دانشگاه
خواجه نصیرالدین طوسی
K. N. Toosi University
of Technology

Computer Vision

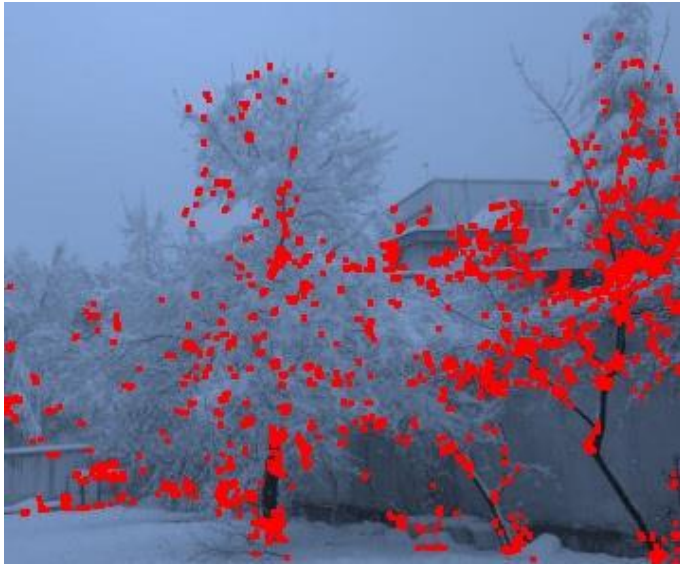
Lecture 8: Region Descriptors (HOG & SIFT)

Dr. Esmaeil Najafi

MSc. Javad Khoramdel

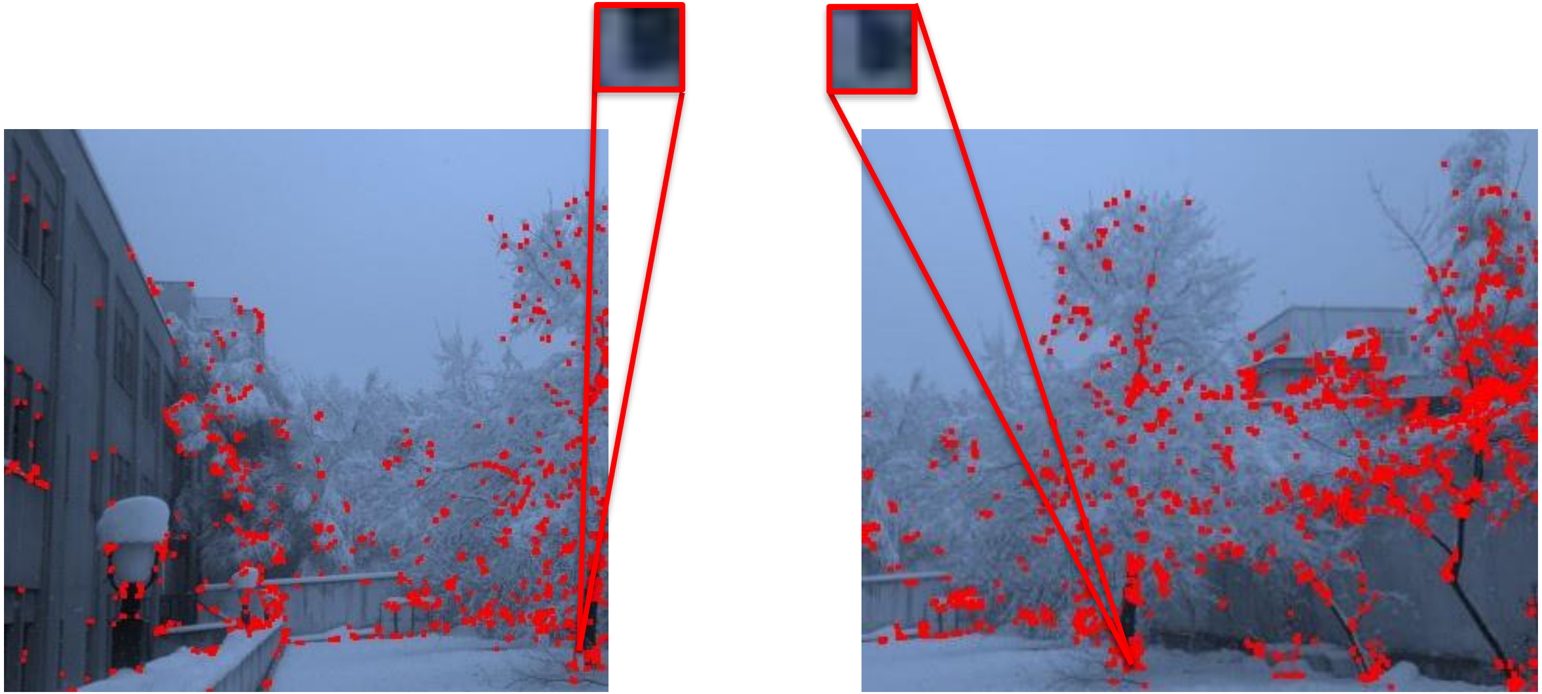
Finding the similar patches

- We've found the corners in each of these two images.



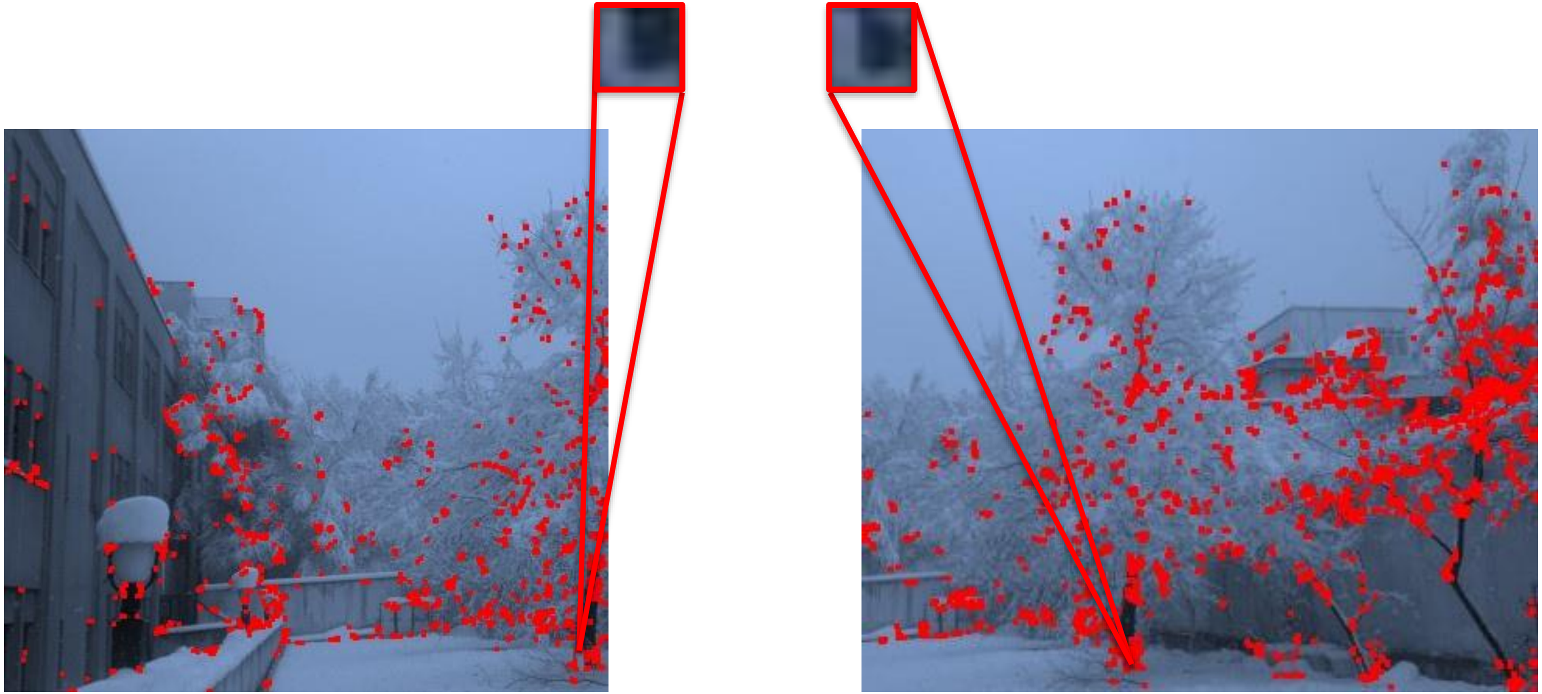
Finding the similar patches

- Now, the question is how do find similar points in these images?



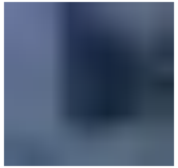
Finding the similar patches

- Given these two patches, how do we compare them? How do we find whether they are similar or not?



Raw pixel values

- Given these two patches, how do we compare them? How do we find whether they are similar or not?
  - Maybe L2 distance between raw pixel values?
    - ✓ For two given image patches, P_1 and P_2, we can calculate the L2 distance between their raw pixel values.
    - ✓ $D(P_1, P_2) = \sqrt{\Sigma_c \Sigma_x \Sigma_y |\mathrm{P}_1(\mathrm{x}, \mathrm{y}, c) - \mathrm{P}_2(\mathrm{x}, \mathrm{y}, \mathrm{c})|^2}$



Raw pixel values

- Maybe L2 distance between raw pixel values??
  - $D(P_1, P_2) = \sqrt{\Sigma_c \Sigma_x \Sigma_y |P_1(x, y, c) - P_2(x, y, c)|^2}$

Red Channel

91	30	33
98	27	36
89	91	92

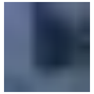
Green Channel

112	42	42
118	40	43
108	106	108

Blue Channel

133	54	51
143	58	61
138	134	133

$D = 55.57$



107	40	82
96	23	52
81	80	85

123	55	93
112	39	64
100	98	100

161	77	116
147	55	81
132	119	127

Raw pixel values

- Maybe L2 distance between raw pixel values??
  - $D(P_1, P_2) = \sqrt{\Sigma_c \Sigma_x \Sigma_y |P_1(x, y, c) - P_2(x, y, c)|^2}$
    - ✓ Very sensitive to lightning, slight changes in light condition causes enormous changes in the distance.

Red Channel | Green Channel | Blue Channel

Red Channel

91	30	33
98	27	36
89	91	92

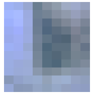
Green Channel

112	42	42
118	40	43
108	106	108

Blue Channel

133	54	51
143	58	61
138	134	133

$D = 404.61$!!!!



Red Channel

169	103	146
157	78	121
145	145	148

Green Channel

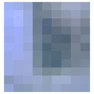
187	121	153
173	103	132
161	160	162

Blue Channel

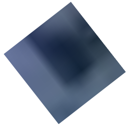
107	40	82
96	23	52
81	80	85

Raw pixel values

- Maybe L2 distance between raw pixel values??
  - $D(P_1, P_2) = \sqrt{\Sigma_c \Sigma_x \Sigma_y |\mathrm{P}_1(\mathrm{x}, \mathrm{y}, \mathrm{c}) \; - \mathrm{P}_2(\mathrm{x}, \mathrm{y}, c)|^2}$
    - ✓ Very sensitive to lightning, slight changes in light condition causes enormous changes in the distance.
    - ✓ Not only the light condition, but also rotation, shift, scale (zoom in, zoom out) changes the pixel values.



Lightning



Rotation

Scale
(Zoom in)

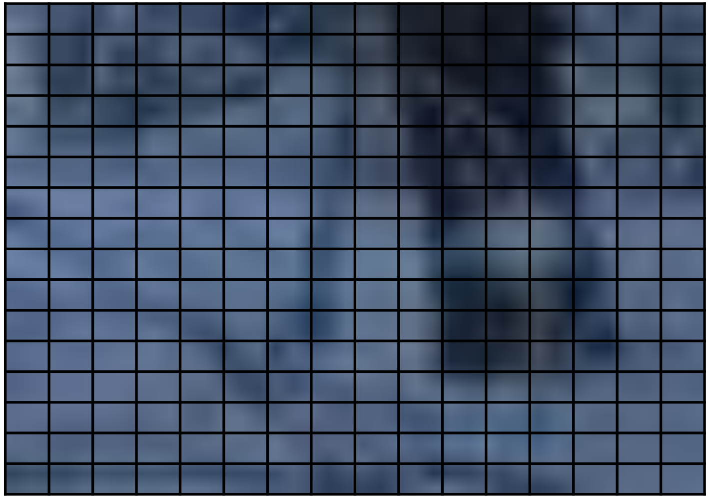
Shift

HOG: Histogram of Oriented Gradients

- HOG was proposed by Dalal and Triggs in 2005.
- It is a better image descriptor than raw pixel values.
- HOG requires the following steps:
  - Computing the gradients
  - Bin the gradients
  - Aggregating the blocks (4x4, 16x16 cells)
  - Normalize gradient magnitudes
- HOG is not reliant on magnitude, it depends just on direction
  - Invariant to some lighting changes
- Dalal and Triggs trained a SVM classifier over these HOG features to detect pedestrians.

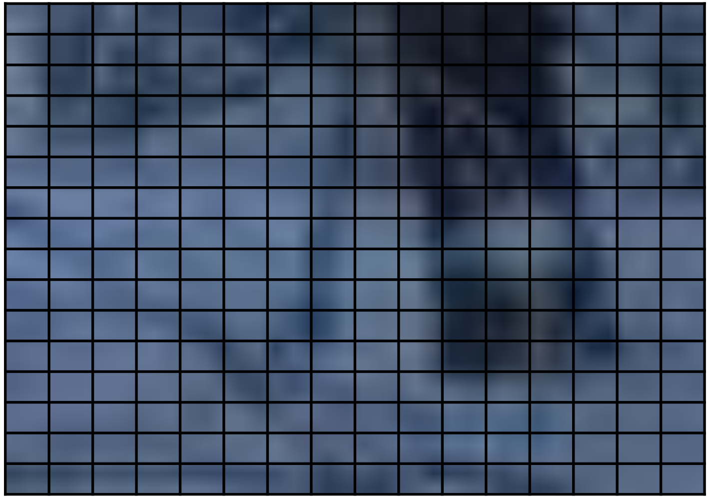
HOG: Histogram of Oriented Gradients

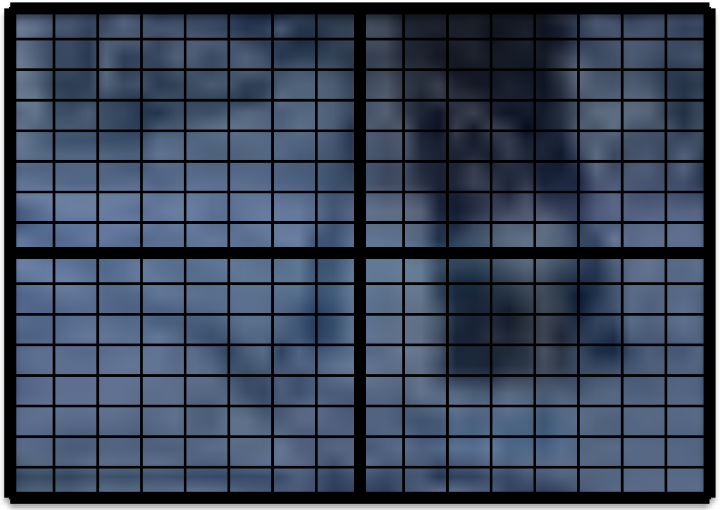
- Pick a neighborhood around your point of interest
  - In this case, a 16x16 neighborhood was chosen.



HOG: Histogram of Oriented Gradients

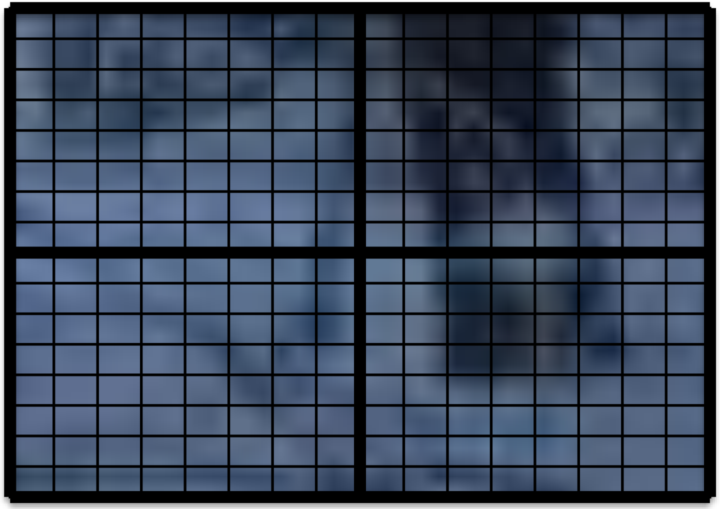
- Pick a neighborhood around your point of interest
  - In this case, a 16x16 neighborhood was chosen.
- Divide the chosen region into smaller areas
  - In this case, we chose 4 sub cells, each of them is 8x8.

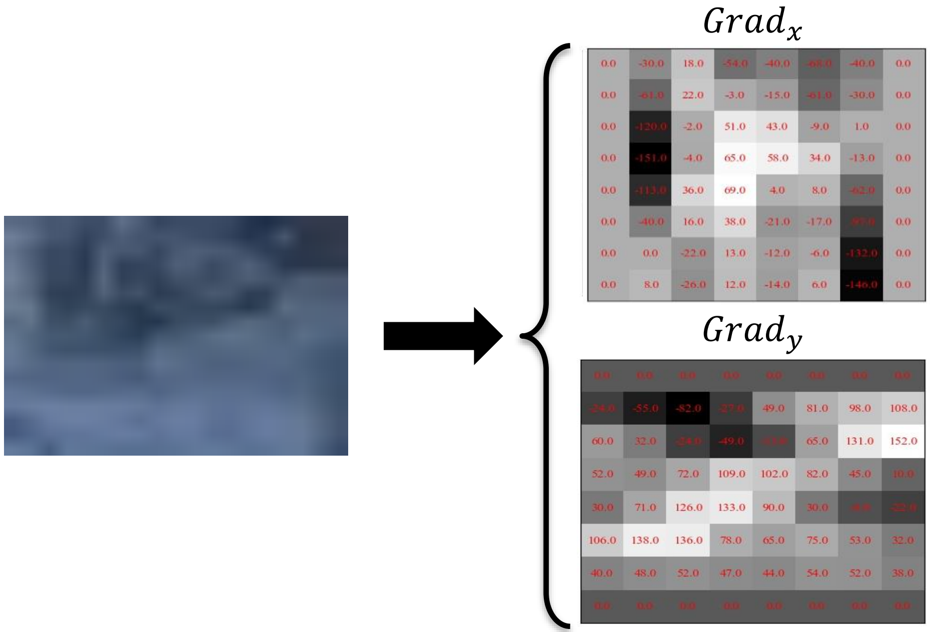




Computing the gradients

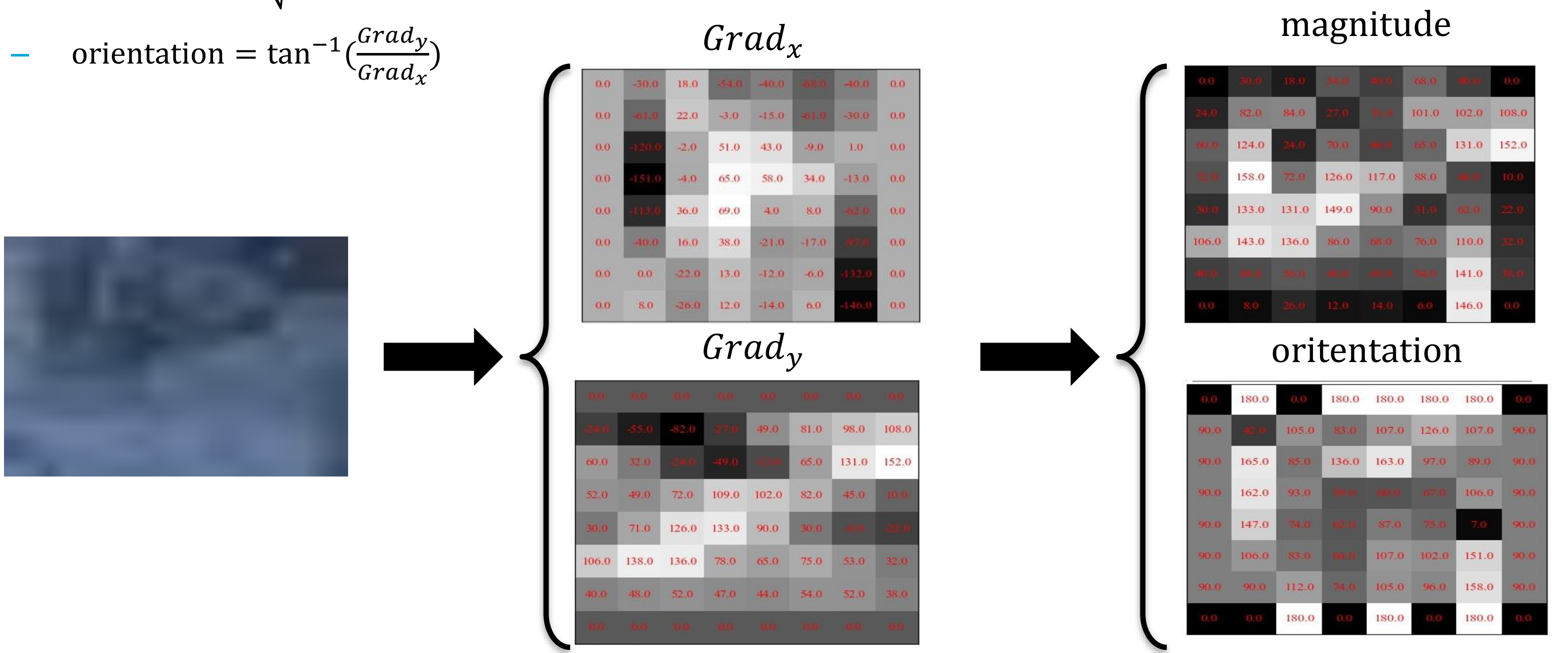
- For each of these cells, calculate the gradient in X and Y directions
  - Simply convolve the image with SobelX and SobelY filters.





Computing the gradients

- Having $Grad_x$ and $Grad_y$, we can calculate the gradient magnitude and orientation in each location:
  - magnitude $= \sqrt{Grad_x^2 + Grad_y^2}$
  - orientation $= \tan^{-1}(\frac{Grad_y}{Grad_x})$



Bin the gradients

- Given magnitude and orientation of gradient at each location, can you guess what is the most frequent orientation for this region?

magnitude

0.0	30.0	18.0	34.0	40.0	68.0	40.0	0.0
24.0	82.0	84.0	27.0	51.0	101.0	102.0	108.0
60.0	124.0	24.0	70.0	44.0	65.0	131.0	152.0
52.0	158.0	72.0	126.0	117.0	88.0	46.0	10.0
30.0	133.0	131.0	149.0	90.0	31.0	62.0	22.0
106.0	143.0	136.0	86.0	68.0	76.0	110.0	32.0
40.0	48.0	56.0	48.0	48.0	54.0	141.0	38.0
0.0	8.0	26.0	12.0	14.0	6.0	146.0	0.0

oritentation

0.0	180.0	0.0	180.0	180.0	180.0	180.0	0.0
90.0	42.0	105.0	83.0	107.0	126.0	107.0	90.0
90.0	165.0	85.0	136.0	163.0	97.0	89.0	90.0
90.0	162.0	93.0	99.0	60.0	67.0	106.0	90.0
90.0	147.0	74.0	62.0	87.0	75.0	7.0	90.0
90.0	106.0	83.0	94.0	107.0	102.0	151.0	90.0
90.0	90.0	112.0	74.0	105.0	96.0	158.0	90.0
0.0	0.0	180.0	0.0	180.0	0.0	180.0	0.0

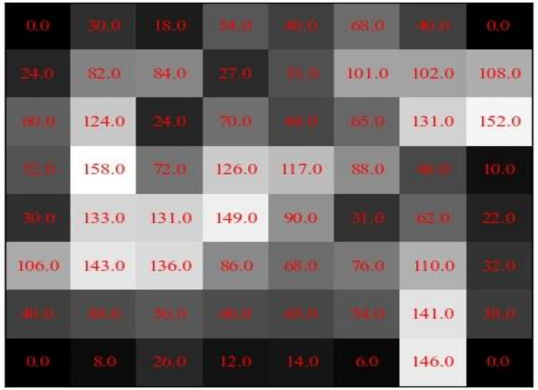
Bin the gradients

- Find the histogram of orientations with 9 bins $(0 - 180°)$

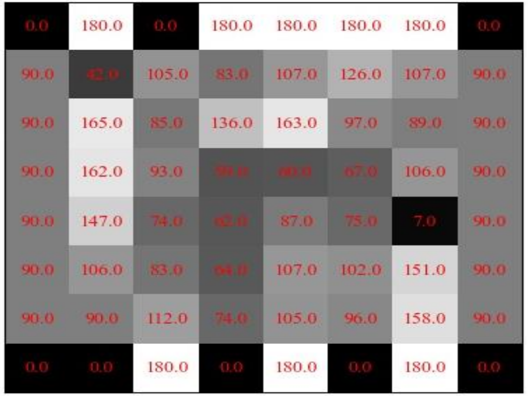
Of course we can use more bins or choose a bigger range. The $(0 - 180)$ was suggested in the original paper

image patch

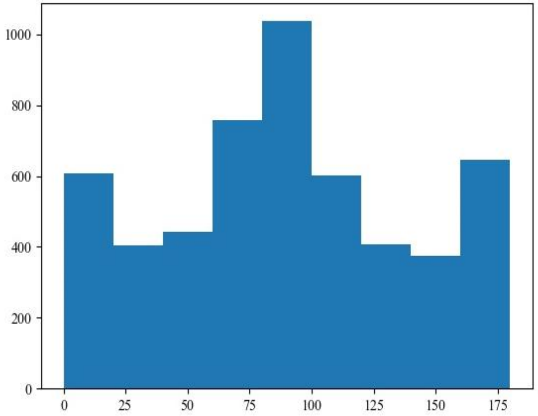
magnitude



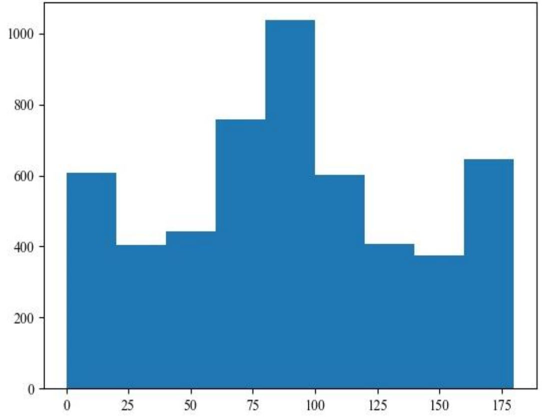
oritentation



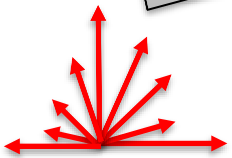
Histogram of Oriented Gradients



Bin the gradients



It's another way to visualize the HOG features. The angle of each vector shows orientation of gradients, the magnitude of the vector shows the gradient magnitude.



Each element of this vector shows frequency of the corresponding bin (for example the 1st bin frequency is 607)

HOG Feature descriptor

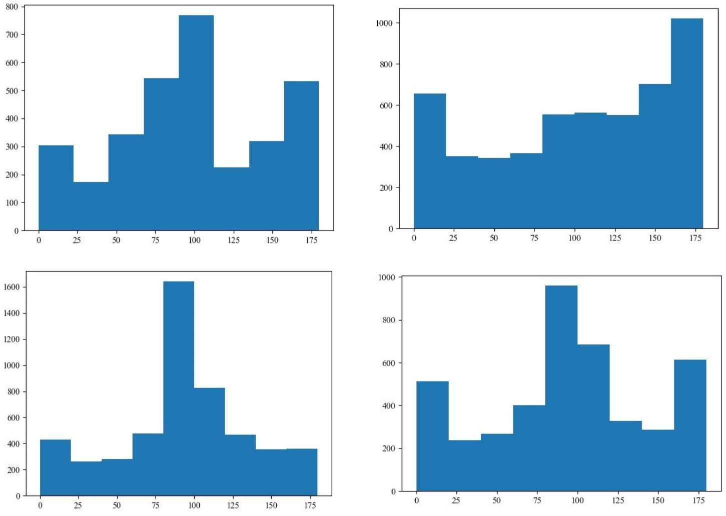
$$[\,607, 403, 444, 758, 1037, 601, 407, 374, 645]$$

Aggregating the blocks

- We can repeat the process for each of those four cells.

Aggregating the blocks

- Each cell has a 9×1 feature vector. We can combine these vectors to make a 36×1 feature vector.



HOG feature discriptor

$$\begin{bmatrix} 0.33 \\ 0.22 \\ \vdots \\ 0.45 \\ 0.11 \end{bmatrix}_{36 \times 1}$$

Normalization reduces the sensitivity to the lightning condition. We don’t care for the actual gradient magnitudes, we only care about how

HOG in action!

HOG

Feature descriptor 1 (FD_1): $\begin{bmatrix} 0.33 \\ 0.22 \\ \vdots \\ 0.45 \\ 0.11 \end{bmatrix}_{36\times1}$

L2 distance $= 0.62$

HOG

Feature descriptor 2 (FD_2): $\begin{bmatrix} 0.30 \\ 0.24 \\ \vdots \\ 0.42 \\ 0.13 \end{bmatrix}_{36\times1}$

How about lightning?

HOG

Feature descriptor 1 (FD_1): $\begin{bmatrix} 0.33 \\ 0.22 \\ \vdots \\ 0.45 \\ 0.11 \end{bmatrix}_{36\times 1}$

L2 distance $= 0.61$

HOG

Feature descriptor 2 (FD_2): $\begin{bmatrix} 0.33 \\ 0.21 \\ \vdots \\ 0.43 \\ 0.08 \end{bmatrix}_{36\times 1}$

Although the light condition was considerably changed, the L2 distance did not!

Rotation?

There is a considerable increment in L2 distance.

HOG

Feature descriptor 1 (FD_1): $\begin{bmatrix} 0.33 \\ 0.22 \\ . \\ . \\ . \\ 0.45 \\ 0.11 \end{bmatrix}_{36\times 1}$

HOG

Feature descriptor 2 (FD_2): $\begin{bmatrix} 0.15 \\ 0.35 \\ . \\ . \\ . \\ 0.15 \\ 0.18 \end{bmatrix}_{36\times 1}$

L2 distance $= 0.71$

HOG is sensitive to the orientation

- In HOG, we keep track of the orientations of the gradients.
- In other words, orientation plays a key role in the extracted features.
- This makes the HOG sensitive to the changes in orientations.

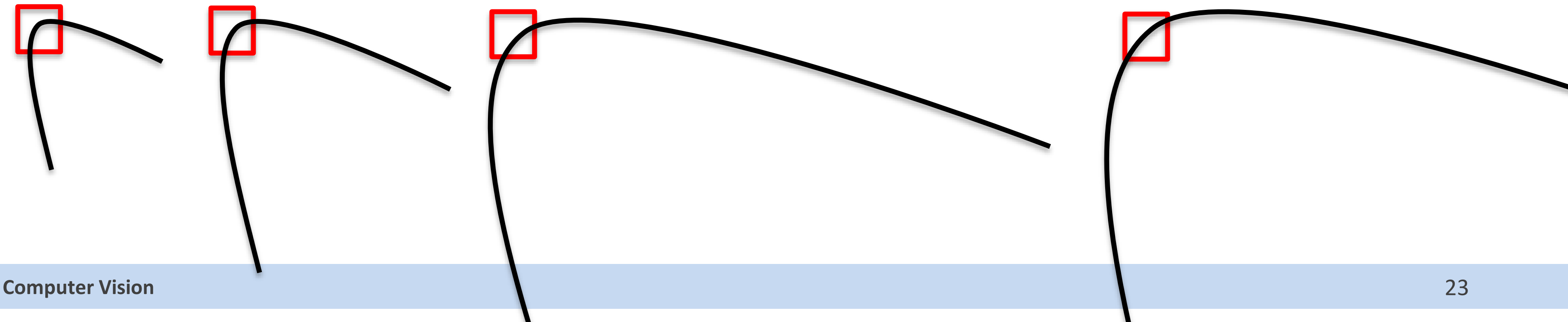
Orientation is not the only drawback

- Do you remember how did we find key points in the image?
  - We used Harris corner detector to find the corners in the image!

- Then we extracted HOG features around these corner points.

- Harris corner detector is sensitive to the scale!

As the size of the object increases, the corner detector can not detect the corner properly. Harris corner detector is sensitive to the changes in scale (zoom in, zoom out)



Want features and keypoints invariant to scaling, rotation, etc.

- Scale Invariant Feature Transform (SIFT)
  - Lowe et al. 2004, many images from that paper
- Find the scale-invariant response map
- Find keypoints
- Extract rotation-invariant descriptors
  - Normalize based on orientation
  - Normalize based on lighting

Scale-invariant response map

- Blur the input image with gaussian kernel with σ
- Take the blurred image and convolve it with a gaussian kernel with $s^1\sigma$.
- Repeat this procedure until you complete an octave.

scale

(1^{st} octave)

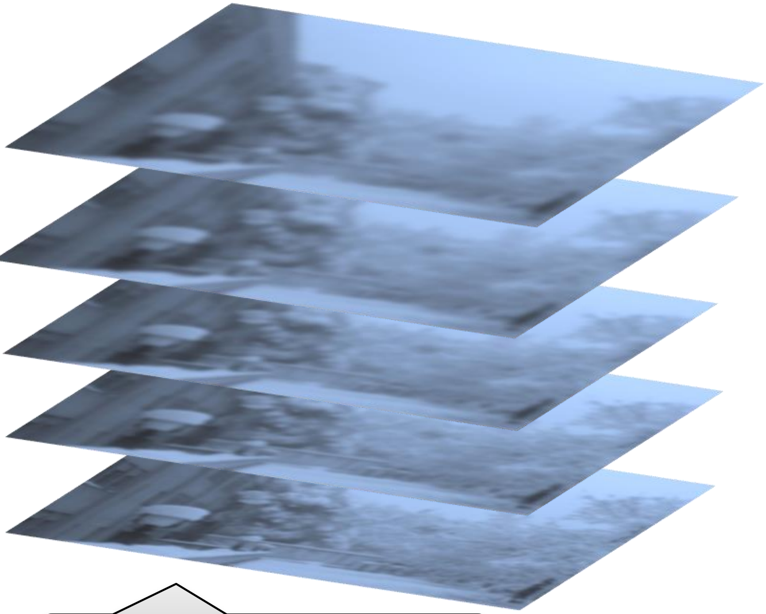
$s^4.\sigma$

$s^3.\sigma$

$s^2.\sigma$

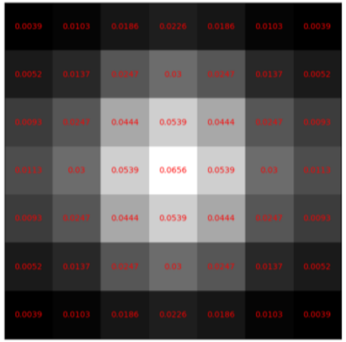
$s^1.\sigma$

$s^0.\sigma$



This is called "Gaussian Pyramid".

Gaussian kernel at scale 0. In this case, σ and s were 1.6 and $2^{\frac{1}{4}}$, respectively.



scale

(1^{st} octave)

$s^4.\sigma$

$s^3.\sigma$

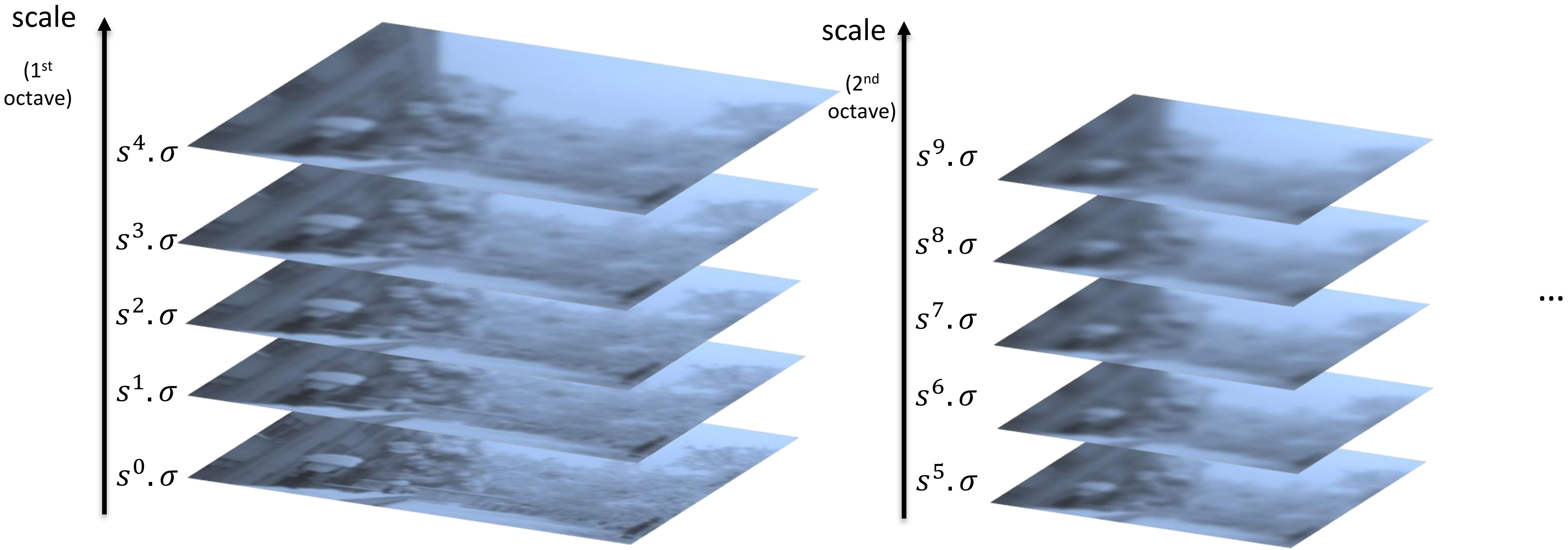
$s^2.\sigma$

$s^1.\sigma$

$s^0.\sigma$

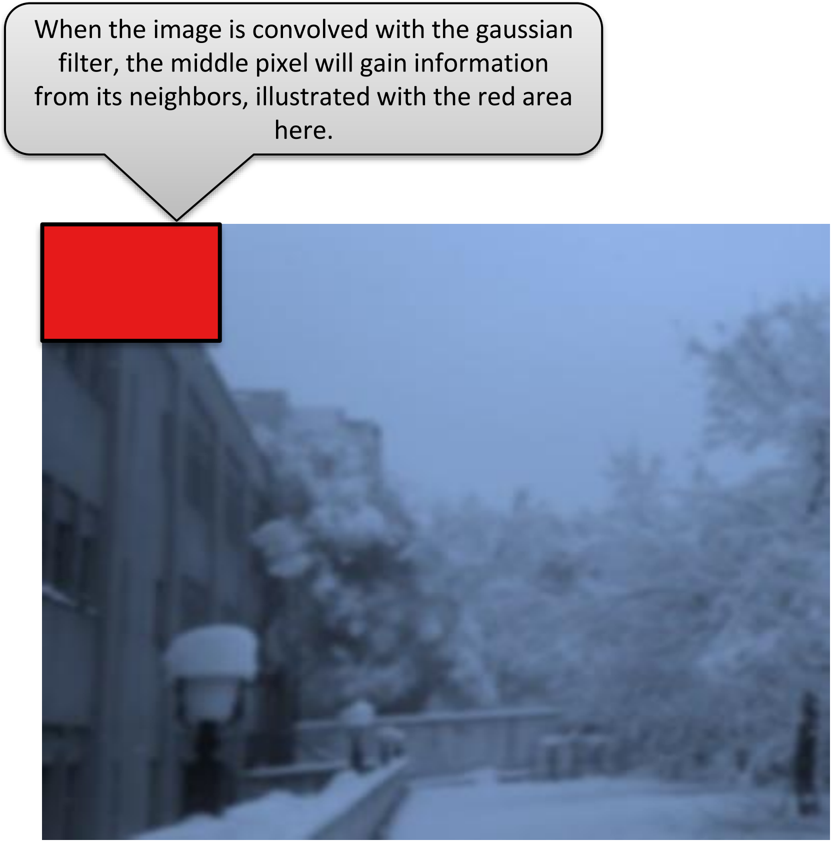
Scale-invariant response map

- When the first octave is finished, resize the last blurred image to half its size and complete the next octave the same as before.



What is happening?

- What is going on with the scale?



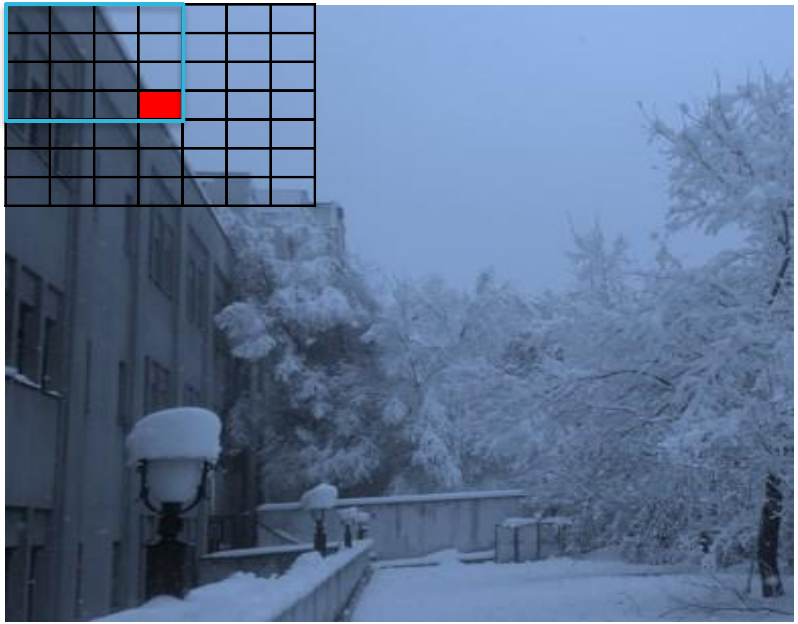
What is happening?

- What is going on with the scale?

What is happening?

- What is going on with the scale?

The last neighbors will have information about this large area.



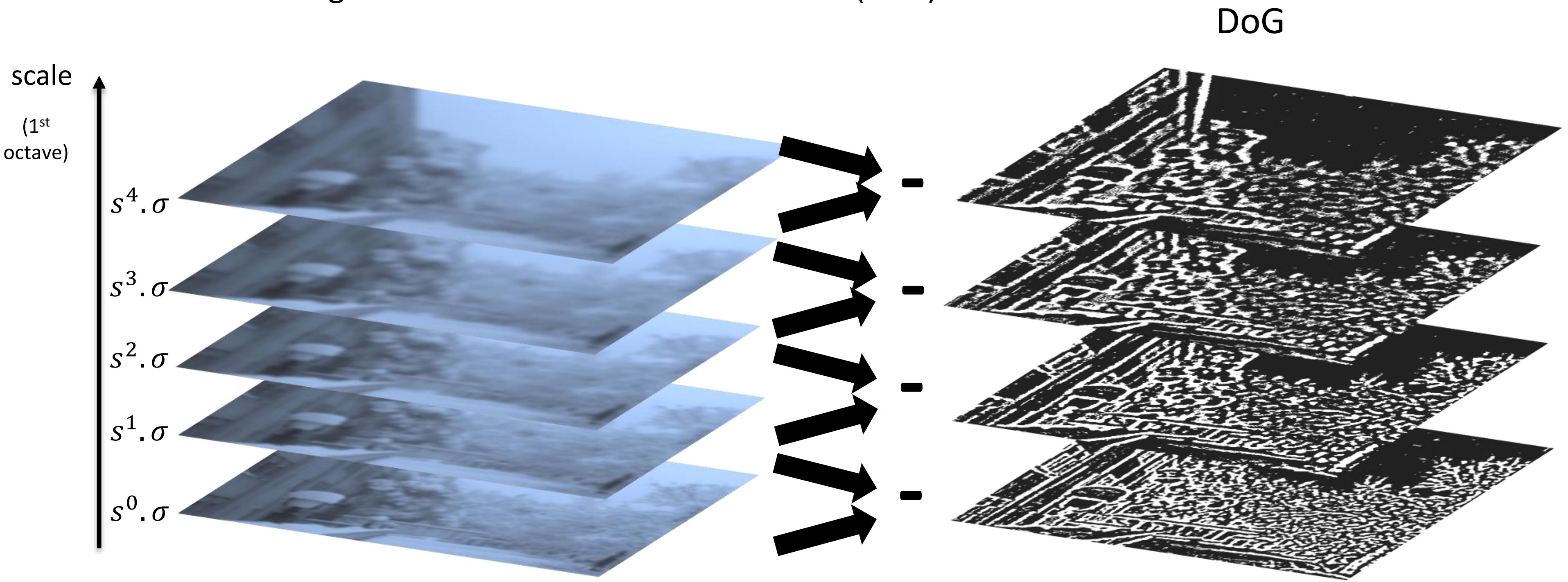
What is happening?

- What is going on with the scale?

When we move to the scale 2, the blurred image from the scale 1 will be used here. Now, the middle pixel would have access to the information from its neighbors and will have information from a larger area.

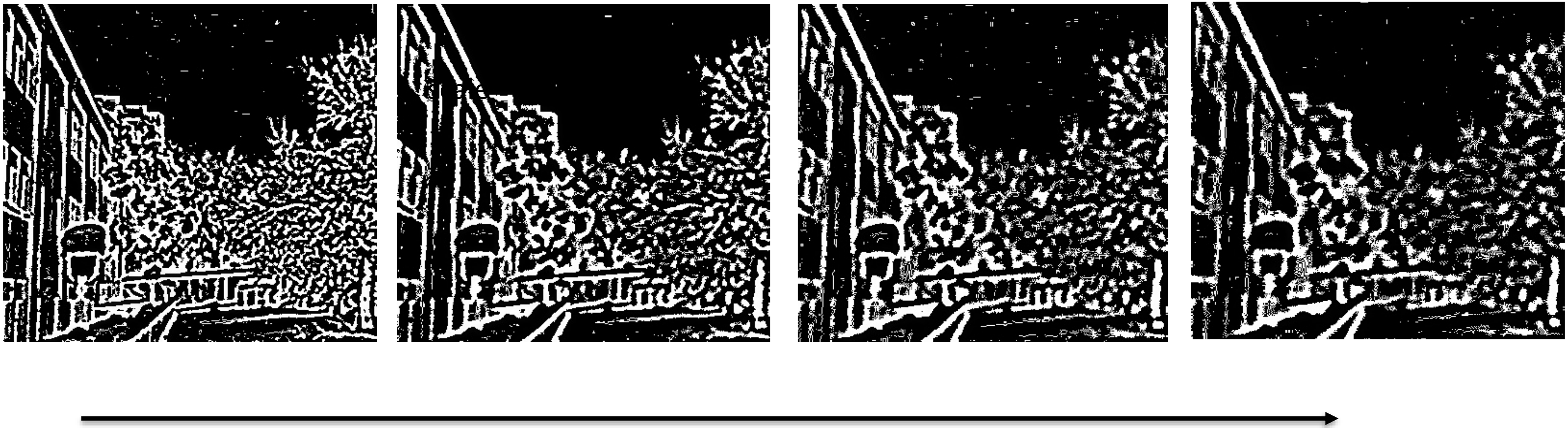
Extract DoG features at multiple scales

- Subtract each of two consecutive gaussian filtered images.
- The result images are called Difference of Gaussian (DoG).



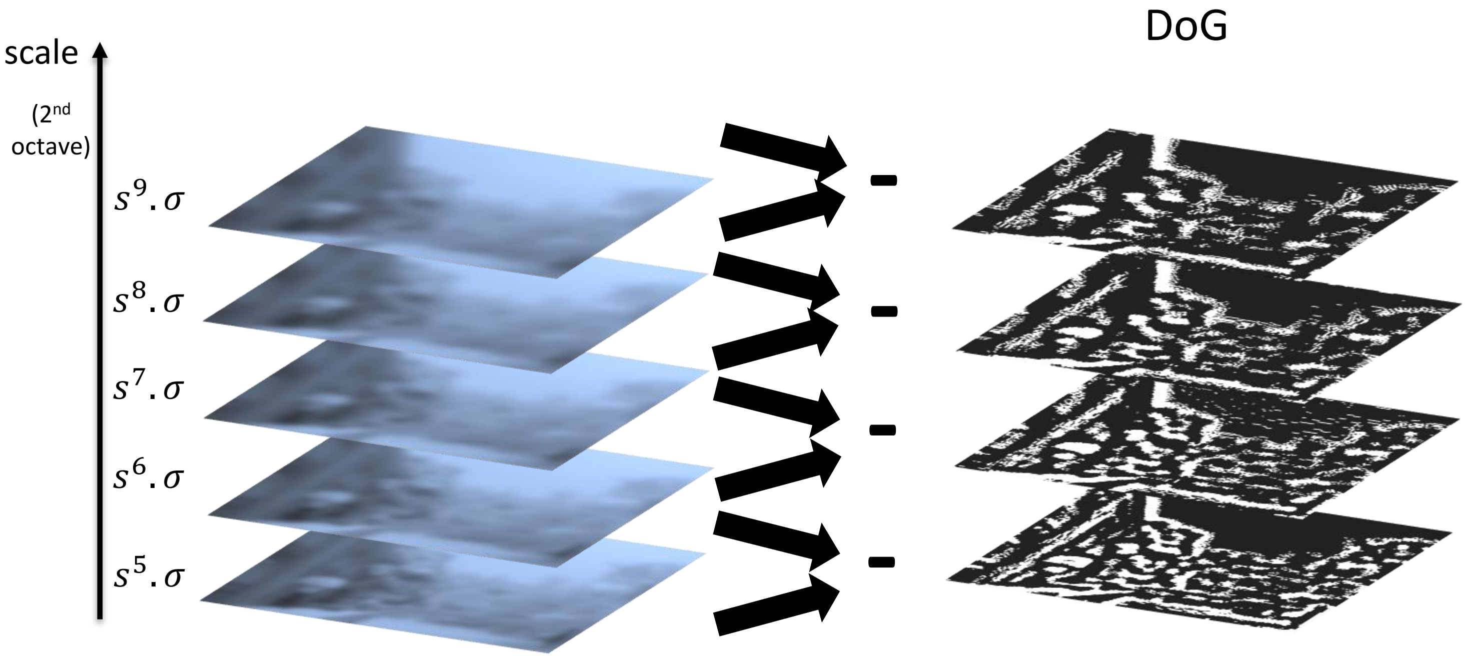
Extract DoG features at multiple scales

- It seems that DoG extracts edges!
- The DoG extracts thicker edges and neglects the thinner ones as we go to the upper scales.



Extract DoG features at multiple scales

- Don't forget the other octaves!



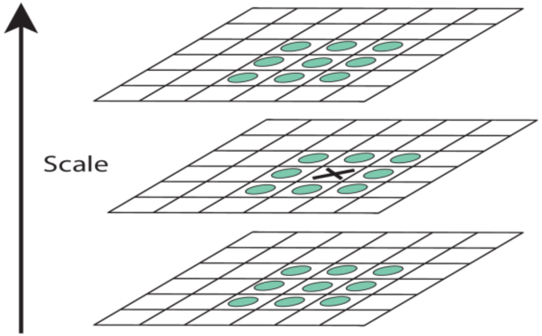
Extract DoG features at multiple scales

DoG acts like a bandpass filter. You can see noise in this image but not in the other ones. These noises' frequency was in the bandpass for this scale.

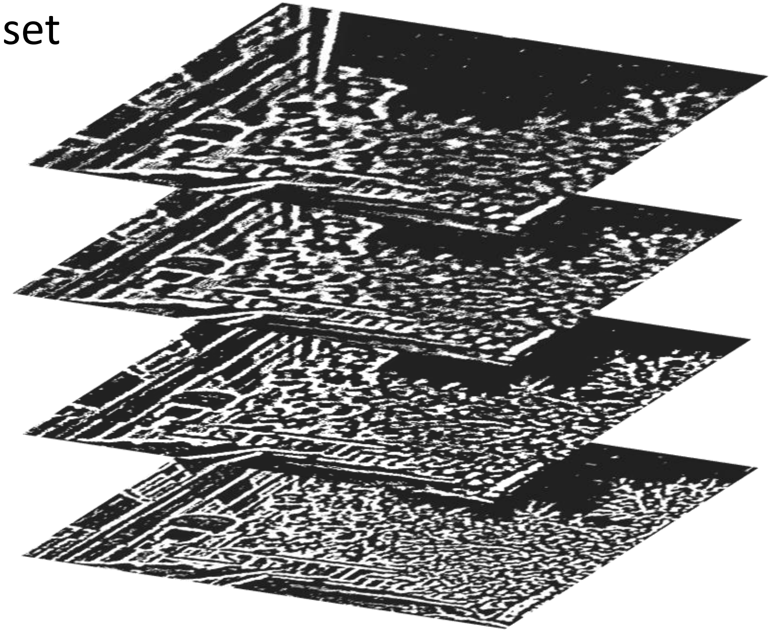
Find local-maxima in location and scale

- For each pixel, compare it to its neighbor from the current scale, and the neighbors from upper and lower scales.
- If that point is local maxima, keep that; otherwise, set that to zero.

Each point would have 26 neighbors:
8 neighbors from the current scale, 9 neighbors from the lower scale, and 9 neighbors from the upper scale.



DoG



Throw out weak responses and edges

- For each DoG image, calculate the 2nd derivative in x direction ($Grad_{xx}$), 2nd derivative in y direction ($Grad_{yy}$) and derivative in x-y direction ($Grad_{xy}$).

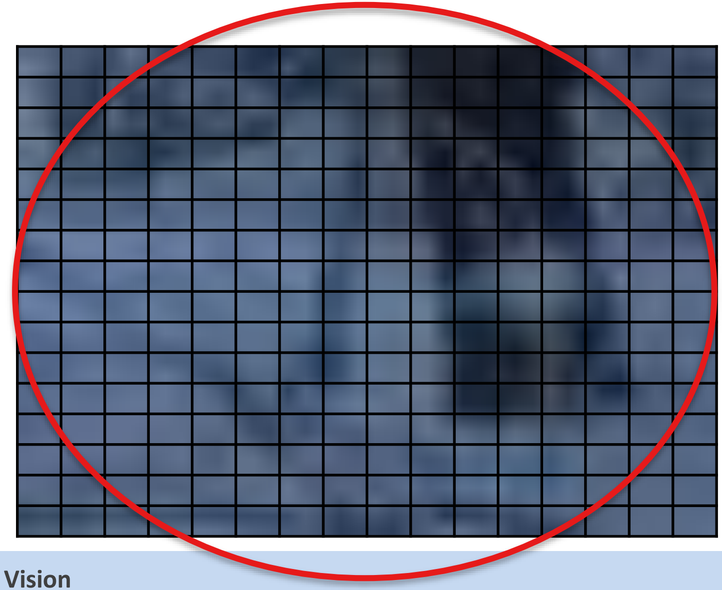
- For each pixel on DoG images make the Hessian matrix:
  - $H = \begin{bmatrix} Grad_{xx} & Grad_{xy} \\ Grad_{xy} & Grad_{yy} \end{bmatrix}$
    - ✓ If both eigenvalues of H are large, there are lots of things going on at the location (important point)
      - ➢ If both eigenvalues are high, we probably have corner or blob at that location.
      - ➢ If both eigenvalues are small, there is nothing at that location.
      - ➢ If one of the eigenvalues are large and the other one is small, it is probably an edge (changes in one direction)

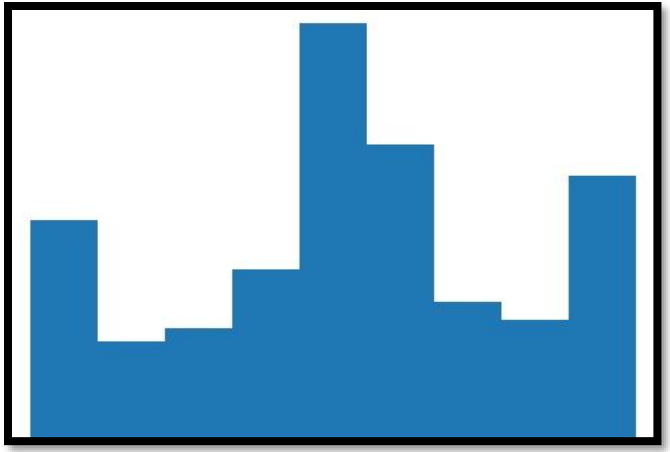
Throw out weak responses and edges

- $H = \begin{bmatrix} Grad_{xx} & Grad_{xy} \\ Grad_{xy} & Grad_{yy} \end{bmatrix}$
  - If both eigenvalues of H are large, there are lots of things going on at the location (important point)
    - ✓ If both eigenvalues are high, we probably have corner or blob at that location.
    - ✓ If both eigenvalues are small, there is nothing at that location.
    - ✓ If one of the eigenvalues are large and the other one is small, it is probably an edge (changes in one direction)
- Same as Harris corner detector, we do not' need to calculate the eigenvalues directly. Instead, we can estimate them with Trace and Determinant:
  - $\frac{Trace(H)}{Det(H)} < \frac{(r+1)^2}{r}$
    - ✓ Where r is the desired ratio of the larger eigenvalue to the smaller one. If $\frac{Trace(H)}{Det(H)} < \frac{(r+1)^2}{r}$, throw out the point. It probably does not have large eigenvalues.

Find main orientation of patches

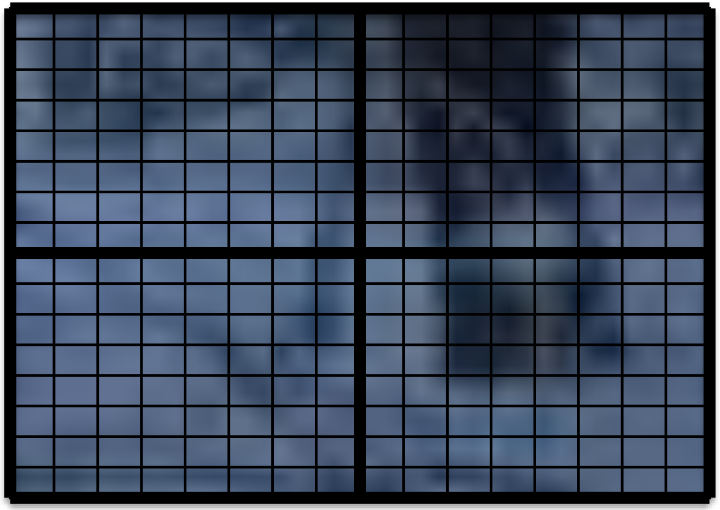
- Look at weighted histogram of nearby gradients
  - The gradient at the middle receive more weights than the gradients far from the middle.
  - Any gradient within 80% of peak gets its own descriptor
    - ✓ So, we may end up with multiple keypoints per pixel
- Descriptors are normalized based on main orientation





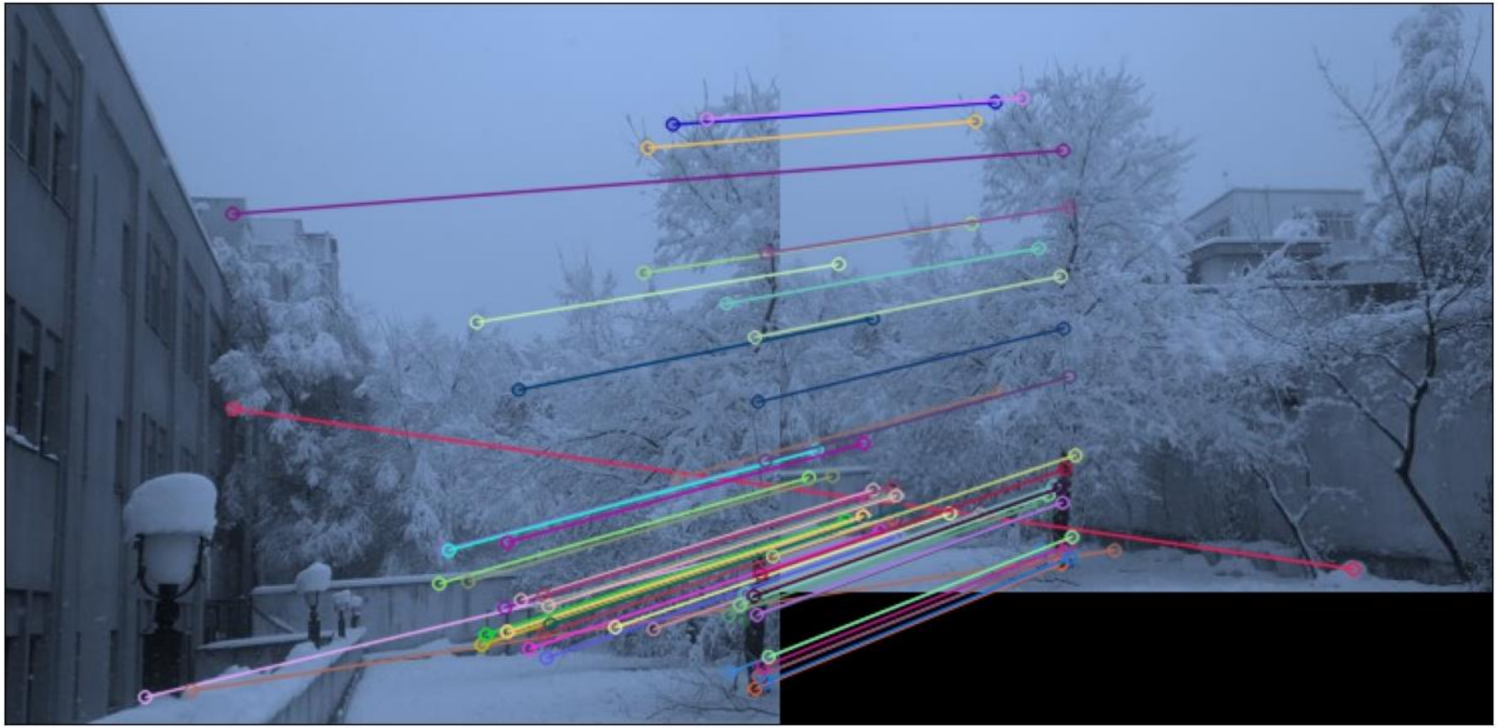
Keypoints are normalized gradient histograms

- Divide into subwindows (2x2, 4x4)
- Bin gradients within subwindow, get histogram
  - Normalize to unit length
  - Clamp at maximum .2
  - Normalize again
  - Helps with lighting changes!



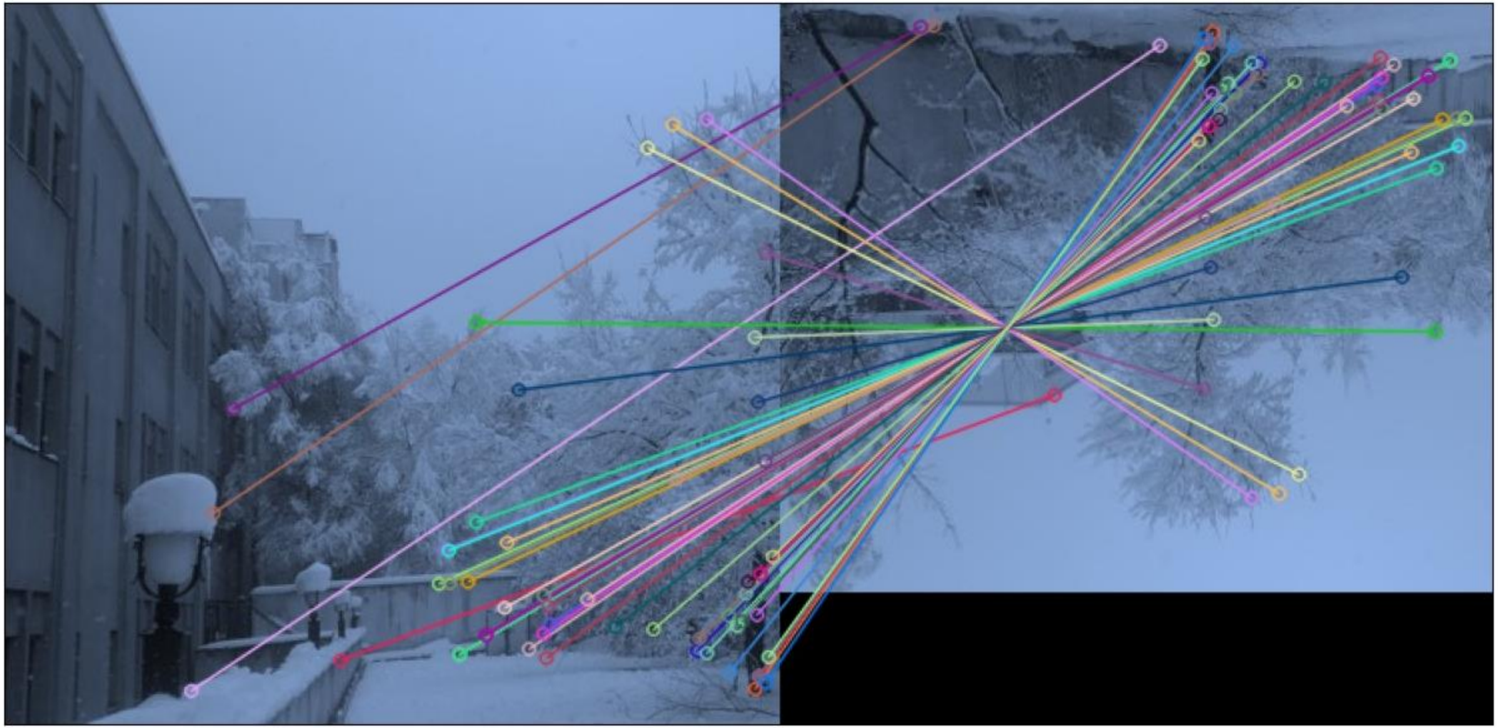
SIFT in action!

- Matching with different scales



SIFT in action!

- Matching with different scales and orientations



SIFT in action!

- Matching with different scales, orientations, and lightning!