Scale-Invariant Feature Transform

What's the deal with features?

- What are we looking for?
  - Strong features
  - Invariant to changes (affine and perspective/occlusion)
  - Solve the problem of correspondence
    - Locate an object in multiple images (i.e. in video)
    - Track the path of the object, infer the motion

Scale Invariant Feature Transform (SIFT)

- Choosing features that are invariant to image scaling and rotation
- Also, partially invariant to changes in illumination and 3D camera viewpoint
- Well localized in both spatial and frequency domain
  - Resistant to noise, clutter, and occlusion
- Features are highly distinctive, matched with high probability against large number of features

Motivation

- Earlier Methods
  - Harris corner detector
    - Sensitive to changes in image scale
    - Finds locations in image with large gradients in all directions
  - No method was fully affine invariant
    - Although the SIFT approach is not fully invariant it allows for considerable affine change
    - SIFT also allows for changes in 3D viewpoint

SIFT Algorithm Overview

1. Constructing the Scale space
2. Laplacian of Gaussian approximation
   - Difference-of-Gaussian function
3. Keypoint localization
   - Keypoints thresholded for stability
4. Orientation assignment
5. SIFT construction

How do we choose keypoints?

- Looking for features (locations) that are stable (invariant) across all possible scale changes
  - Use a continuous function of scale (scale space)
- Which scale-space kernel will we use?
  - The Gaussian Function

Scale-Space of Image

- Scale-Space of image is generated by a scale of σ for G(x,y,kσ) in one octave. In practice, we double the size of original image to generate the seed of the first octave.
- The next octave ‘s image size is equal a half of the previous one.
- One image *L* in the Scale space :

$$L(\mathrm{x}, \mathrm{y}, \mathrm{k}\sigma) = \mathrm{G}(\mathrm{x}, \mathrm{y}, \mathrm{k}\sigma) * \mathrm{I}(\mathrm{x}, \mathrm{y})$$

$G(x, y, k\sigma)$ - variable-scale Gaussian

$I(x, y)$ - input image

- To detect stable keypoint locations, we find the scale-space extrema in difference-of-Gaussian function

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

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Difference of Gaussian

1. Efficient to compute
   - Smoothed images *L* needed later so *D* can be computed by simple image subtraction
2. Close approximation to scale-normalized Laplacian of Gaussian
   - Required for true scale invariance

Scale-normalized Laplacian of Gaussian $(\sigma^2\nabla^2 G)$

- Need to find a relationship between *D and* $\sigma^2\nabla^2 G$
  - Heat diffusion equation (important function in Partial Differential Equation):

$$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G$$

  - Compute finite difference approximation to $\frac{\partial G}{\partial \sigma}$

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

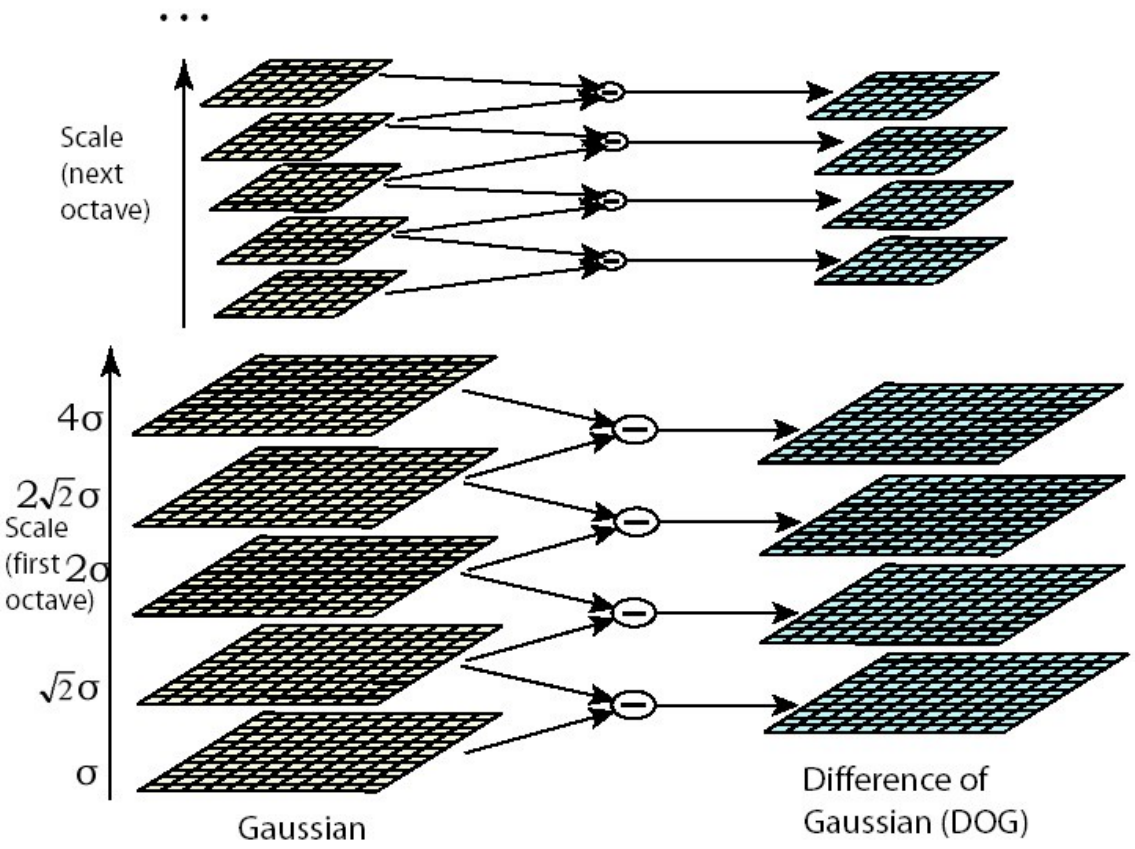
$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\nabla^2 G$$

  - (k-1) term
    - Constant over scales, does not affect keypoint location
    - Approximation error will go to zero as k → 1, but found that does not affect stability of keypoint.

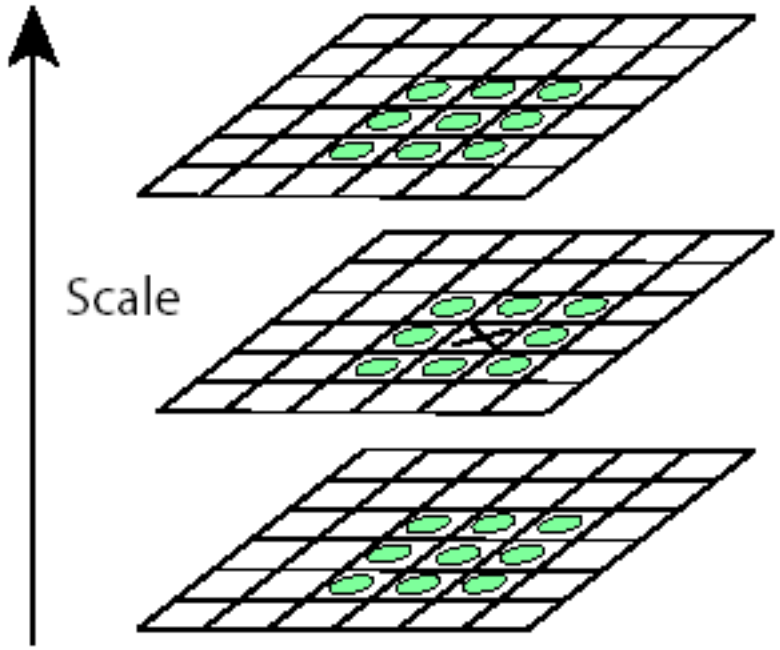
Keypoint Selection

1. Supersample original image
2. Compute smoothed images using different scales σ for entire octave
3. Compute DoG images from adjacent scales for entire octave
4. Subsample image 2σ of current octave and repeat process (2-3) for next octave
5. Isolate keypoints in each octave by detecting extrema in DoG compared to neighboring pixels

Images of process



Keypoint Detection



X is defined as a “key point” if it is the greatest or least of all 26 neighbors in each octave.

Keypoint Localization

- From difference-of-Gaussian local extrema detection we obtain approximate values for keypoints.
- If these approximations were used directly, the number of keypoints is usually a large number in one image.

The Taylor Series Expansion

- Take Taylor Series Expansion of scale-space function D(x,y,σ)
  - Use up to quadratic terms

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

  - $x = (x, y, \sigma)^T$ offset from this sample point
  - We take derivative and set to 0 to find location of extremum.

$$\bar{x} = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D}{\partial x}$$

Thresholding Keypoint: Low Contrast

- The function value at the extrema is used to reject unstable extrema (Low contrast)
  - Evaluate

$$D(\bar{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\bar{x}$$

  - Threshold to reject extrema:

    The absolute value of D is less than 0.03.

Thresholding Keypoint: Along Edge

- Difference-of-Gaussian function will be strong along edges
  - Some locations along edges are poorly determined and will become unstable when even small amounts of noise are added
  - These locations will have a large principal curvature across the edge but a small principal of curvature perpendicular to the edge
  - Therefore we need to compute the principal curvatures at the location and compare the two

Computing the Principal Curvatures

- Hessian matrix

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

  - The eigenvalues of H are proportional to principal curvatures

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta \qquad \alpha = r\beta$$

$$Det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

    r is the ratio between two eigenvalues of *H.*

  - We are not concerned about actual values of eigenvalue, just the ratio of them

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

    In practice, we utilize *r=10* to reject keypoints if

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} \leq \frac{(r+1)^2}{r}$$

Example

Keypoint detection

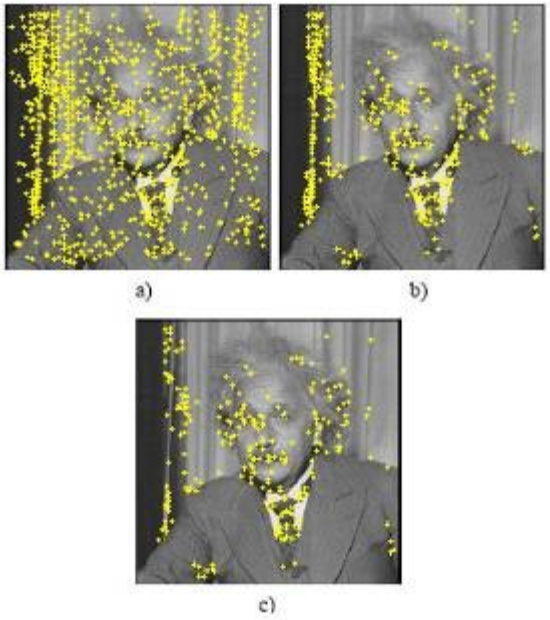


Figure 5: a) Maxima of DoG across scales. b) Remaining keypoints after removal of low contrast points. C) Remaining keypoints after removal of edge responses (bottom).

Final keypoints with selected orientation and scale

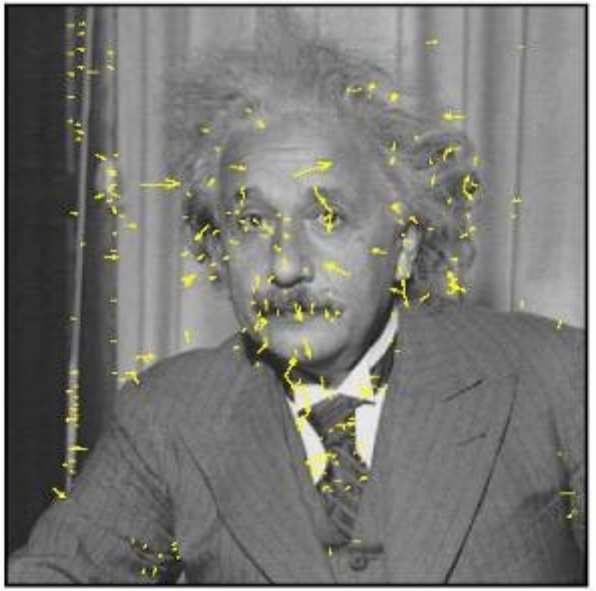


Figure 6: Extracted keypoints, arrows indicate scale and orientation.

Assigning an Orientation 1

- We finally have a keypoint that we are going to keep
- The next step is assigning an orientation for the keypoint
  - Used in making the matching technique invariant to rotation
  - Used for determining whether or not a training image is actually in test image

Assigning an Orientation 2

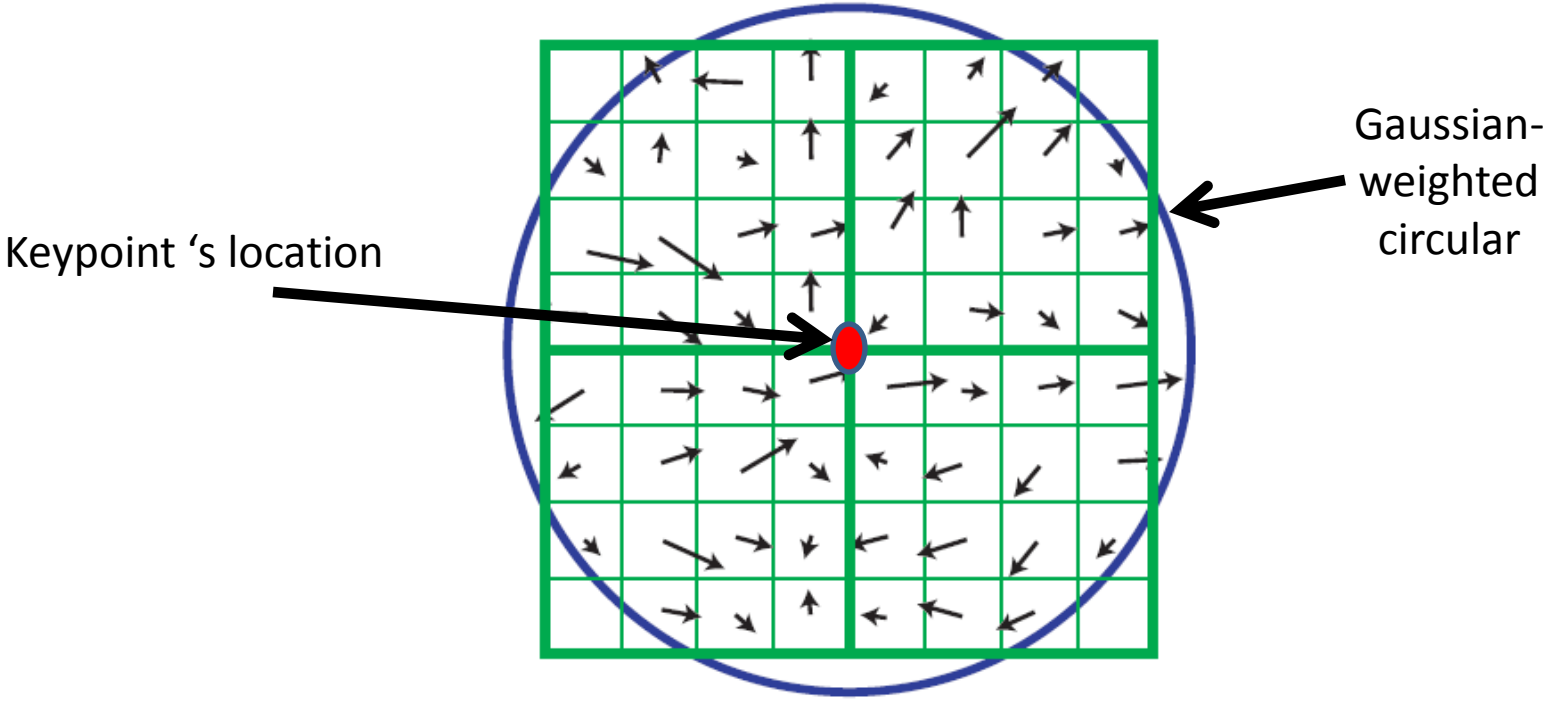
- Gaussian smoothed image, *L*, with closest scale is chosen (scale invariance)
- Points in region around keypoint are selected and magnitude and orientations of gradient are calculated

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Assigning an Orientation 3

- Orientation histogram formed with 36 bins. Sample is added to appropriate bin and weighted by gradient magnitude and Gaussian-weighted circular window with a of σ 1.5 times scale of keypoint.

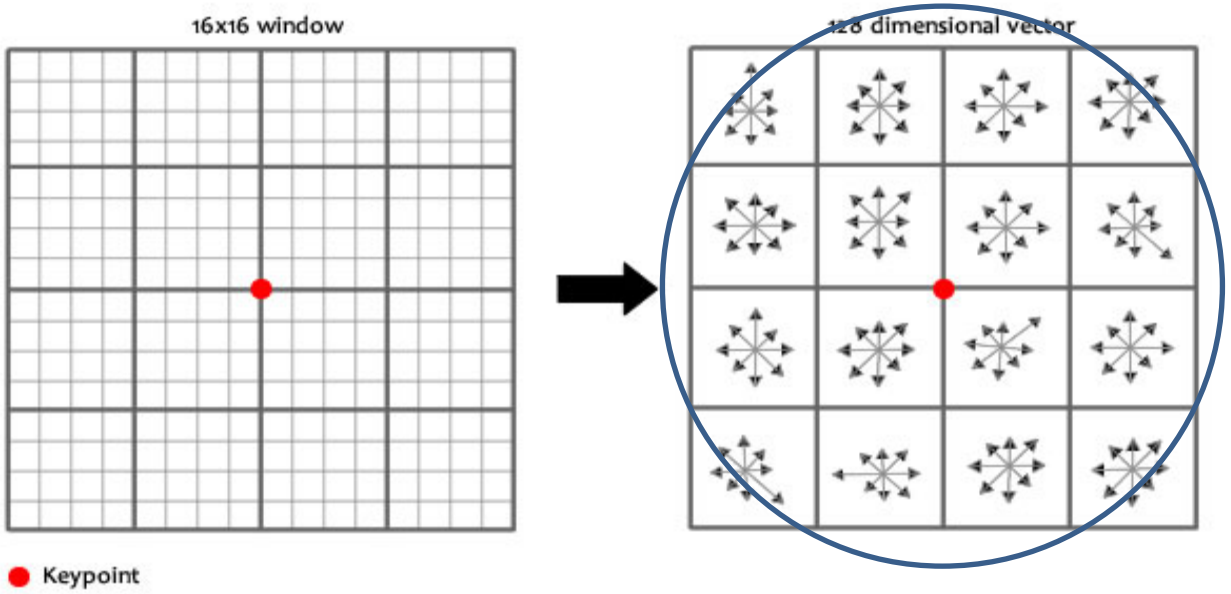


Assigning an Orientation 4

- Highest peak in orientation is found along with any other peaks within 80% of highest peak
- 3 closest histogram values to each peak are used to interpolate (fit to a parabola) a better accurate peak

Assigning an Orientation 5

- As of now each keypoint has 4 dimesions: x location, y location, scale, and orientation



A 4x4 array from 16 x 16 window with 8 orientations = 128-dimensional feature vector

Keypoint Descriptor 1

- Provides invariance to changes in illumination and 3D camera viewpoint
- Is calculated for a region around a keypoint as opposed to directly for the keypoint
- Like before, magnitudes and orientations are calculated for points in region around keypoint using *L* of nearest scale
- To ensure orientation invariance the gradient orientations and coordinates of descriptor are rotated relative to orientation of keypoint

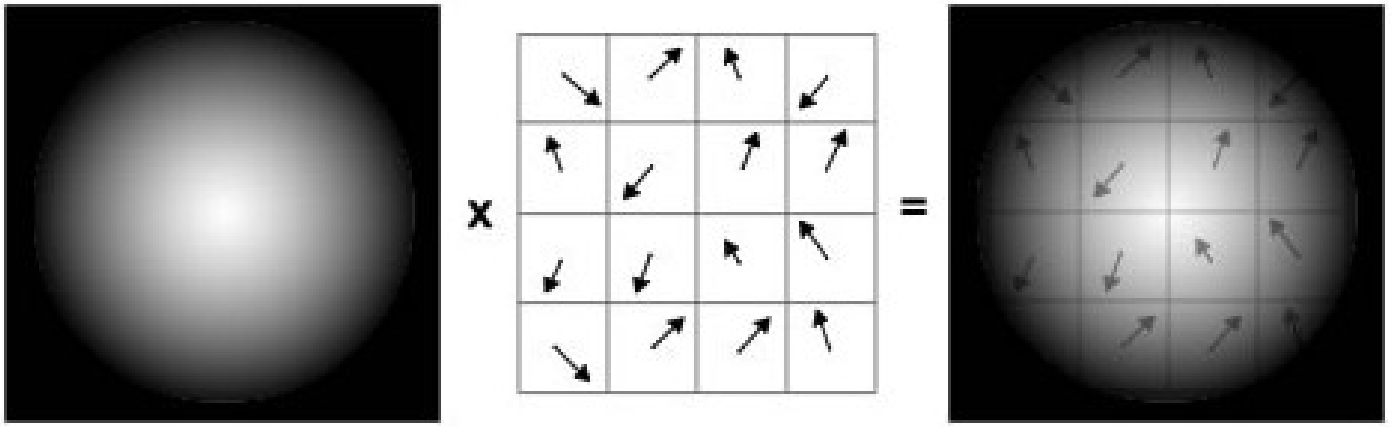
Keypoint Descriptor 2

- Magnitude of each point is weighted with σ of one half the width of the descriptor window
  - Stops sudden changes in descriptor due to small changes in position of the window
  - Gives less weight to gradients far from keypoint
- Samples are divided into 4x4 subregions around keypoint. Allows for a change of up to 4 positions of a sample while still being included in the same histogram

Keypoint Descriptor 3

- Avoiding boundary effects between histograms
  - Trilinear interpolation used to distribute value of gradient of each sample into adjacent histogram bins
    - Weight equal to 1 – d , where d is the distance of a specific sample to the center of a bin
- Vector normalization
  - Done at the end to ensure invariance to illumination change (affine)
  - Entire vector normalized to 1
  - To combat non-linear illumination changes values in feature vector are thresholded to no larger than 0.2 and then the vector is normalized again.

Weighted Keypoint Descriptor



Normal distribution

Keypoint descriptor

Weighted keypoint descriptor

Keypoint Descriptor Complexity

- An interesting aside
- The size of the descriptor is rn^2
- As the vector grows in size
  - Its ability to be discriminated in a larger feature database will improve
  - Its sensitivity to distortions from 3D viewpoints and occlusions will increase
- In the paper they use a 4x4 array with 8 orientations giving a 128 dimensional feature vector

SIFT features

- Up to this point we have:
  - Found rough approximations for features by looking at the difference-of-Gaussians
  - Localized the keypoint more accurately
  - Thresholded poor keypoints
  - Determined the orientation of a keypoint
  - Calculated a 128 feature vector for each keypoint (keypoint descriptor)

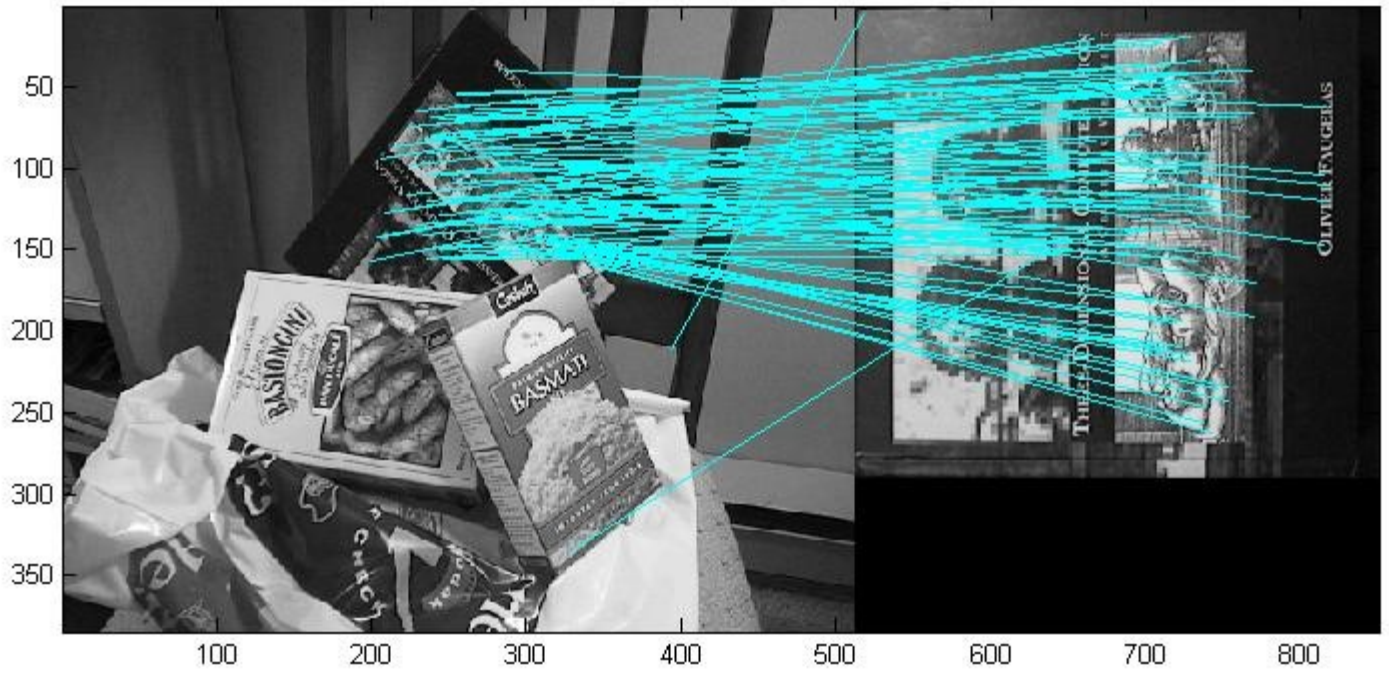
Matching Keypoints between images

- Tale of two images (or more)
  - One image is the training sample of what we are looking for
  - The other image is the world picture that might contain instances of the training sample
- Both images have features associated with them across different octaves
- How do we match features between the two?

Matching Keypoints between images (cont.)

- Nearest Neighbor Algorithm
  - Independently match all keypoints in all octaves in one image with all keypoints in all octaves in other image
  - How to solve problem of features that have no correct match with opposite image (i.e. background noise, clutter)
    - Max distance (did not perform well)
    - Ratio of closest nearest neighbor with second closest nearest neighbor)

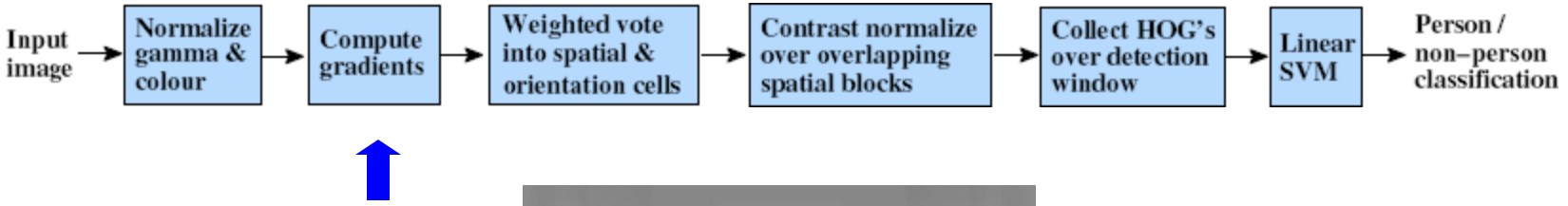
Example

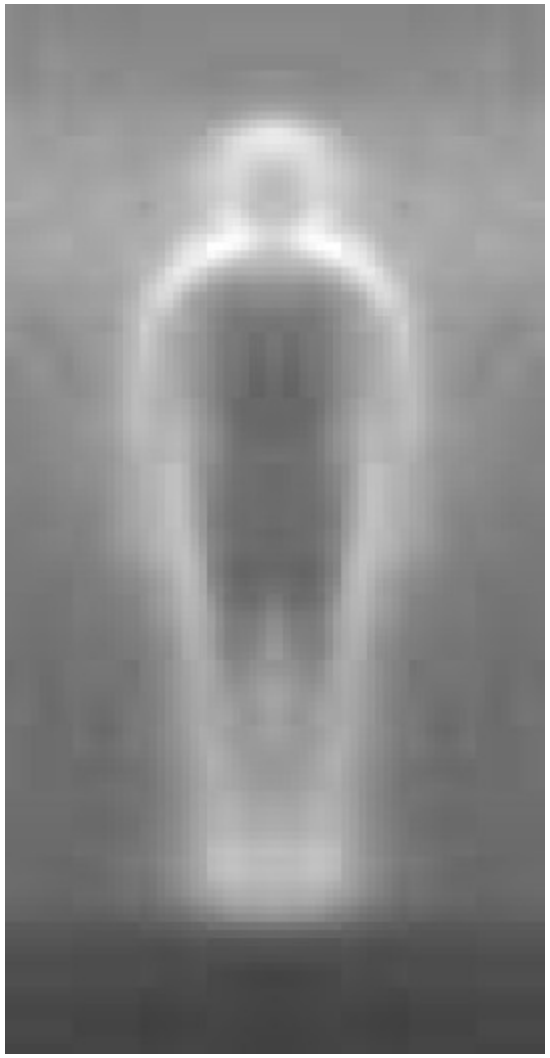

50
100
150
200
250
300
350
100
200
300
400
500
600
700
800
BASMATI
OLIVER TAUGEAS

References

- David G. Lowe, **"Distinctive image features from scale-invariant keypoints,"** *International Journal of Computer Vision,* 60, 2 (2004), pp. 91-110.

Histogram of Gradients





-1	0	1

centered

-1	1

uncentered

1	-8	0	8	-1

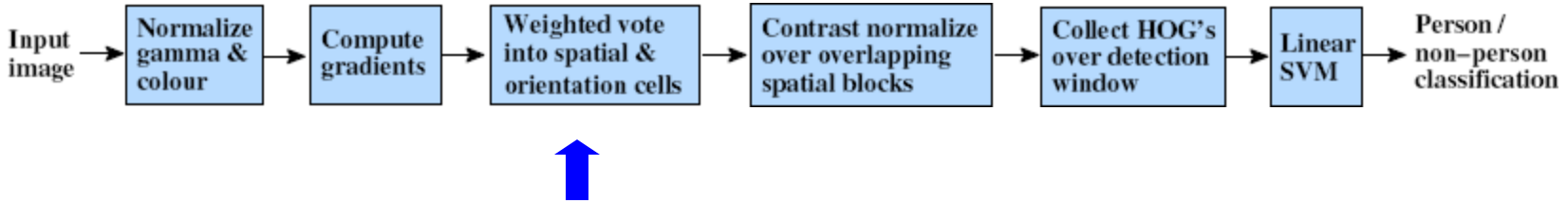
cubic-corrected

0	1
-1	0

diagonal

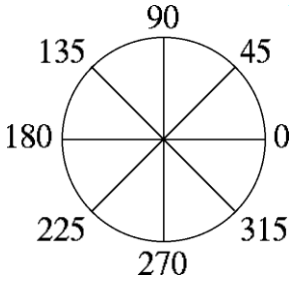
-1	0	1
-2	0	2
-1	0	1

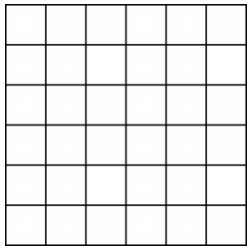
Sobel



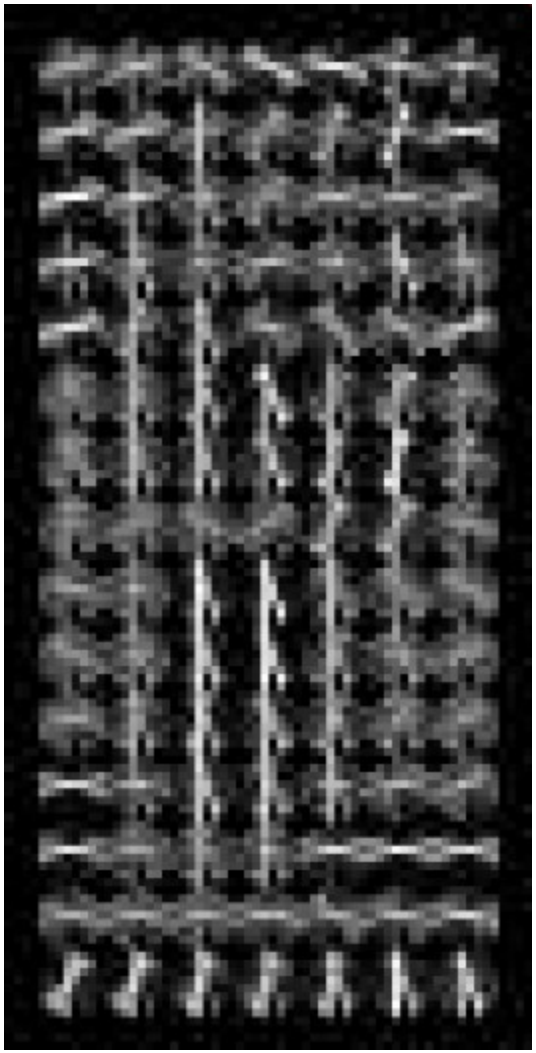
- Histogram of gradient orientations

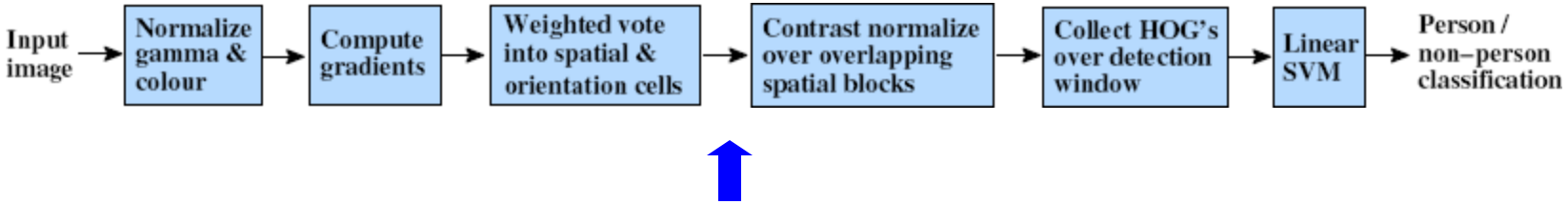
  -Orientation -Position

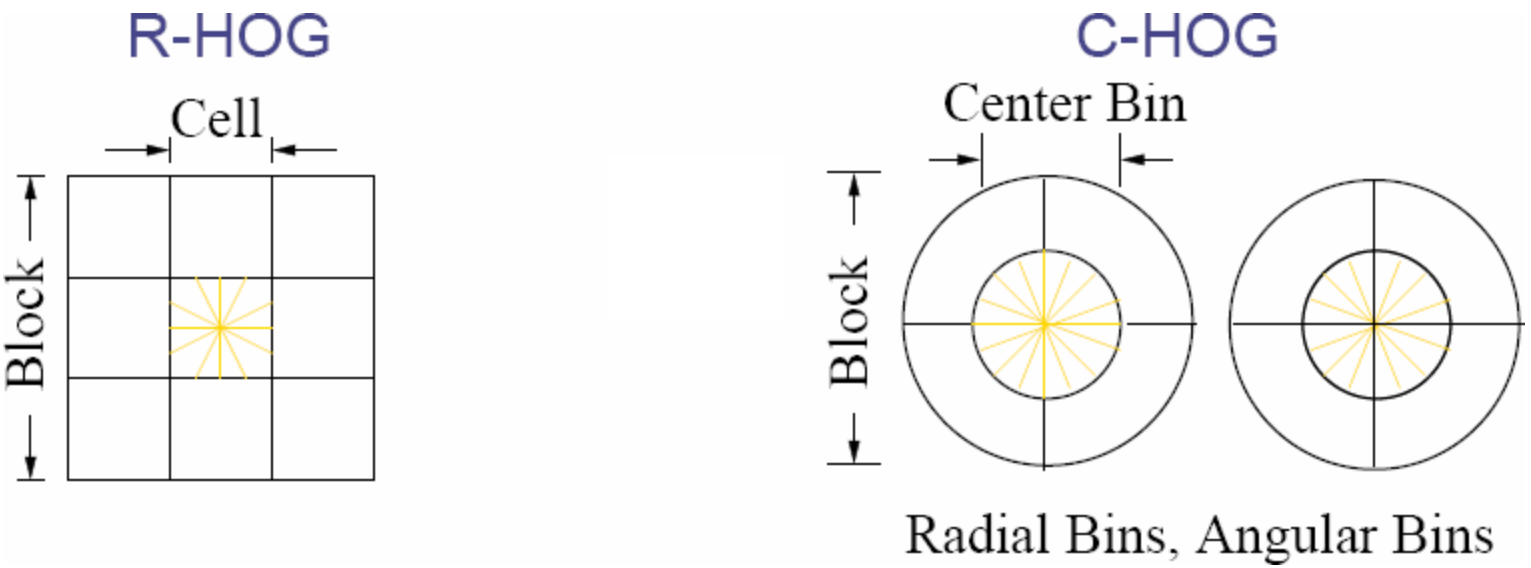


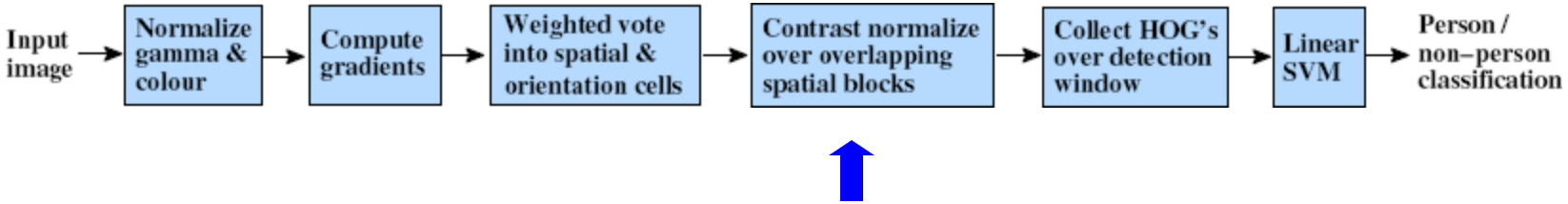


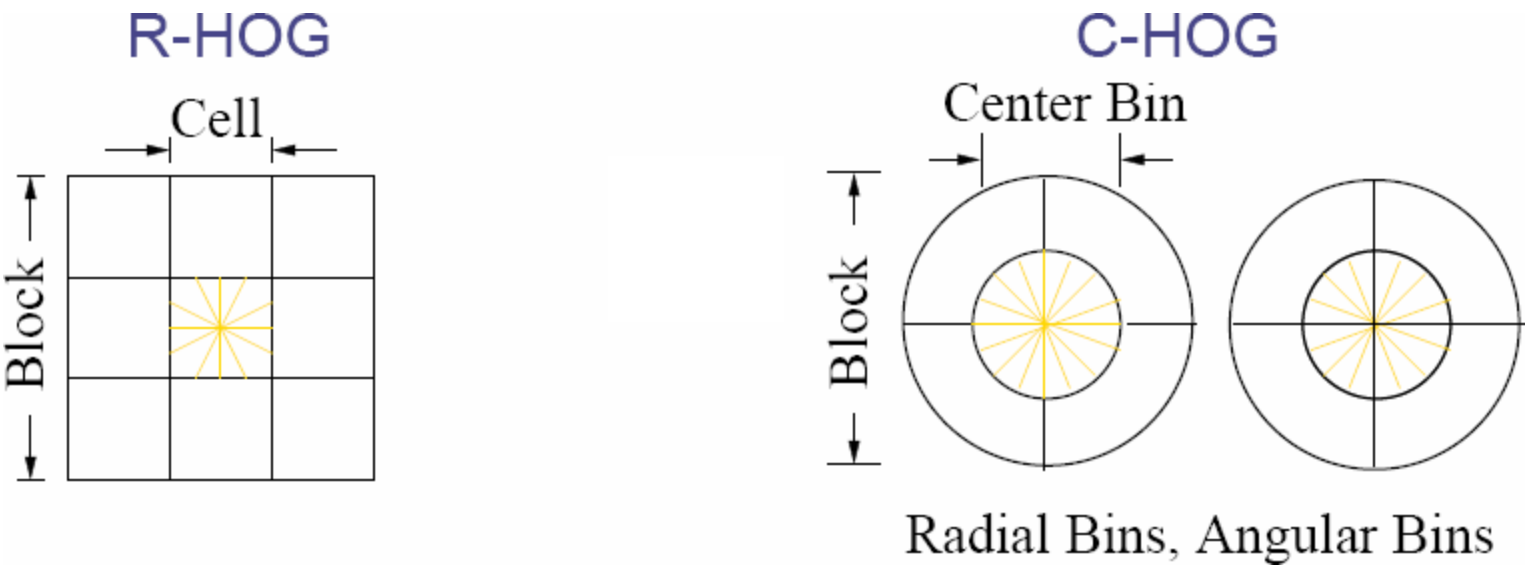
  – Weighted by magnitude




Input image
Normalize gamma & colour
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window
Linear SVM
Person / non−person classification


R-HOG
Cell
Block
C-HOG
Center Bin
Block
Radial Bins, Angular Bins



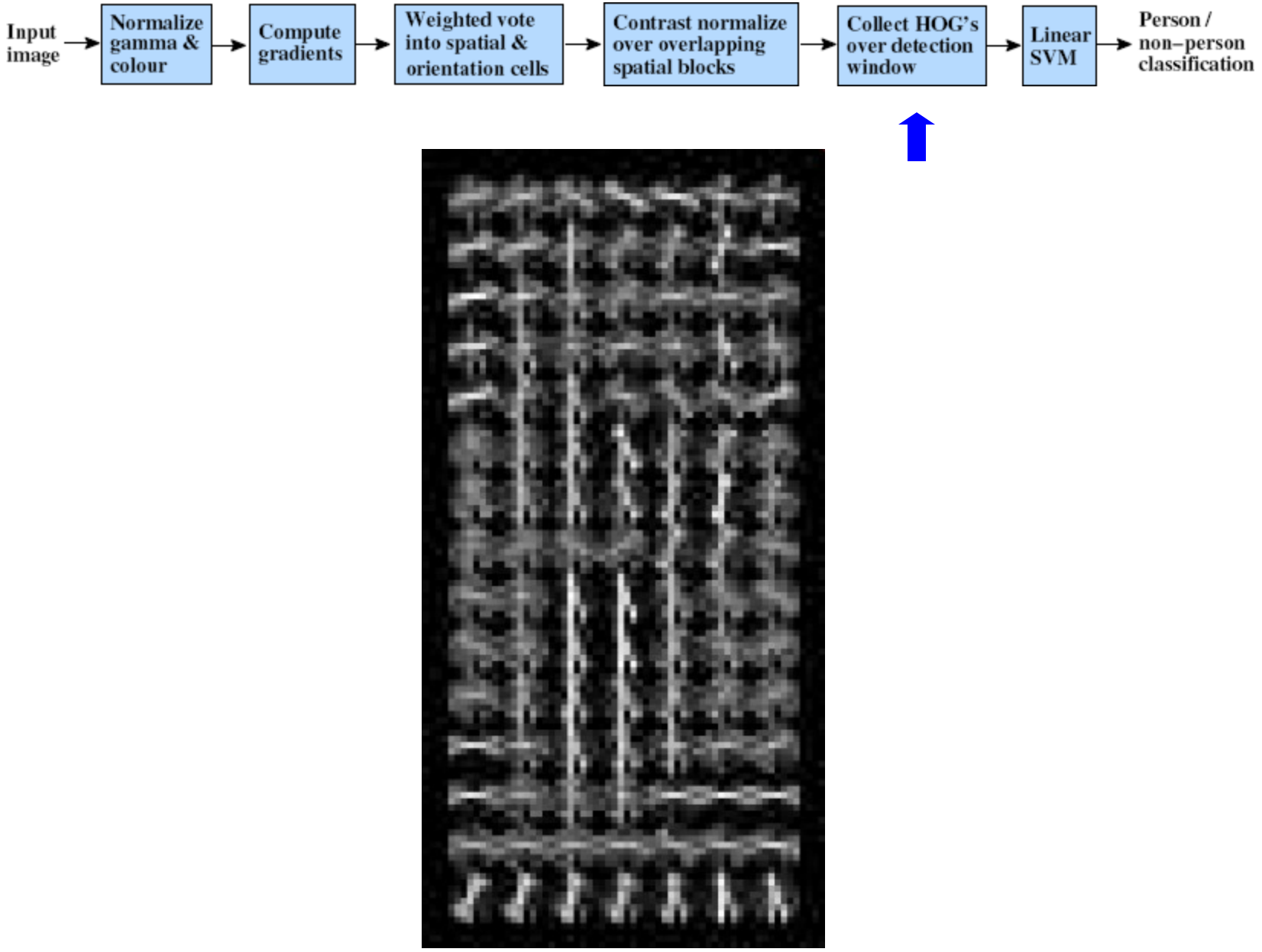


$L1 - norm : v \longrightarrow v/(||v||_1 + \epsilon)$

$L1 - sqrt : v \longrightarrow \sqrt{v/(||v||_1 + \epsilon)}$

$L2 - norm : v \longrightarrow v/\sqrt{||v||_2^2 + \epsilon^2}$

$L2 - hys$: L2-norm, plus clipping at .2 and renomalizing


Input image
Normalize gamma & colour
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window
Linear SVM
Person / non−person classification

References

- N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” CVPR 05, pp. 886-893.