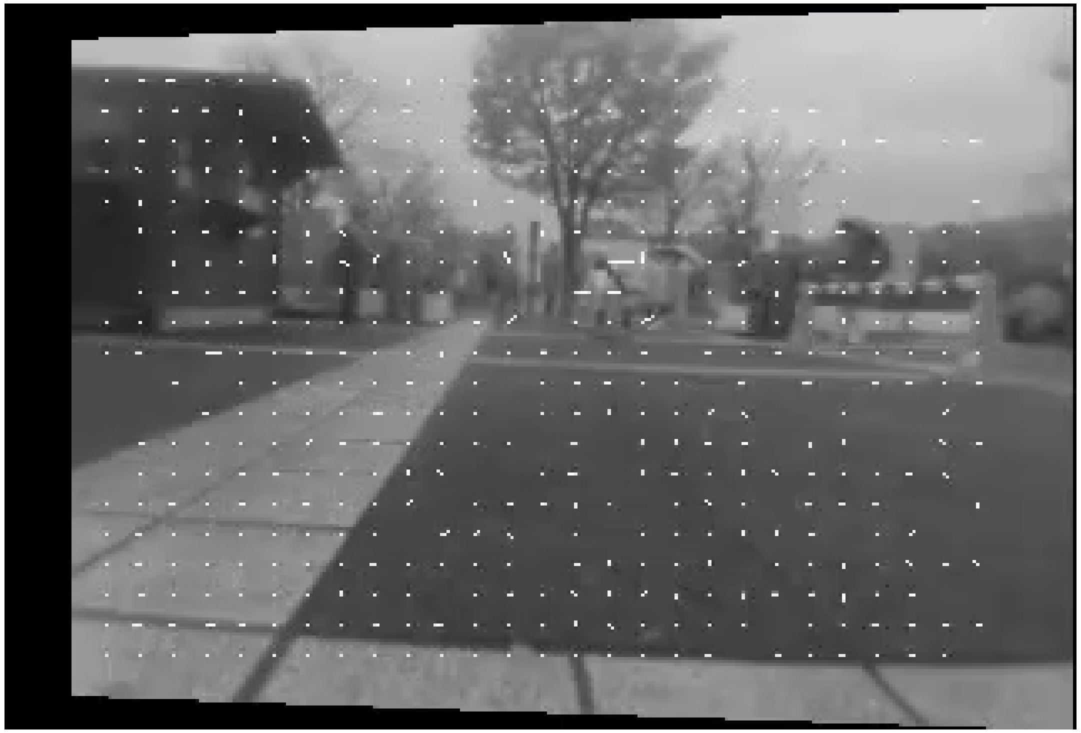
Why Optical Flow?



Why Optical Flow?

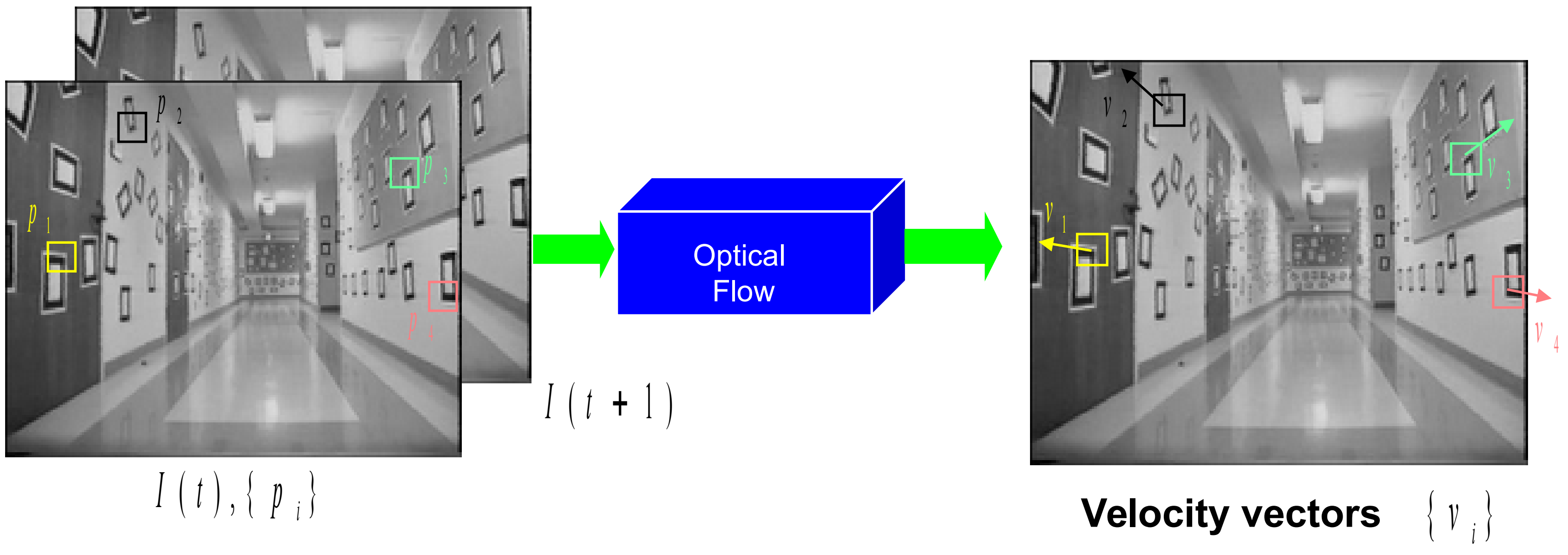


Optical Flow Video



https://www.youtube.com/watch?v=o8NOabnZPIY

What is Optical Flow?

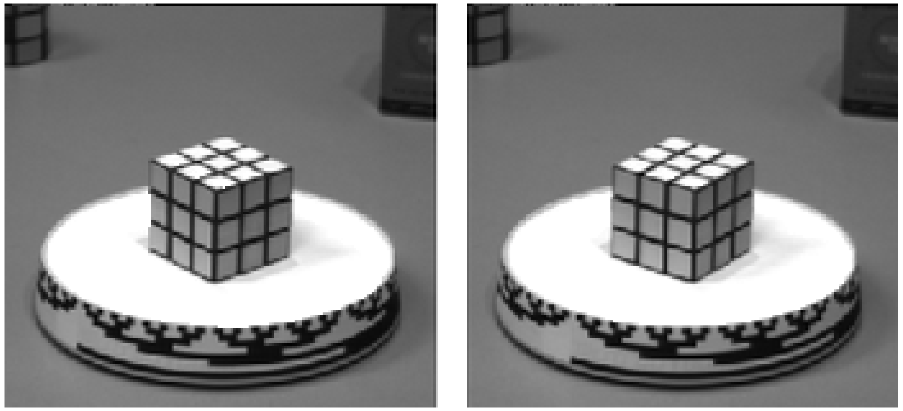
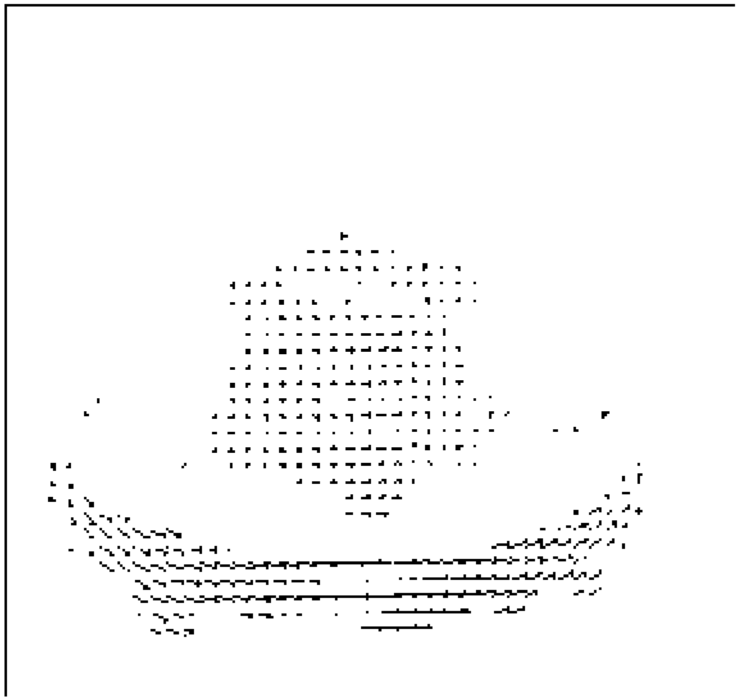


"Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image"

- Horn and Schunk, 1981

Motion Fields

- The motion field is the projection of the 3D scene motion into the image

Motion Fields and Camera Movement

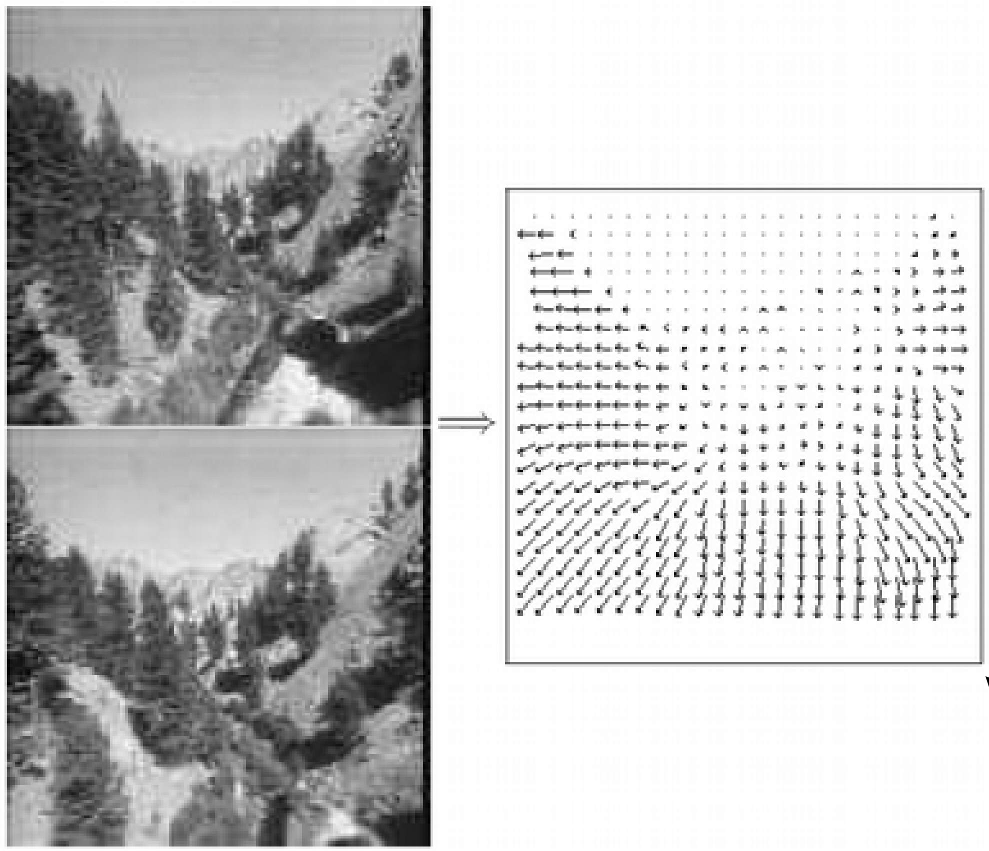


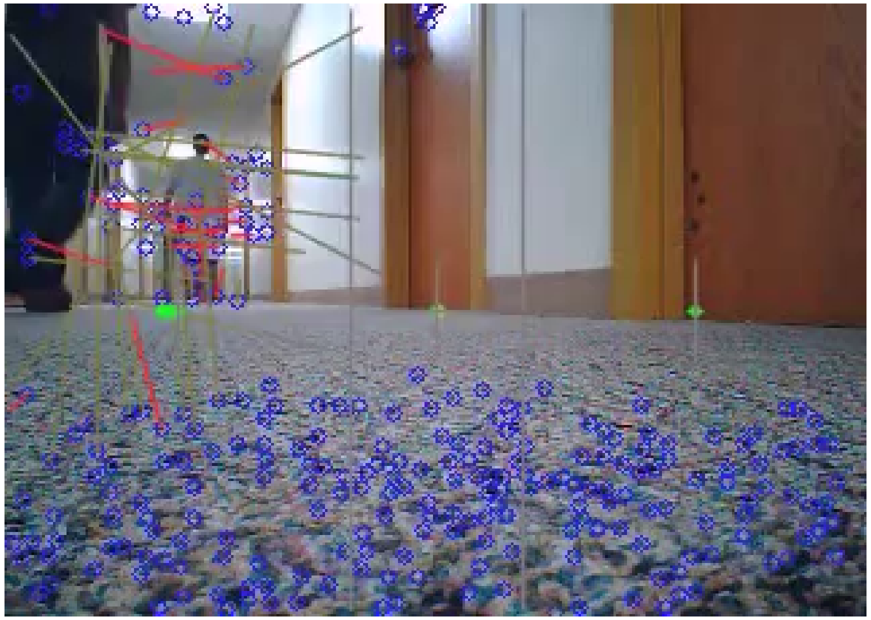
Figure 1.2: Two images taken from a helicopter flying through a canyon and the computed optical flow field.

Length of flow vectors inversely proportional to depth Z of 3D point

points closer to the camera move more quickly across the image plane

Figure from Michael Black, Ph.D. Thesis

Visual-odometry for Drones



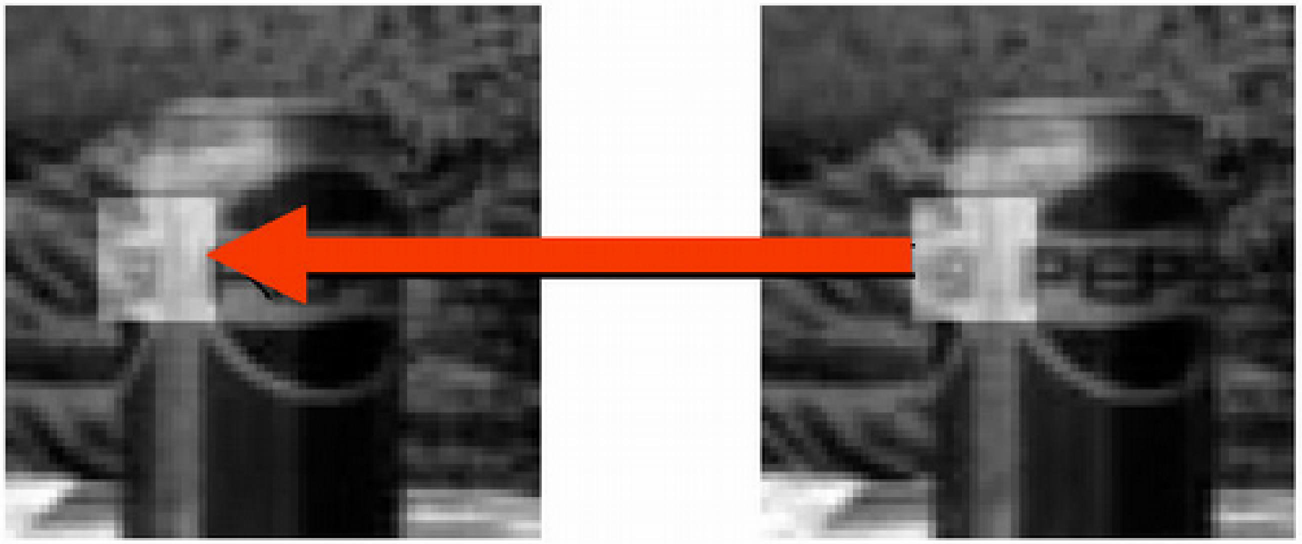
https://www.youtube.com/watch?v=V4r2HXGA8jw

Computing Optical Flow

- Given a set of points in an image, find those same points in another image
- Or, given point $[u_x, u_y]^T$ in image I_1 find the point $[u_x + \delta_x, u_y + \delta_y]^T$ in image I_2 that minimizes ε:

$$\varepsilon(\delta_x, \delta_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I_1(x,y) - I_2(x+\delta_x, y+\delta_y)\right)$$

Optical Flow Assumptions



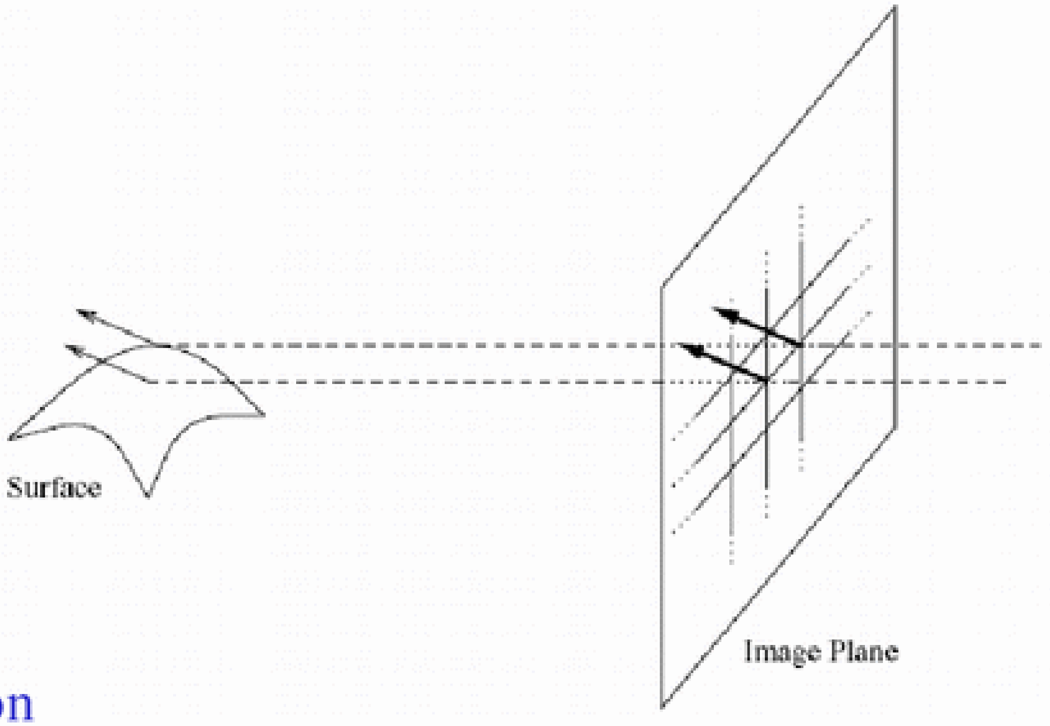
Assumption

Image measurements (e.g. brightness) in a small region remain the same although their location may change.

$$I(x+u, y+v, t+1) = I(x, y, t)$$

(assumption)

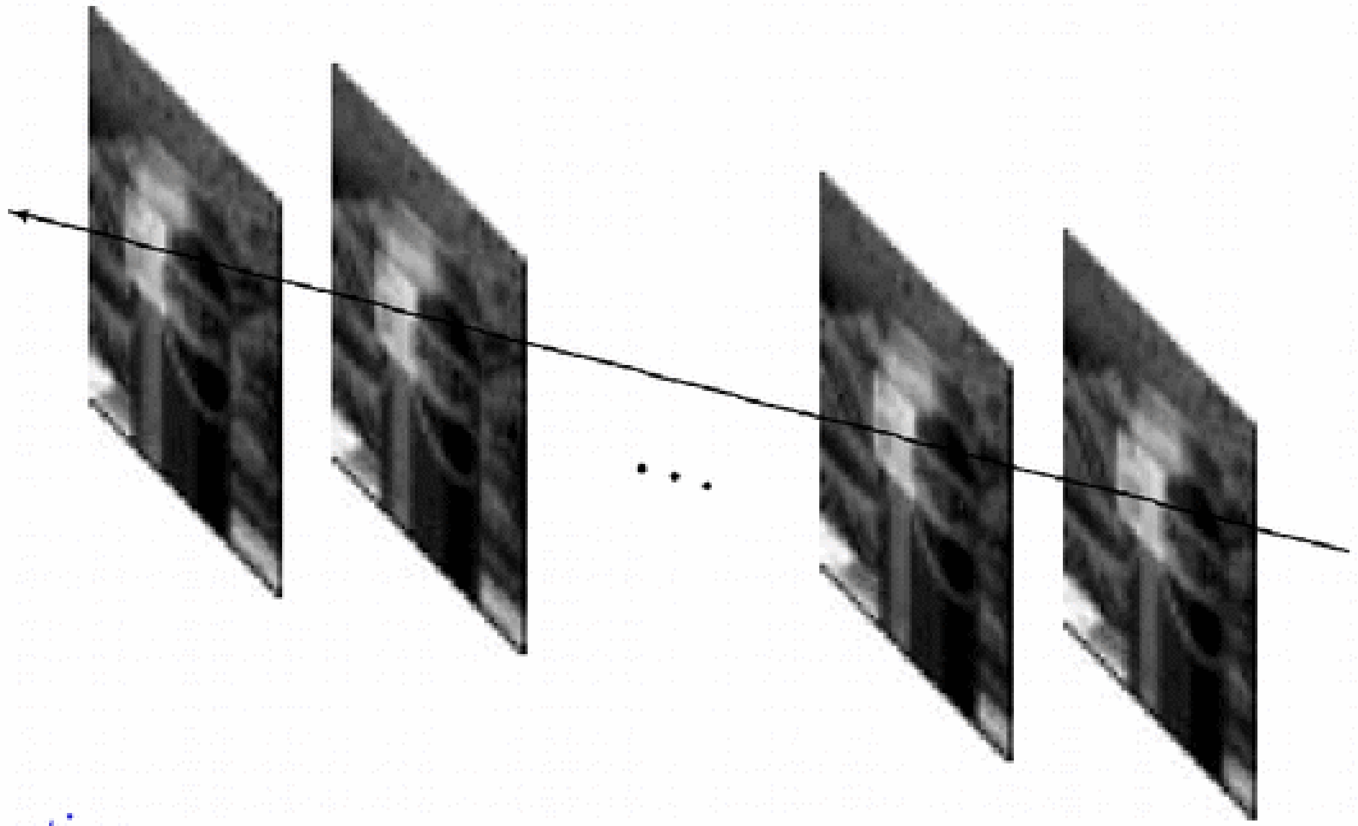
Spatial Coherence



Assumption

* Neighboring points in the scene typically belong to the same surface and hence typically have similar motions.
* Since they also project to nearby points in the image, we expect spatial coherence in image flow.

Temporal Persistence

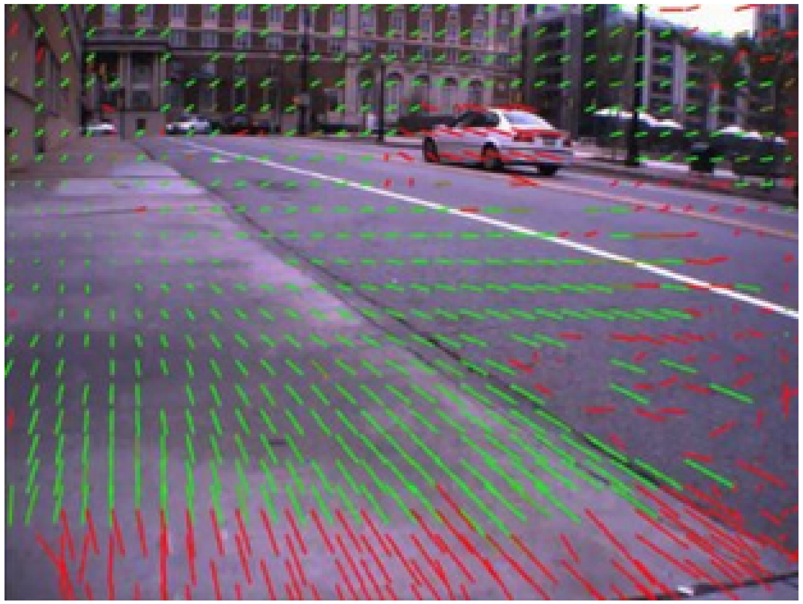


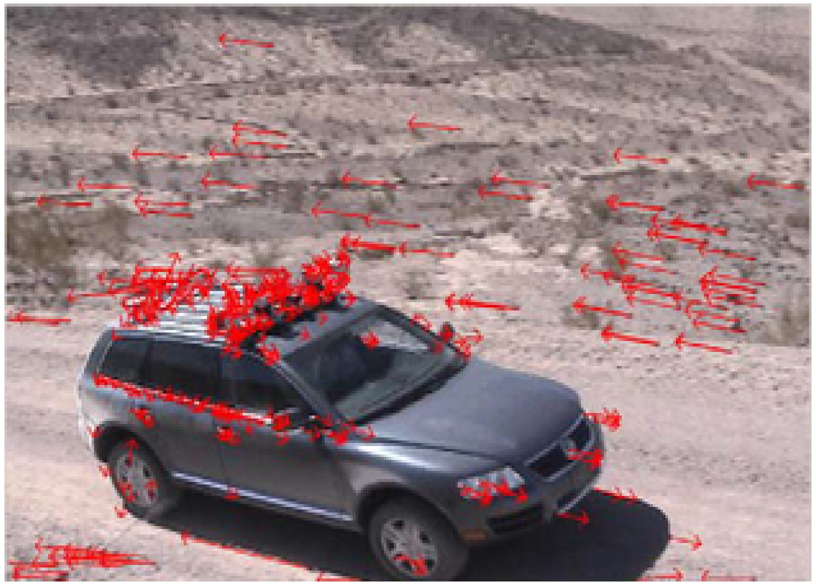
Assumption:

The image motion of a surface patch changes gradually over time.

* Slide from Michael Black, CS143 2003

Dense vs. Sparse Optical Flow





Code

- MATLAB:
  - Iterative Pyramidal LK Optical Flow
  - http://www.mathworks.com/matlabcentral/fileexchange/23142-iterative-pyramidal-lk-optical-flow
- OpenCV
  - http://robots.stanford.edu/cs223b05/notes/optical_flow_demo.cpp

To summarize…

- Feature detectors:
  - Find interest points in image (e.g., using difference of Gaussians, Harris corner detection, etc.)
- Feature descriptors
  - Each detected feature can be represented by a numerical descriptor encoding orientation, scale, etc.

To summarize…

- Optical Flow
  - Computes how pixels (or features) move from frame to frame
  - Dense optical flow computes a movement vector for each pixel
  - Sparse optical flow computes a movement vector only for a subset of pixels (e.g., the pixels that are interest points)
  - Optical flow can be used to infer movement of objects as well as movement of the camera

Project Breakout

- How does computer vision relate to your project?
- Will your robot need to process visual data – if so, what are some of the tasks and functions you will need?

THE END