University of Nottingham
UK | CHINA | MALAYSIA

COMP3065
Computer Vision

Topic 3 – Point Features: SIFT

Dr. Tianxiang Cui

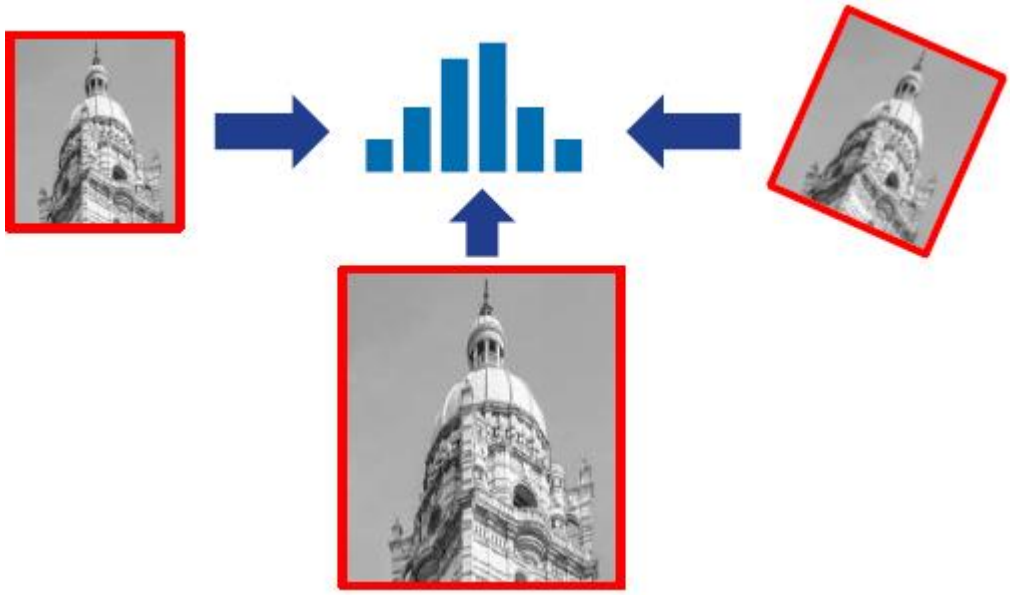
2025 Spring

Outline

- Feature points
- SIFT:
  - Scale-space extrema detection
  - Keypoint localization
  - Orientation assignment
  - Keypoint descriptor
- Properties/usages of SIFT

Feature Descriptors

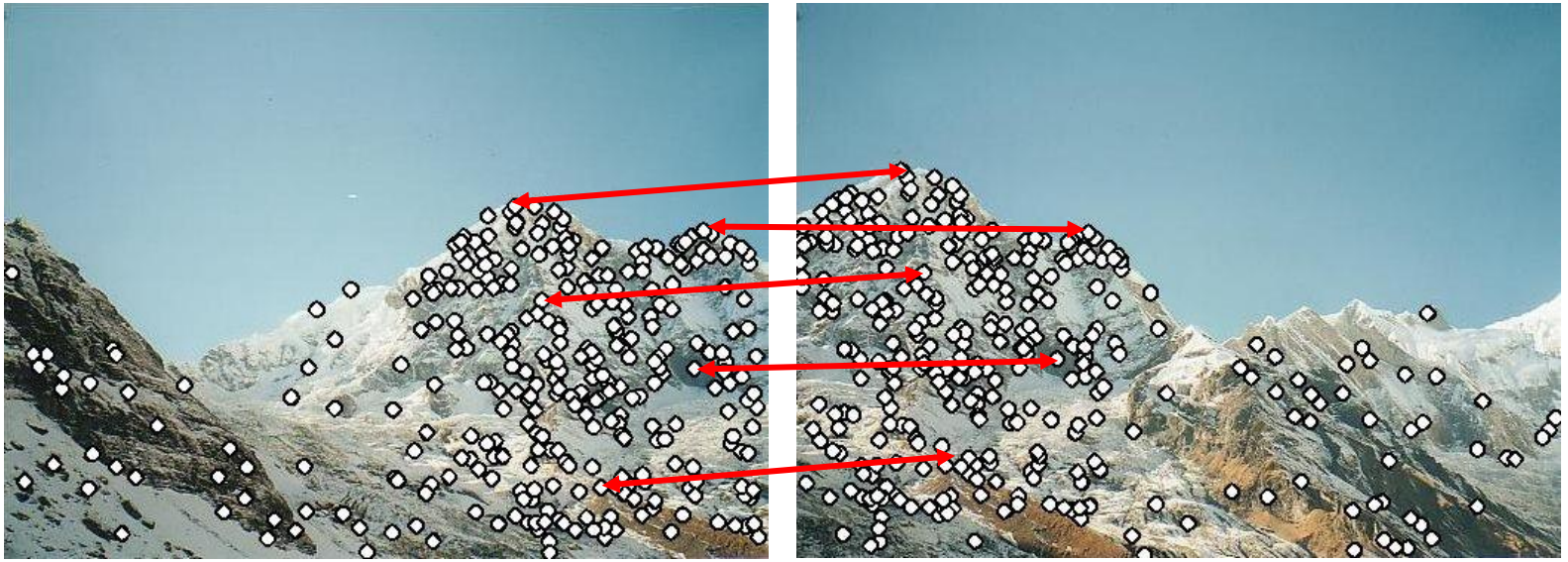
- We can now create feature vectors describing sections of an image.
  - Color
  - Texture
  - Shape
- Most feature descriptors can be thought of histograms (counts), ideally they should be
  - Robust and Distinctive
  - Compact and Efficient



Regions, Patches and Feature points

- Given an image there are many areas that could be considered, how do we find the important ones?
  - Shall we treat every areas equally?

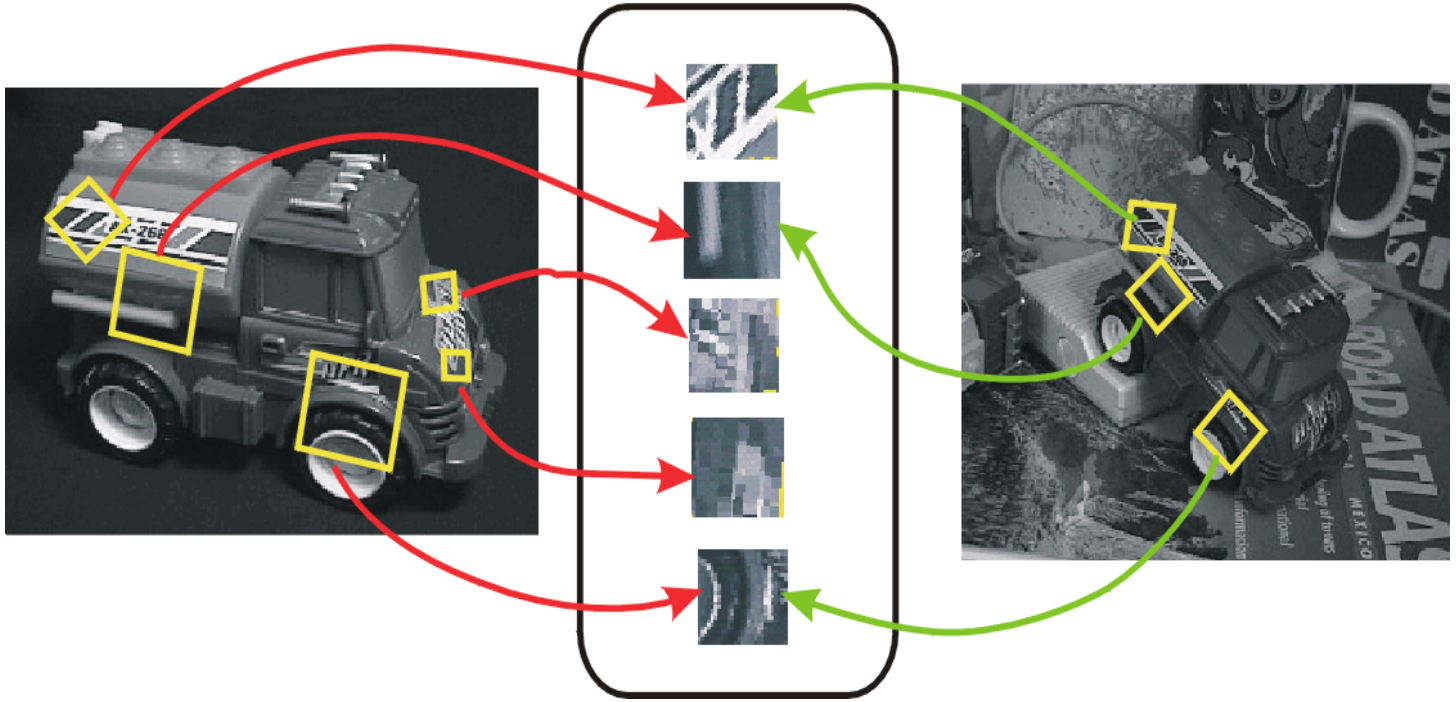
Pixels vs Feature Points



- Find a set of distinctive key points
- Extract feature descriptor around each interest point as vector
  - Matching between views, tracking over time……

Invariance

- A good feature detection/description algorithm should produce similar results when conditions vary.



- We want invariance to scale, translation, rotation, and illumination changes.

SIFT

- **Scale Invariant Feature Transform** (Lowe, ijcv-2004)
  - Transform image data into scale-invariant coordinates relative to local features.
  - ~77,786 citations, possibly the most successful vision paper ever.

- The method
  - Scale-space extrema detection (for **scale invariance**)
  - Keypoint localization (for **translation invariance**)
  - Orientation assignment (for **rotation/orientation invariance**)
  - Keypoint descriptor (for **illumination invariance**)

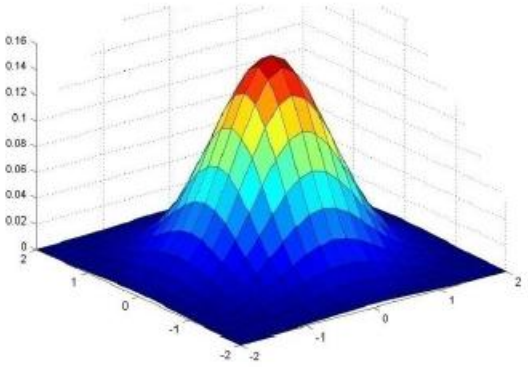
SIFT Overview

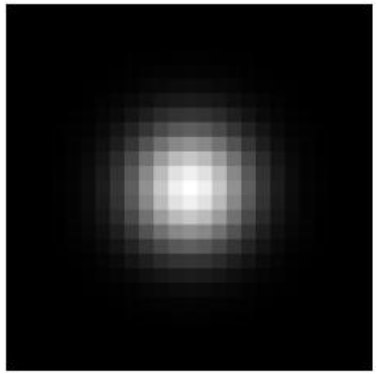
- Scale-space extrema detection
  - Search over all scales and image locations
  - Detect points that are invariant to scale and orientation
- Keypoint localization
  - A model is fit to determine the location and scale. Keypoints are selected based on measures of their stability
- Orientation assignment
  - Compute best orientation for each keypoint region
- Keypoint descriptor
  - Use local image gradients at selected scale and rotation to describe each keypoint region

The Scale Space

- Real world objects are meaningful only at a certain scale
- Ideally we want to get rid of some details from the image
  - Need to ensure we do not introduce new false details
  - Use **Gaussian Blur**

Gaussian Kernel



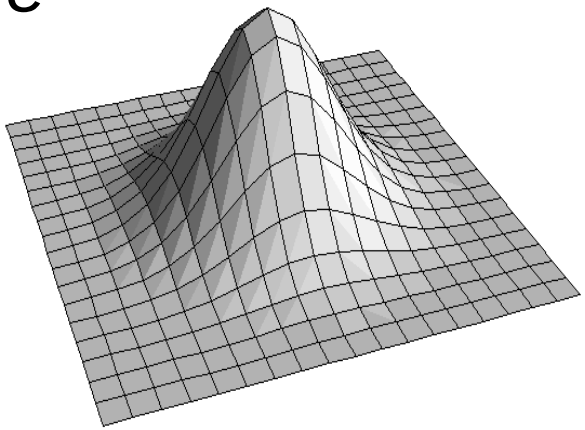


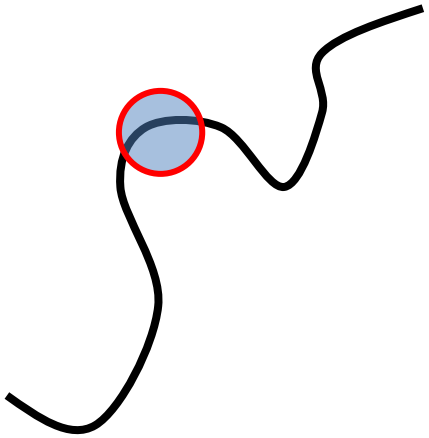
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Scale Invariance

- Find points whose surrounding patches (at some scale) are distinctive.
- Convolution with a Gaussian mask gives some idea of what is going on around a pixel.
- Gaussian masks have a natural scale.

$$P(x,y) = \frac{1}{\sigma^2 2\pi} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

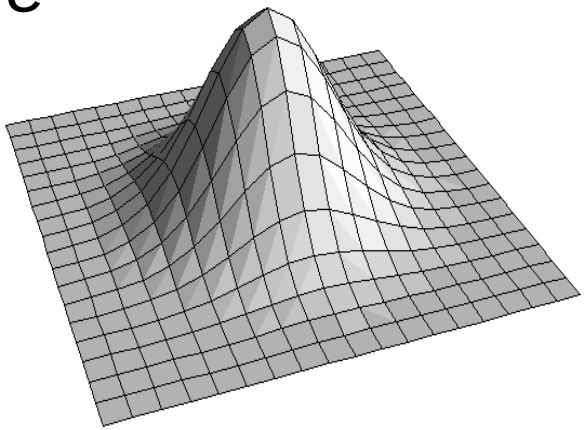


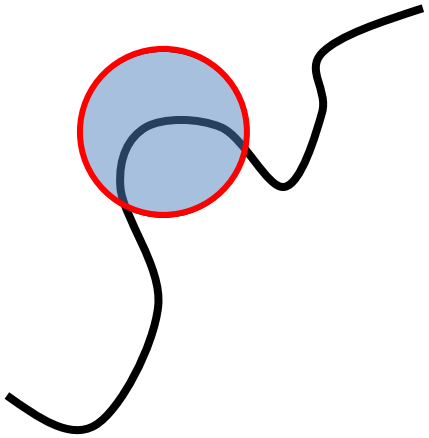


Scale Invariance

- Find points whose surrounding patches (at some scale) are distinctive.
- Convolution with a Gaussian mask gives some idea of what is going on around a pixel.
- Gaussian masks have a natural scale.

$$P(x,y) = \frac{1}{\sigma^2 2\pi} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

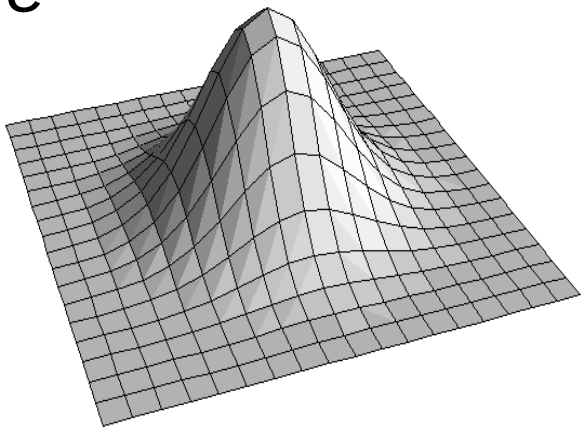


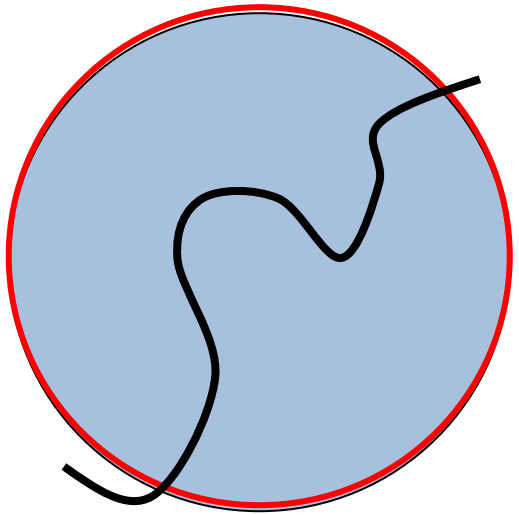


Scale Invariance

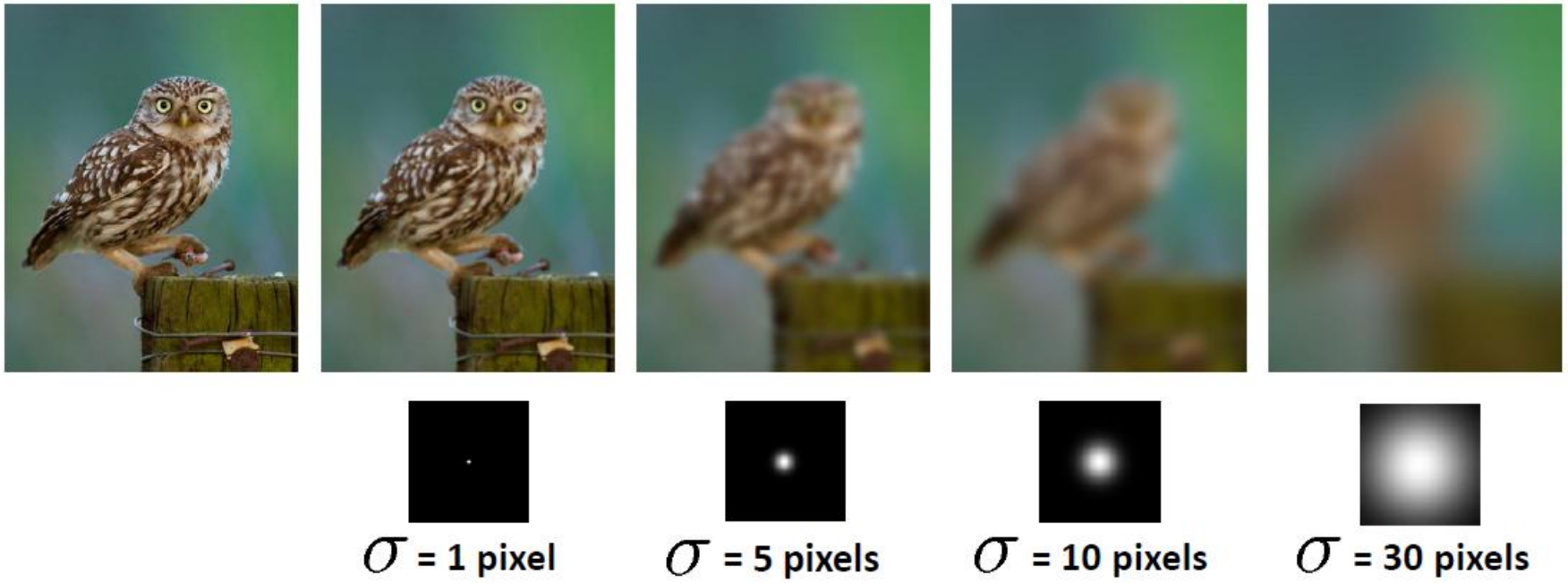
- Find points whose surrounding patches (at some scale) are distinctive.
- Convolution with a Gaussian mask gives some idea of what is going on around a pixel.
- Gaussian masks have a natural scale.

$$P(x,y) = \frac{1}{\sigma^2 2\pi} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$





Gaussian Filters



Larger values of σ produce greater blurring

Scale Spaces in SIFT

- Scale space is separated into octaves
- In each octave, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images
- Adjacent Gaussians are subtracted to produce the DOG
- Once a complete octave has been processed, the Gaussian image is resampled by taking every second pixel in each row and column to start the next level

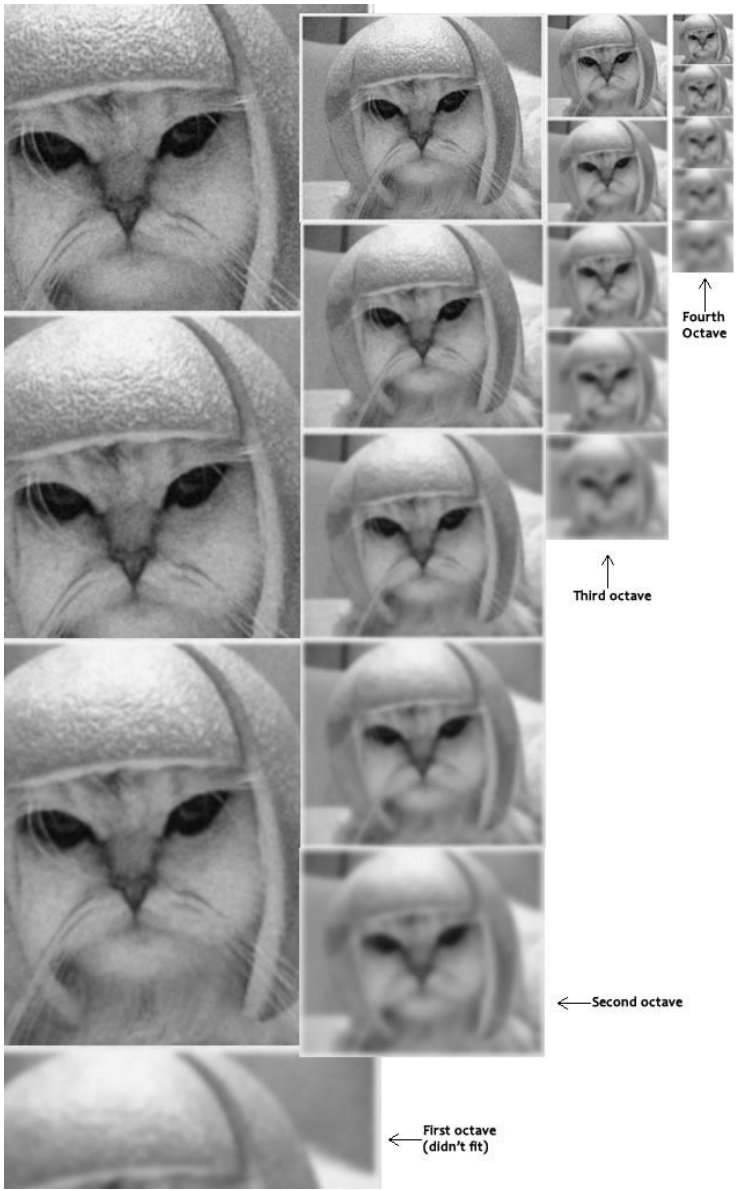
Scale Spaces in SIFT

- The number of octaves and scale depends on the size of the original image
  - Each octave's image size is half of the previous one.

- Repeatedly convolve the Gaussian with the initial image
  - Blurring

- $L(x,y,\sigma) = G(x,y,\sigma)*I(x,y)$

Scale space of an image | Gaussian | Initial image



SIFT Step 1: Summary

- Several octaves of the original image are generated
- Each octave's image size is half of the previous one
- Within an octave, images are progressively blurred using the Gaussian Blur operator

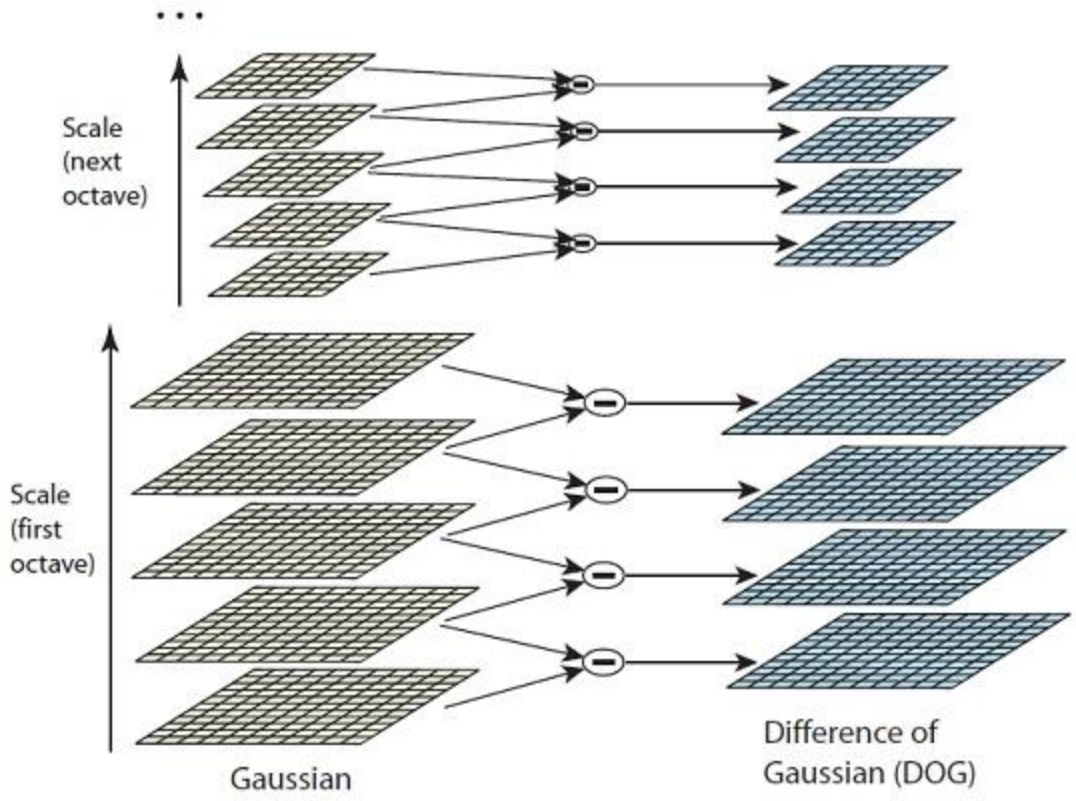
Laplacian of Gaussian

- The Laplacian is a measure of the 2nd spatial derivative of an image: $L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$
- It highlights regions of **rapid intensity change** and is therefore often used for **edge**/**corner**/**keypoint** detection
- The second order derivative is extremely sensitive to noise
  - The blur smooths it out the noise and stabilizes the second order derivative
- However, computationally intensive

Difference of Gaussians

- Calculate the **difference** between **two consecutive** Gaussian images

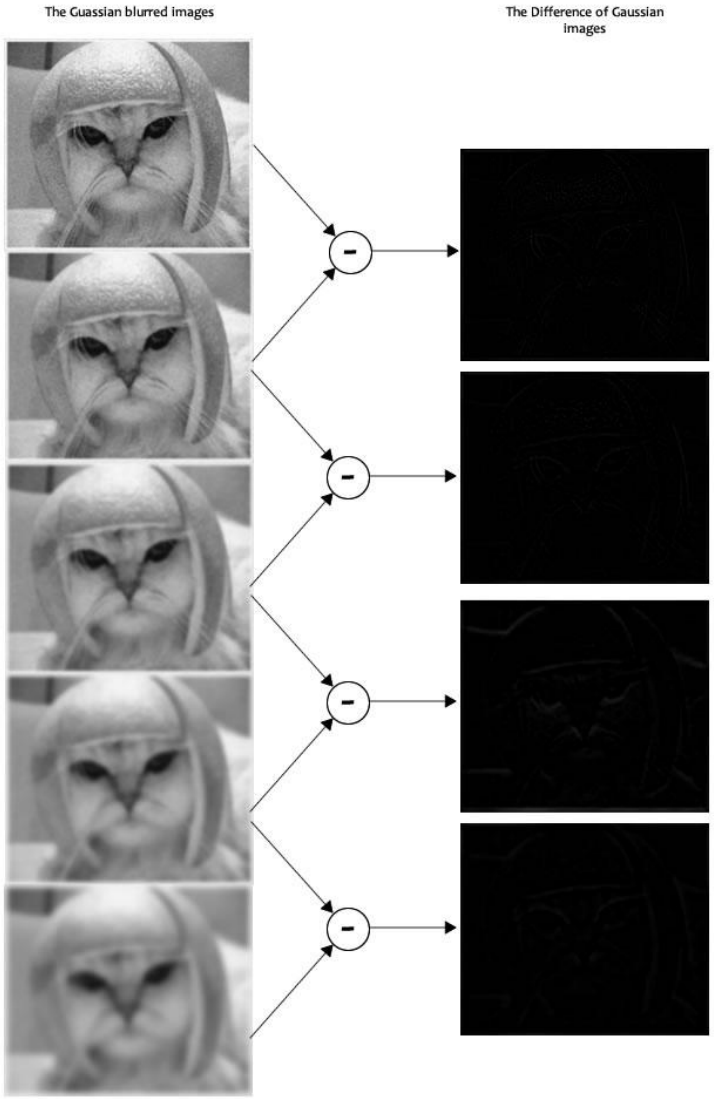
- These Difference of Gaussian images are approximately equivalent to the Laplacian of Gaussian



Difference of Gaussians

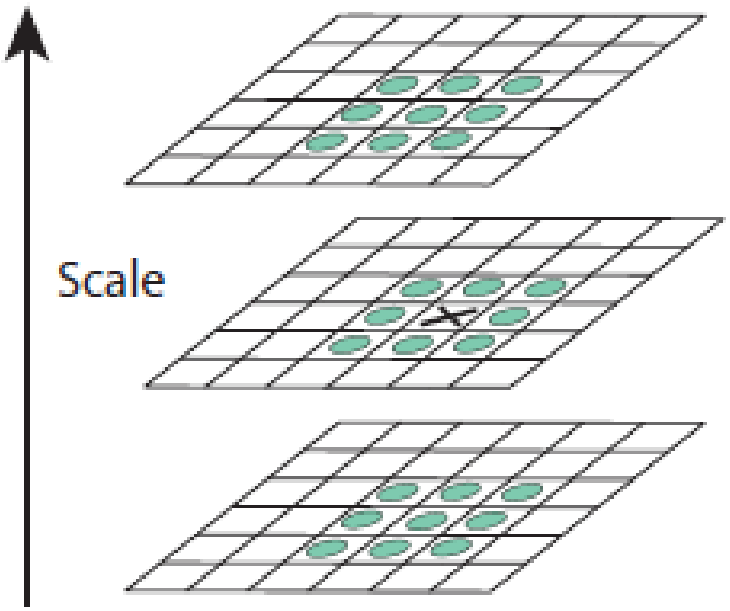
- Calculate the **difference** between **two consecutive** Gaussian images

- These Difference of Gaussian images are approximately equivalent to the Laplacian of Gaussian



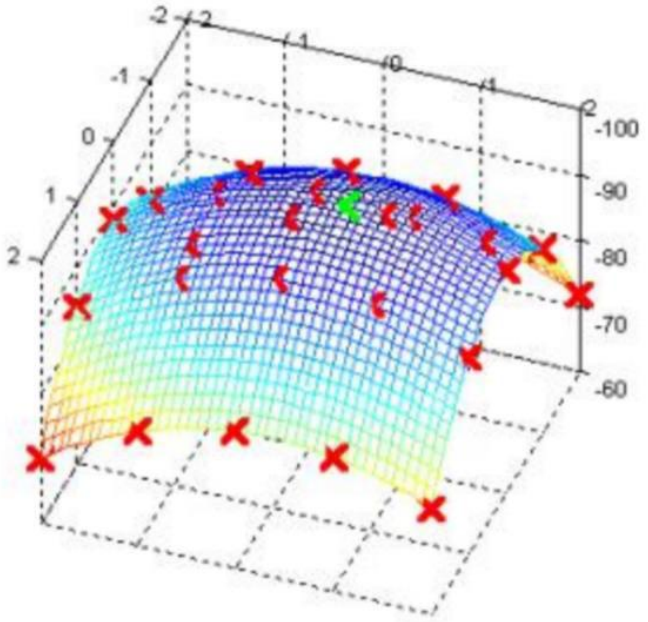
Extrema Detection

- Locate maxima/minima in **DoG** images
- A point is marked as a "key point" if it is the **greatest** or **least** of all neighbors
  - A non-maxima or non-minima position won't have to go through all checks
  - The keypoints are not detected in the lowermost and topmost scales
- The marked points are the approximate maxima/minima



Extremum Detection: Aside

- The maxima/minima almost never lies exactly on a pixel
- For each candidate key point, interpolation of nearby data is used to accurately determine its position



Extremum Detection: Aside

- This can be done by fitting a 3D quadratic function (e.g quadratic Taylor expansion of the Difference-of-Gaussian scale space function) to improve interpolation accuracy

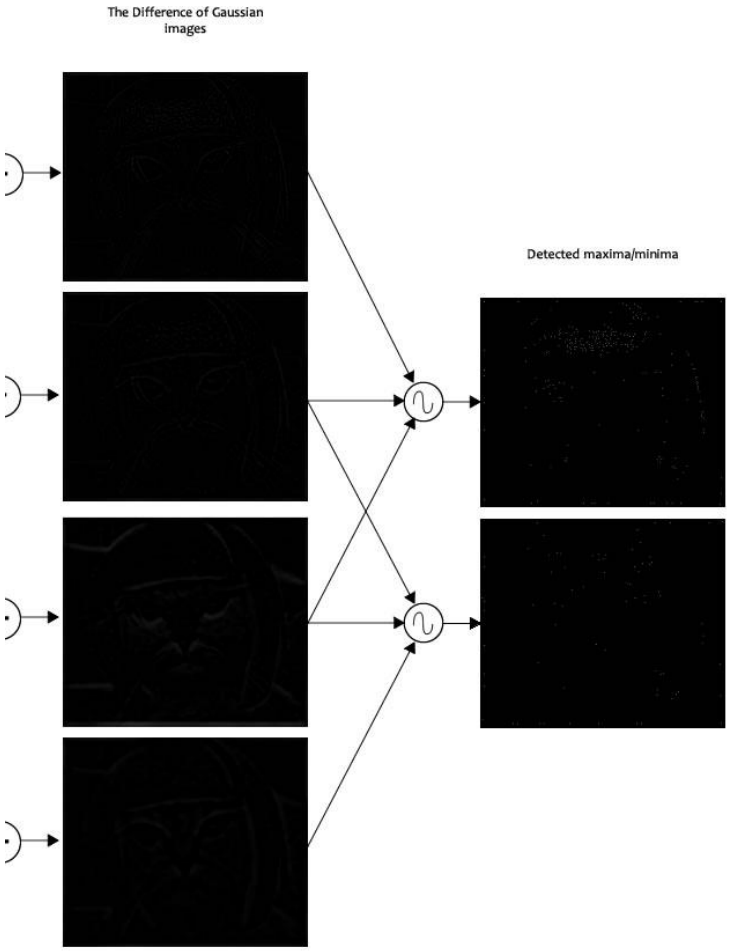
$$D(\mathrm{X}) = D + \frac{\partial D^T}{\partial \mathrm{X}}\mathrm{X} + \frac{1}{2}\mathrm{X}^T\frac{\partial^2 D}{\partial \mathrm{X}^2}\mathrm{X}$$

where $\mathrm{X} = (x, y, \sigma)^T$

- *D* and its derivates are evaluated at the sample point
- The offset for the extremum location is determined by taking the derivative of the above function with respect to X and setting it to zero

Extrema Detection

- The author of SIFT recommends generating two extrema images
- ? DoG images
- ? Gaussian blurred images



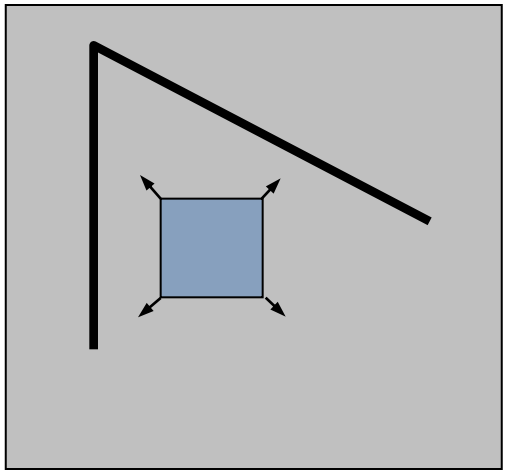
Getting Rid of Low Contrast Keypoints

- Some of the detected keypoints lie along an edge, or they don't have enough contrast
  - In both cases, they are not as useful as features and should be eliminated

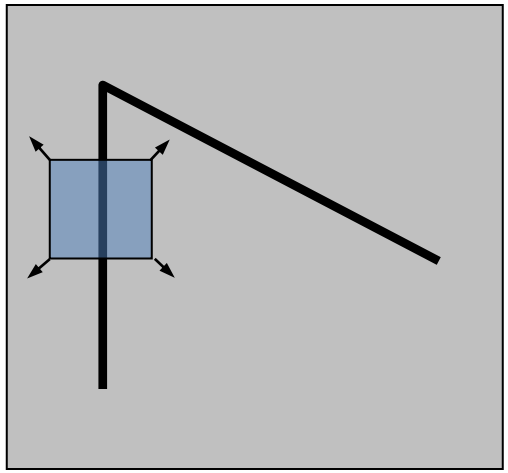
- If the DoG value at an extrema is less than a threshold (e.g 0.03), the keypoint is rejected

Getting Rid of Edges

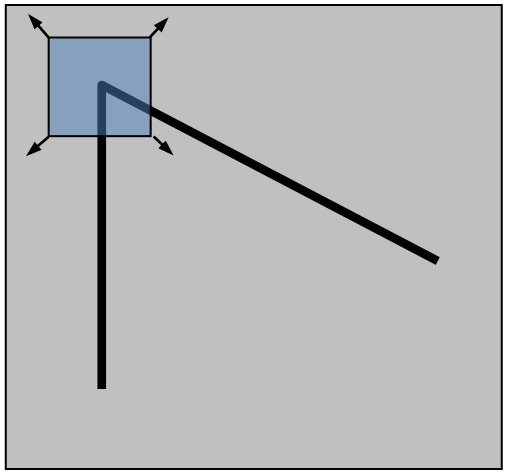
- The gradients of the key point are calculated:



"flat" region:
no change in all directions – all gradients are small



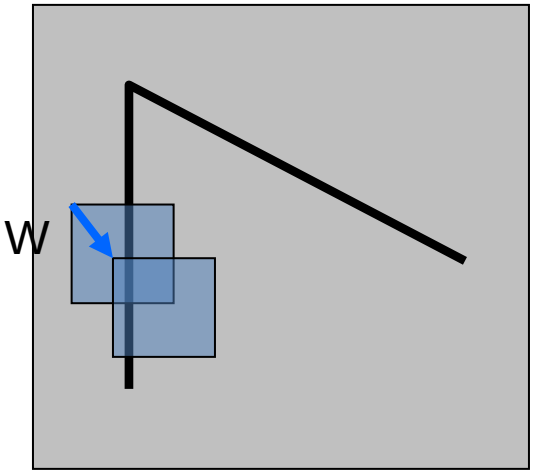
"edge":
change normal to the edge direction – the gradient along the edge will be small



"corner":
significant change in all directions – all gradients are large

Aside: Harris Corner Detector

- Consider shifting the window W by (u,v),
  - Compare each pixel value before and after moving using the sum of squared differences (SSD)
  - A large value of *E* is desired
  - High values of the terms inside the square brackets are needed



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

Aside: Harris Corner Detector

- Taylor series expansion gives

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

- If the motion (u,v) is small, then first order approximation should be good.

$$\begin{aligned} I(x+u, y+v) &\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

- Plugging this into the SSD equation…….

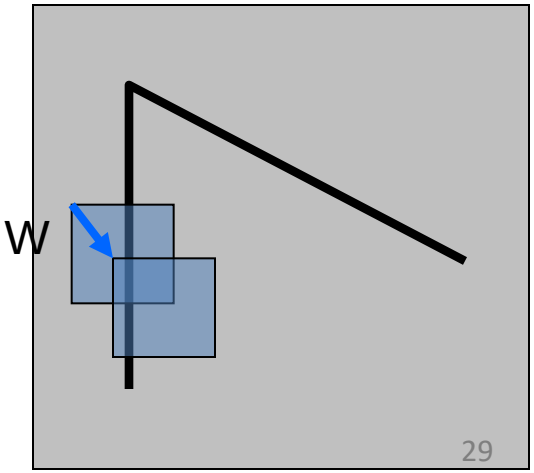
Aside: Harris Corner Detector

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2$$

$$\approx \sum_{(x,y)\in W} [I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)]^2$$

$$\approx \sum_{(x,y)\in W} \left[[I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$



Aside: Harris Corner Detector

- This can be re-written

$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} u \\ v \end{bmatrix}$$

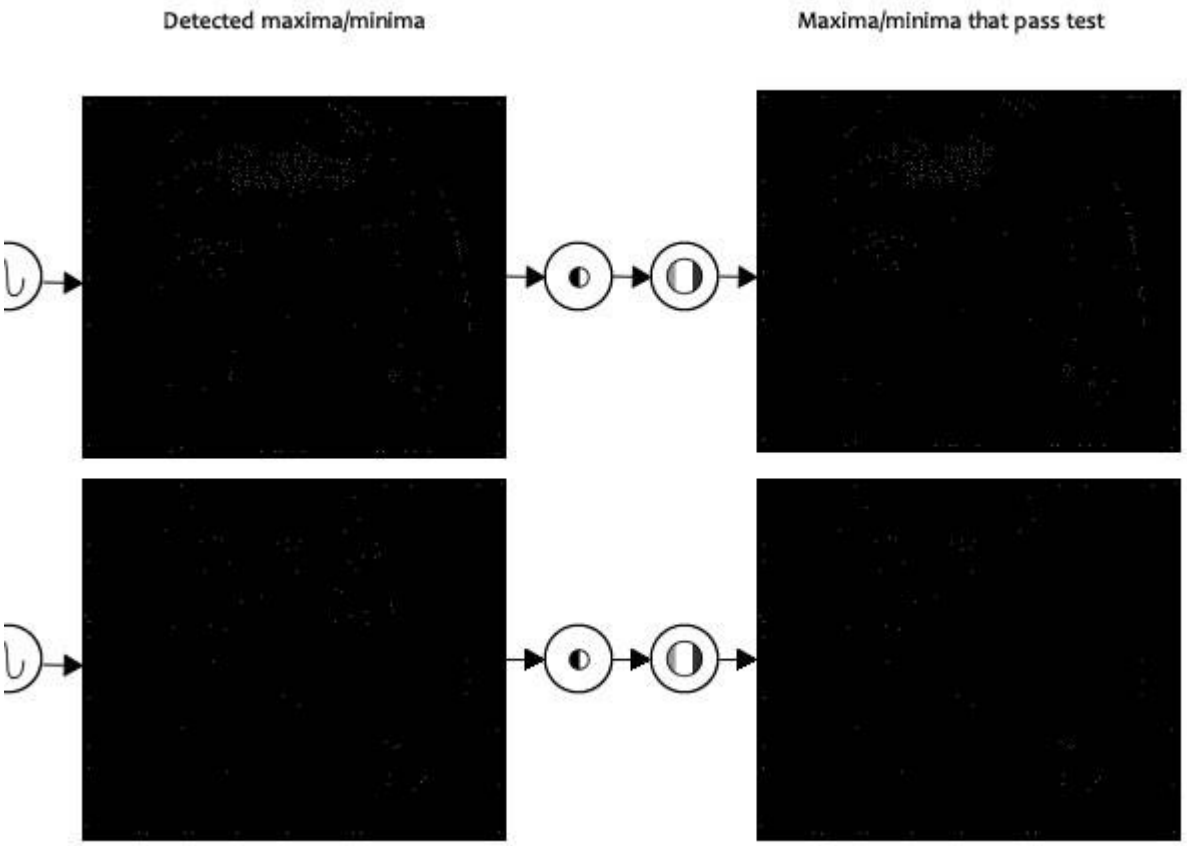
- Calculate measure of corner response: (($\lambda_1\lambda_2$) are eigenvalues of **H**)
  - *R = det(**H**) – k(trace(**H**)^2)*
  - *det(**H**) = $\lambda_1\lambda_2$ trace(**H**) = λ_1+λ_2 k ∈[0.04, 0.06]*
- Threshold on value *R*, suppress non-maximum value

SIFT: Eliminating Edge Responses

- In SIFT, efficiency is increased by just calculating the ratio of two eigenvalues
  - $r = \lambda_1/\lambda_2$
- Calculate:
  - $trace(\mathbf{H})^2/det(\mathbf{H}) = (\lambda_1+\lambda_2)^2/(\lambda_1\lambda_2) = (r\lambda_2+\lambda_2)^2/(r\lambda_2\lambda_2) = (r+1)^2/r$
- If the above ratio for a candidate keypoint is larger than a threshold, the keypoint is poorly localized and hence rejected

Keypoint Localization

- Both extrema images go through the two tests: the contrast test and the edge test



SIFT Step 2: Summary

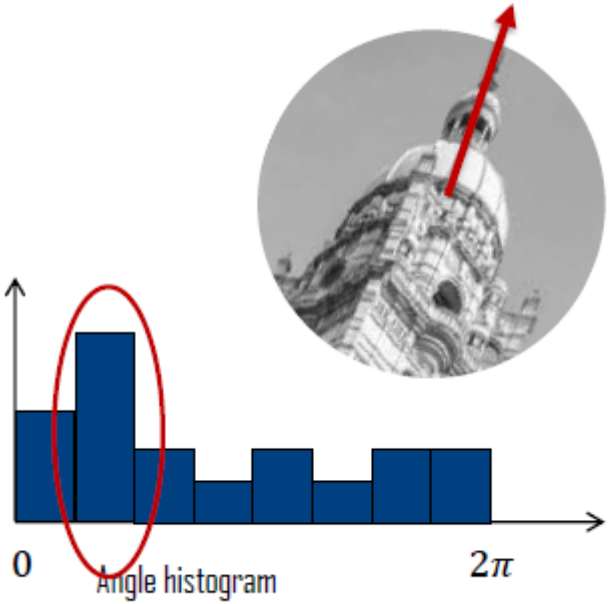
- Produce DoGs using two consecutive images in an octave for all octaves
- Detect the maxima/minima in the DoG images
- Reject the keypoints if they had a low contrast or if they were located on an edge

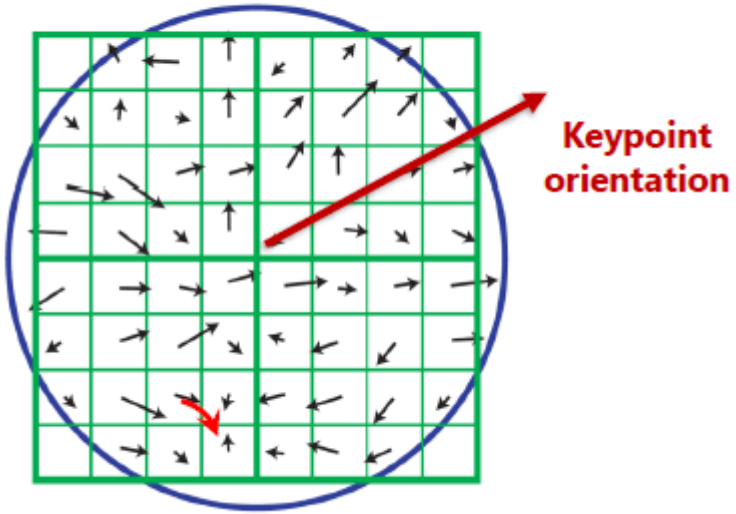
Orientation Estimation

- Each keypoint should be characterized by location, scale, **orientation**
- Collect gradient directions and magnitudes around each keypoint
- Figure out the most prominent orientation in that region
- Assign this orientation to the keypoint
- Any later calculations are done relative to this orientation
  - This ensures rotation invariance

SIFT Step 3: Summary

- To assign an orientation we calculate the gradient magnitude and direction of a small region around the keypoint
- Using the histogram, the most prominent gradient orientation is identified
  - Peak of the histogram
- Assign it to the keypoint





SIFT Descriptor

- At this point, each keypoint has the **location, scale, orientation**
- Next is to compute a descriptor for the local image region(window) around each keypoint
- Region normalization
  - Rotate the window to standard orientation
  - Scale the window size based on the scale at which the point was found

SIFT Descriptor

- Take 16x16 square window around detected keypoint.
- Compute edge orientation (angle of the gradient – 90 degree) for each pixel.
- Throw out weak edges (threshold gradient magnitude).
- Create histogram of surviving edge orientations.

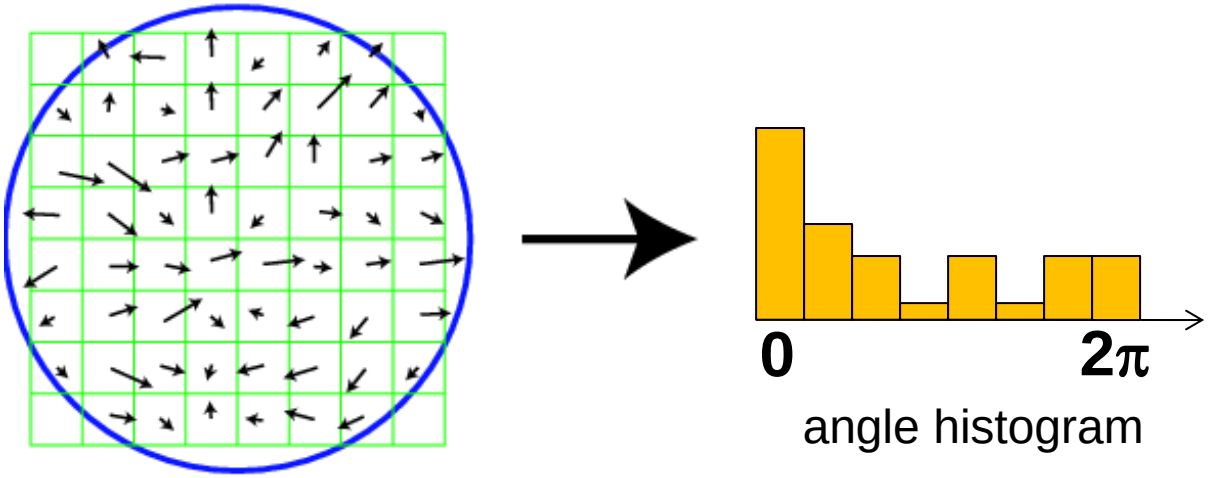
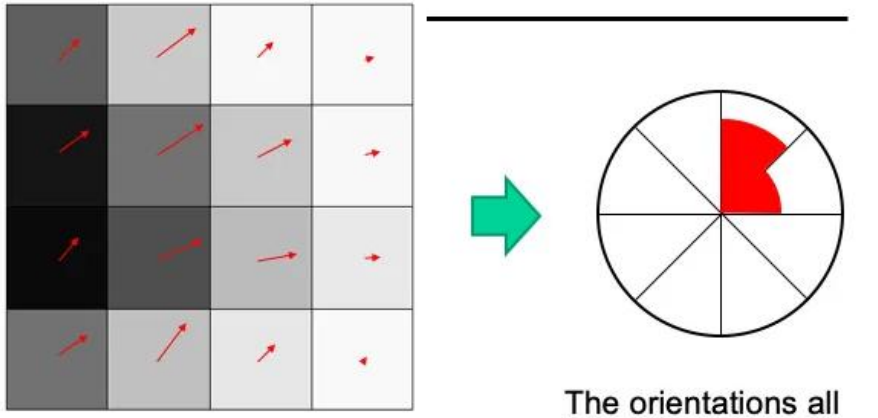


Image gradients

SIFT Descriptor

- Full version:
  - Divide the 16x16 window into a 4x4 grid of cells
  - Within the 4x4 cell, the orientations and gradient magnitudes are calculated
  - Put these orientations into an 8 bin histogram (the amount added to the bin depends on the magnitude of the gradient)



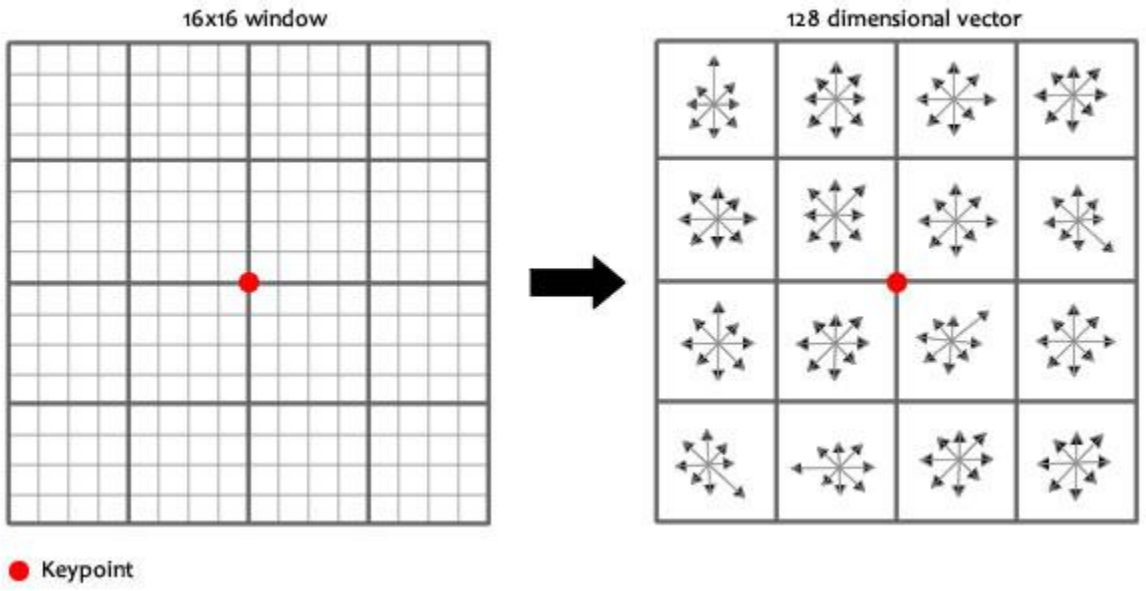
Orientations in each of the 16 pixels of the cell

The orientations all ended up in two bins: 11 in one bin, 5 in the other. (rough count)

5 11 0 0 0 0 0 0

SIFT Descriptor

- Full version:
  - ? cells * ? orientations = ? dimensional descriptor
  - Normalize the vector
  - Clamp all vector values > 0.2 to 0.2
  - Renormalize



SIFT: Invariance Properties

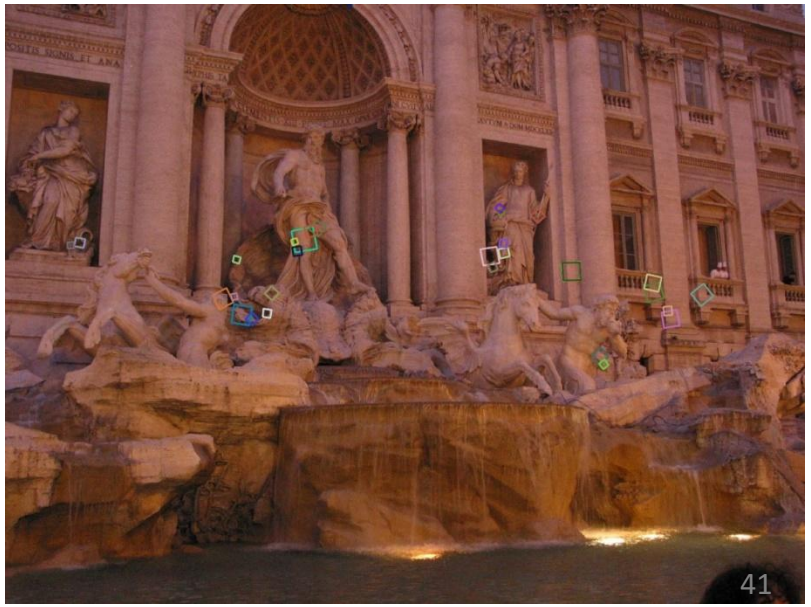
- To be robust to intensity value changes
  - Use gradient orientations
- To be scale invariant
  - Estimate the scale using scale space extrema detection
  - Calculate the gradient after Gaussian smoothing with this scale
- To be orientation invariant
  - Rotate the gradient orientations using the dominant orientation in a neighborhood
- To be Illumination invariant
  - Working in gradient space, so robust to $I = I + b$
  - Normalize vector to [0…1], robust to $I = \alpha I$ brightness changes
  - Clamp all vector values > 0.2 to 0.2, robust to “non-linear illumination effects”

Properties of SIFT

Fast and efficient - can run in real time - lots of code available.

Can handle

- changes in viewpoint, up to about 60 degree out of plane rotation.
- significant changes in illumination, sometimes even day vs. night.



Uses of SIFT

- Pose estimation
- 3D reconstruction
- Object recognition
- Image retrieval
- …

Conclusion

- Features should be distinctive, and invariant to as much as possible:
  - Scale
  - Rotation
  - Translation
  - Illumination
- David Lowe's SIFT addresses the problem explicitly
- Fast approximations exist
  (e.g. SURF: Speeded-Up Robust Features)