컴퓨터 비전 세미나

12191584 김창수

Index

Local feature detection
- SIFT
- SURF

Feature descriptor
- SIFT descriptor
- HOG descriptor
- Image matching

Scale

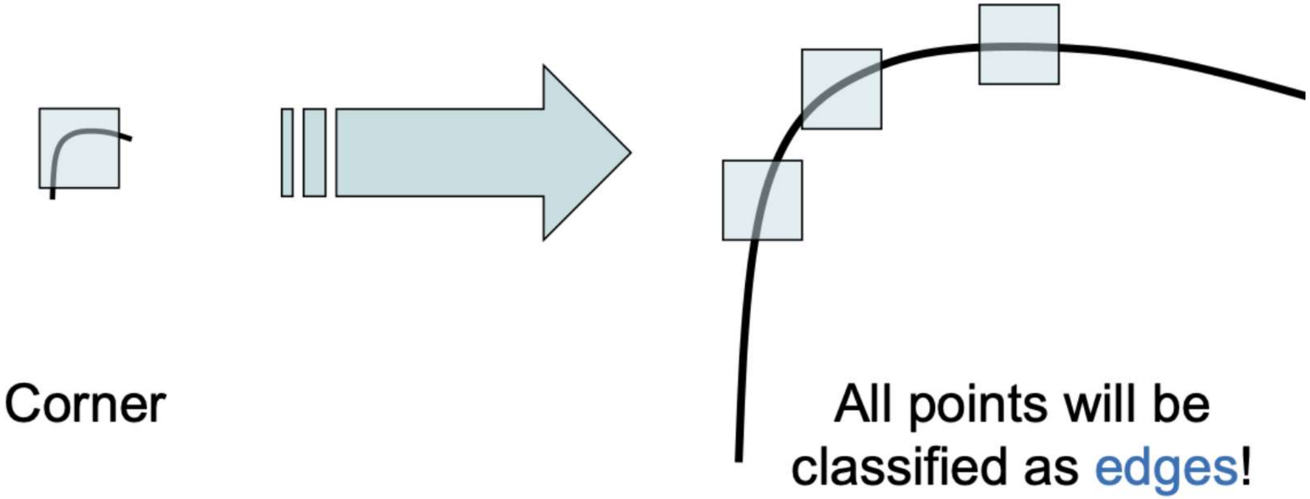
The Scale Image command enlarges or reduces the physical size of the image by changing the number of pixels it contains.

Image features can appear at different scales.

Scale

Scale invariance feature

: feature that remain invariant when the local scale in the image is changed.



Not invariant to image scale!

SIFT

When you have images of different scales and rotations, you need to use the Scale Invariant Feature Transform.

Algorithm

1. Constructing a scale space
2. LoG Approximation
3. Finding keypoints
4. Generate SIFT features

Constructing a scale space

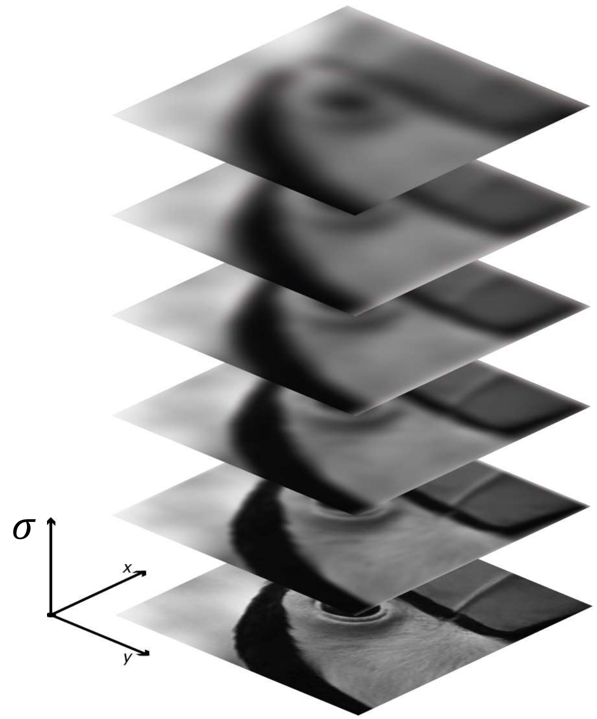
If you want to detect features of different sizes, you achieve this by "zooming out the image" with the various scales of the pyramid.

Constructing a scale space

Construct the scale space of an image by convolving a Gaussian kernel(Blurring) at different scales with the input image.

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

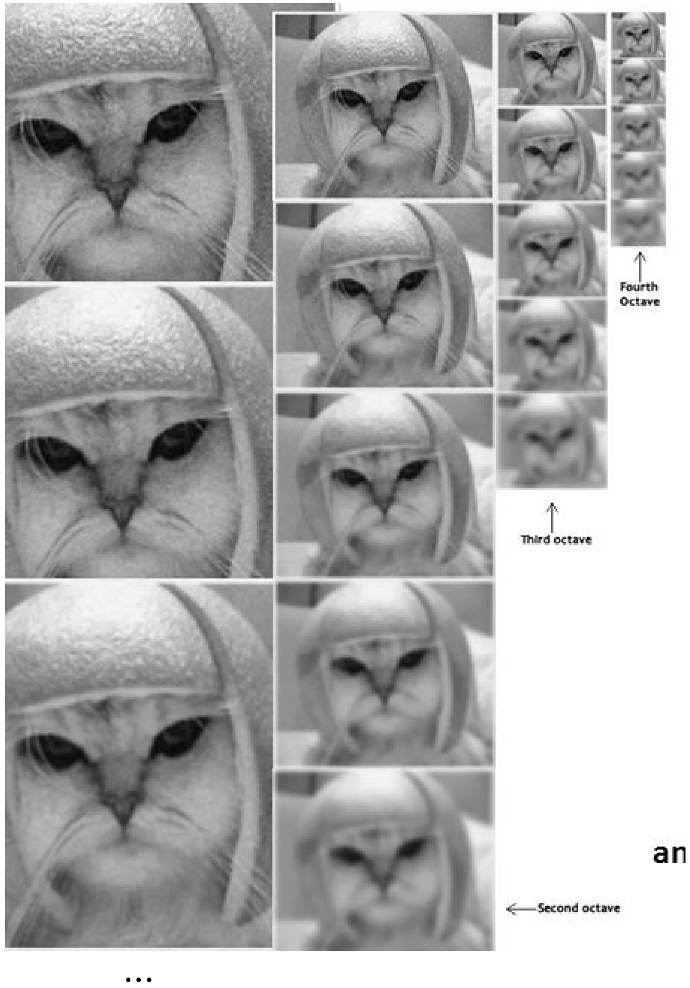
σ determines the width of the Gaussian kernel.



Constructing a scale space

Scale-space is separated into octaves and
the number of octaves and scale
depends on the size of the original image.

Each octave's image size is half the previous one.

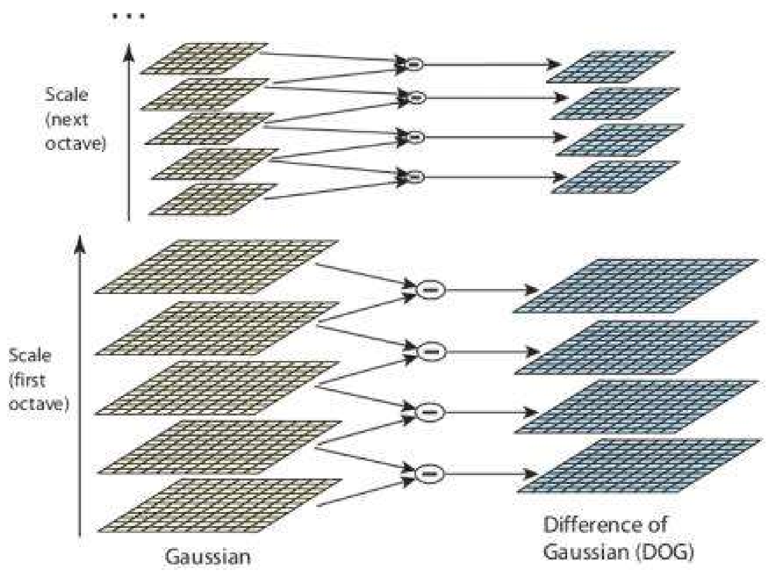


LoG Approximation

Maxima and minima of Laplacian of Gaussian produce the stable image features.

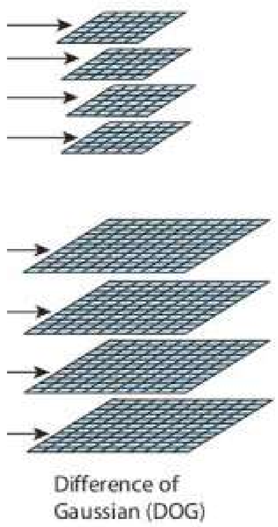
These features are good for finding keypoints.

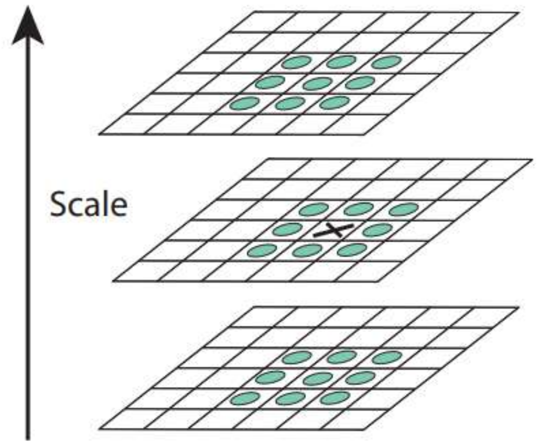
The problem is that LoG is computationally intensive so we use Difference of Gaussians which is an approximation of LoG.



Finding keypoints

Find the maxima and minima of Difference of Gaussian(DoG) pyramids.





X is marked as a "key point" if it is the greatest or least of all 26 neighbors.

Generate SIFT features

Get rid of some unstable features:

- Interpolation of nearby data for accurate position
- Removing low contrast features
- Removing edges

What remains is strong interest points, coordinates with scale $(x,\ y,\ \sigma)$

SURF

It is partly inspired by the SIFT descriptor.

The standard version of SURF is several times faster more robust than SIFT.

Instead of constructing DoG pyramid, SURF uses the Hessian matrix(second derivative) because of its good performance in computation time and accuracy.

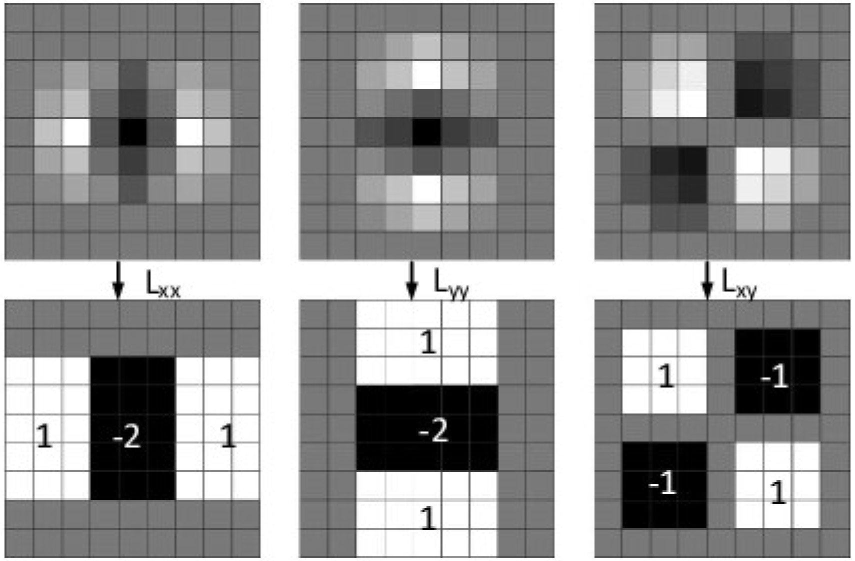
For adapt to any scale, the Hessian matrix H(x, σ) in x at scale σ is defined as:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

SURF

In order to calculate the determinant of the Hessian matrix, first we need to apply convolution with Gaussian kernel, then second-order derivative.

SURF approximates Hessian with box filter.



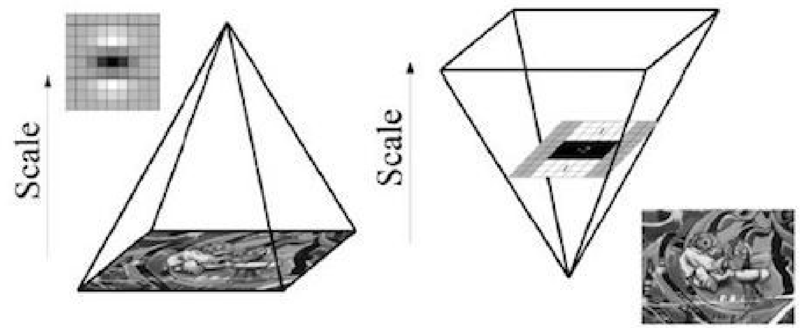
SURF

Convolution with box filters can be computed with 3 integer operations using a precomputed integral image.

The scale space is analyzed by up-scaling the filter size

(9×9 → 15×15 → 21×21 → 27×27, etc)

rather than iteratively reducing the image size.



And find keypoints in the image over scales as SIFT does.

Feature descriptor

A feature detector is an algorithm which takes an image and outputs coordinates.

A feature descriptor is an algorithm which takes an image and outputs feature vectors.

Feature descriptors encode interesting information into a series of numbers that can be used to differentiate one feature from another.

Instead of just using an $(x,\ y)$ pair as a location in “image space”,
you might have a triple $(x,\ y,\ \text{feature vector})$ as location in “scale space”.

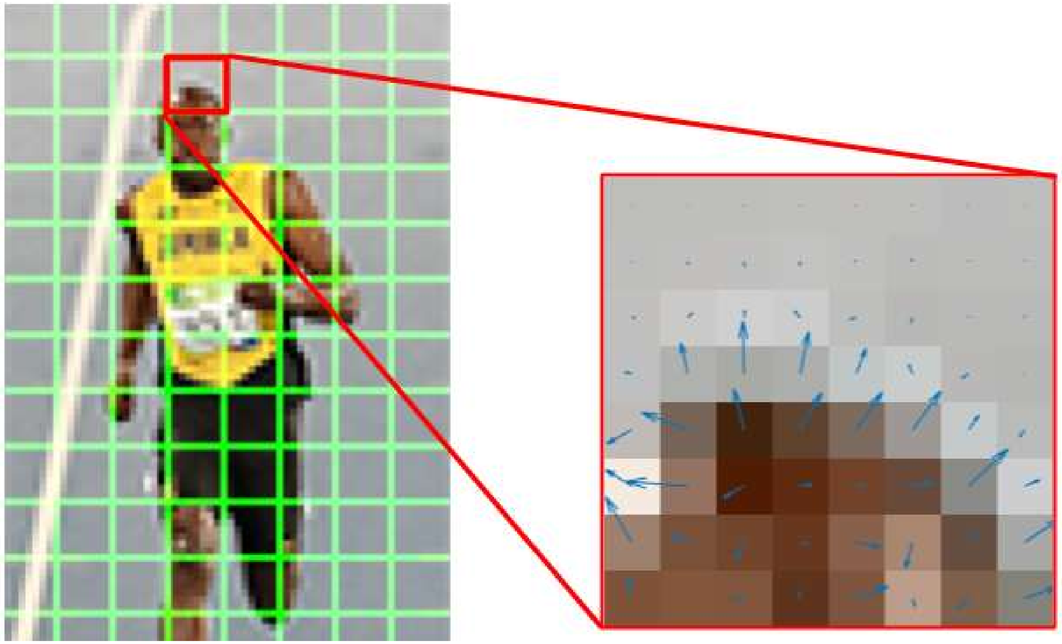
HOG descriptor

1. Divide the image into small connected regions called cells.

2. For each cell, compute a histogram of gradient directions or edge orientations for the pixels within the cell.

$$g = \sqrt{g_x^2 + g_y^2}$$

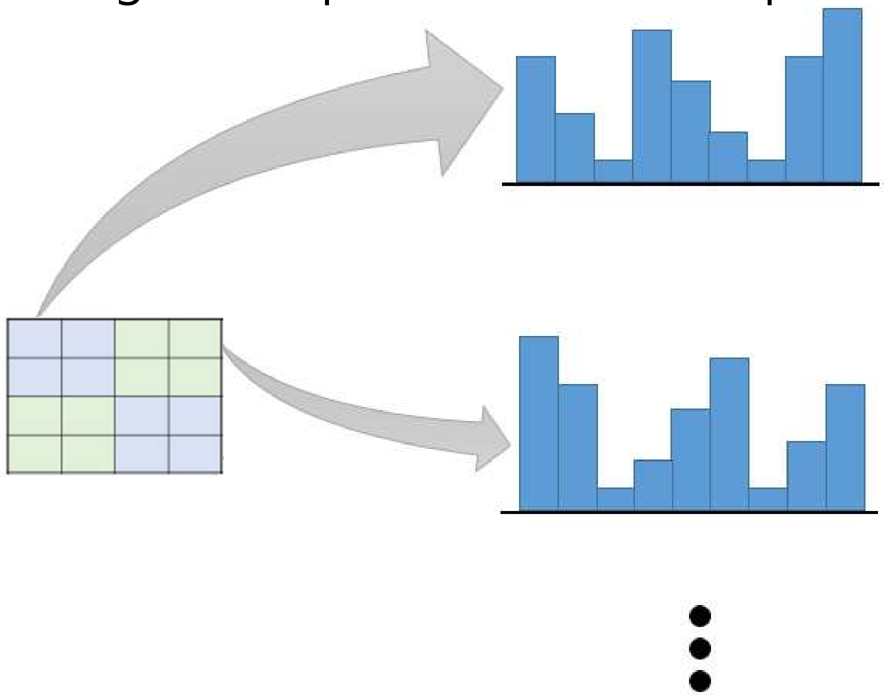
$$\theta = \arctan \frac{g_y}{g_x}$$



HOG descriptor

3. Groups of adjacent cells are called blocks. The grouping of cells into a block is the basis for grouping and normalization of histograms.

4. The set of these block histograms represents the descriptor.



feature vector: | 4 | 1 | 0 | 5 | 6 | 7 | 4 | 3 | 2 | 2 | 5 | 1 | 2 | 6 | 3 | 5 | 7 | 1 | 8 | 4 | 6 | 2 | 1 | 3 | 0 | 4 | 4 | 1 | 4 | 0 | 3 | 4 | 7 | 8 | 4 | 1 |

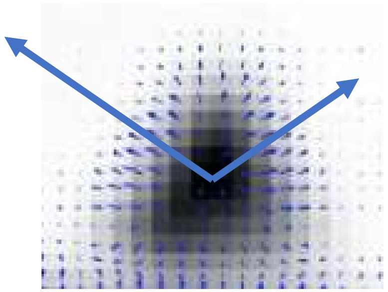
SIFT descriptor

We know the scale at which the keypoint was detected, So we have scale invariance.
$(x,\ y,\ \sigma)$

The next thing is to assign an orientation to each keypoint to make it rotation invariance.

Calculate directions and magnitudes around each keypoint, and figure out the most prominent orientations in that region.

It creates keypoints with same location and scale, but different directions. $(x,\ y,\ \sigma,\ \theta_i)$



SIFT descriptor

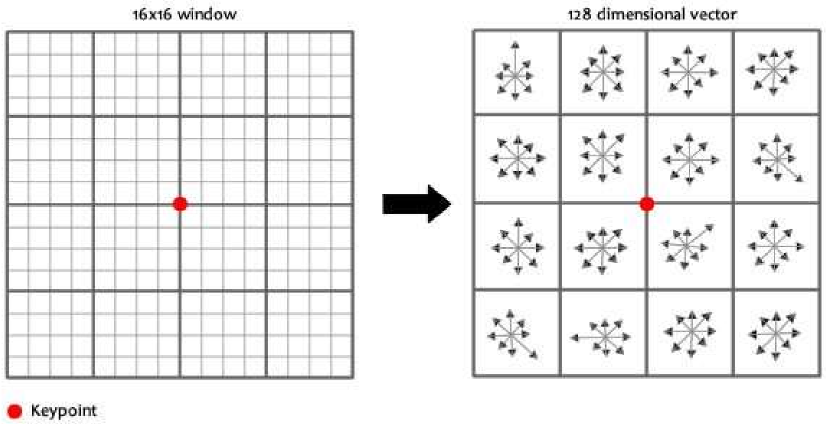
At this point, each keypoint has a location, scale, orientation. $(x,\ y,\ \sigma,\ \theta_i)$

Next is to compute a descriptor about each keypoint that is highly distinctive.

SIFT descriptor

16x16 window around the keypoint is taken and divided into 16 sub-blocks of 4x4 size.

For each sub-block, 8 bin orientation histogram is created.



These 4 x 4 x 8 = 128 numbers form the "feature vector".

This keypoint is uniquely identified by this feature vector $\mathbf{x}$. $(x,\ y,\ \sigma,\ \theta_i,\ \mathbf{x}_i)$