Computer Vision

MSc Informatics option GVR
James L. Crowley

Fall Semester 8 November 2011

Lesson 5

View Invariant Image Description

Lesson Outline:

1. Describing Local Appearance

Appearance is what you see. Consider an image p(i, j).

We seek a set of K local basis functions, $f_k(x,y)$ to describe "appearance" around a point in an image. These are referred to as "local image description functions" or simply "Local features". (x and y are integers).

Each function, $f_k(x,y)$ gives an image "feature", a_k, describing appearance in the neightborhood of the image position p(i, j).

$$a_k(i,j) = \sum_{x=-R}^{R} \sum_{y=-R}^{R} p(i-x, j-y) f_k(x,y)$$

Projection of the image neighborhood *p(i,j)* onto this set of functions gives a

"feature" vector for appearance, $\vec{A}(i,j) = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_K \end{pmatrix}$ around position *p(i,j).*

Ideally the set of local image description functions should be orthogonal

$$\sum_{x=-R}^{R} \sum_{y=-R}^{R} p(i-x, j-y) f_k(x,y) = 0 \quad \text{if } n \neq m \quad \text{for all } (i, j)$$

to minimize the number of features. However, it can be shown that a function cannot be both orthogonal and shift invariant. Because shift invariance is important for robust view invariant recognition, we compromise on orthoganality.

For a variety of reasons, derivatives of the Gaussian function have been found to be very useful as local image features. Chief among these are invariance to affine transformations.

2. Gaussian Derivatives as Local images Features

2.1. The Gaussian Function

The Gaussian Function is $G(x,\sigma) = e^{-\frac{x^2}{2\sigma^2}}$

The Gaussian function is invariant to affine transformations.

$$T_a\{G(x, y, \sigma)\} = G(T_a\{x\}, T_a\{y\}, T_a\{\sigma\})$$

Recall from lesson 2 we saw that $x_r = x_c \frac{F}{z_c}$

The apparent size of an object is inversely proportional to its distance

A change in size (or scale) is a special case of an affine transform:

$$T_s\{G(x, y, \sigma)\} = G(T_s\{x\}, T_s\{y\}, T_s\{\sigma\}) = G(sx, sy, s\sigma)$$

This is just one of the many interesting properties of the Gaussian function.

2.2. Gaussian Derivatives Operators

The Gaussian function is: $G(x,\sigma) = e^{-\frac{x^2}{2\sigma^2}}$

Fourier Transform: $F\{e^{-\frac{x^2}{2\sigma^2}}\} = G(\omega,\sigma) = \sqrt{2\pi}\ \sigma e^{-\frac{1}{2}\sigma^2\omega^2}$

Scale property: $G(x,\sqrt{2}\sigma) = G(x,\sigma) * G(x,\sigma)$

Derivatives:

$$\frac{\partial G(x,\sigma)}{\partial x} = -\frac{x}{\sigma^2} G(x,\sigma) = G_x(x,\sigma)$$

$$\frac{\partial^2 G(x,\sigma)}{\partial x^2} = \frac{x-\sigma^2}{\sigma^4} G(x,\sigma) = G_{xx}(x,\sigma)$$

$$\frac{\partial^3 G(x,\sigma)}{\partial x^3} = \frac{x^3 - x\sigma^2}{\sigma^6} G(x,\sigma) = G_{xxx}(x,\sigma)$$

2.3. 2D Gaussian functions

2D Gaussian Kernel: $G(x,y,\sigma) = e^{-\frac{(x^2+y^2)}{2\sigma^2}}$

Fourier Transform: $F\{e^{-\frac{x^2+y^2}{2\sigma^2}}\} = \frac{\pi}{2\sigma^2} e^{-\frac{1}{2}\sigma^2(u^2+v^2)}$

Separability: $G(x,y,\sigma) = e^{-\frac{(x^2+y^2)}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} * e^{-\frac{y^2}{2\sigma^2}}$

Scale property: $G(x,y,\sqrt{2}\sigma) = G(x,y,\sigma) * G(x,y,\sigma)$

Derivatives:

$$\frac{\partial G(x,y,\sigma)}{\partial x} = -\frac{x}{\sigma^2} G(x,y,\sigma) = G_x(x,y,\sigma)$$
$$\frac{\partial G(x,y,\sigma)}{\partial y} = -\frac{y}{\sigma^2} G(x,y,\sigma) = G_y(x,y,\sigma)$$
$$\frac{\partial^2 G(x,y,\sigma)}{\partial x^2} = \frac{x-\sigma^2}{\sigma^4} G(x,y,\sigma) = G_{xx}(x,y,\sigma)$$
$$\frac{\partial^2 G(x,y,\sigma)}{\partial x \partial y} = \frac{xy}{\sigma^4} G(x,y,\sigma) = G_{xy}(x,y,\sigma)$$
$$\frac{\partial^3 G(x,y,\sigma)}{\partial x^3} = \frac{x^3 - x\sigma^2}{\sigma^6} G(x,y,\sigma) = G_{xxx}(x,y,\sigma)$$

The Laplacian of the Gaussian:

$$\nabla^2 G(x,y,\sigma) = G_{xx}(x,y,\sigma) + G_{yy}(x,y,\sigma)$$

Diffusion Equation:

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

As a consequence: $\nabla^2 G(x,y,\sigma) \approx \left(G(x,y,\sigma_1) - G(x,y,\sigma_2)\right)$

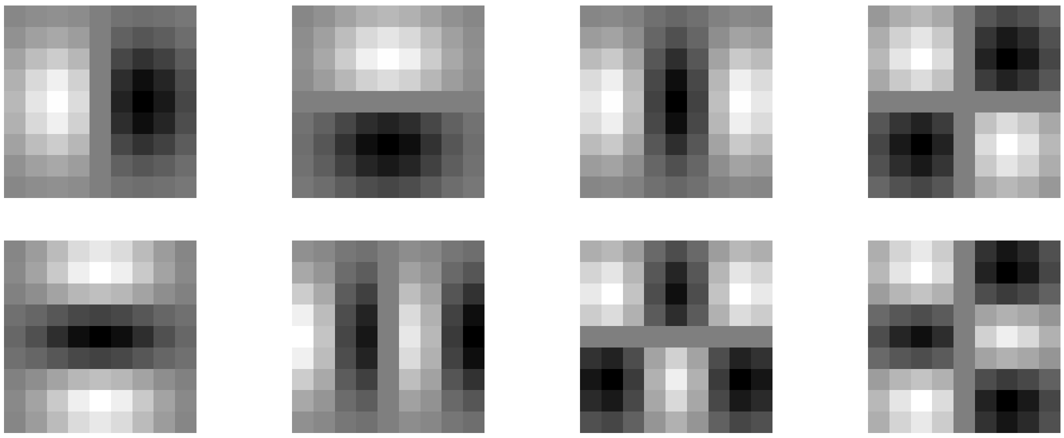
This is called a Difference of Gaussian (DoG) and typically requires $\sigma_1 \geq 1.4\ \sigma_2$
It is common to use: $\nabla^2 G(x,y,\sigma) \approx G(x,y,\sqrt{2}\sigma) - G(x,y,\sigma)$

But note that from the scale property : $G(x,y,\sqrt{2}\sigma) \approx G(x,y,\sigma) * G(x,y,\sigma)$

so that $\nabla^2 G(x,y,\sigma) \approx G(x,y,\sigma) * G(x,y,\sigma) - G(x,y,\sigma)$

We can use these functions to create a basis set of receptive fields for appearance

$G = (G_x, G_y, G_{xx}, G_{xy}, G_{yy}, G_{xxx}, G_{xxy}, G_{xyy}, G_{yyy})$



The Gaussian receptive fields $G_x, G_y, G_{xx}, G_{xy}, G_{yy}, G_{xxx}, G_{xxy}, G_{xyy}, G_{yyy}$.

2.4. The Sampled Gaussian Functions

Let x and y be integers.

The 2D Gaussian Receptive Field is : $G(x,y,\sigma) = \frac{1}{B} W_R(x,y) \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}}$

where

$$w_R(x,y) = \begin{cases} 1 & \text{for } -R \leq x \leq R \text{ and } -R \leq y \leq R \\ 0 & \text{otherwise} \end{cases}$$

Finite windw, $w_R(x,y)$ has $N^2 = (2R+1)^2$ coefficients

Typically: for R should be $\geq 3\sigma$. Recommend $R=4\sigma$

The normalization factor $B = \sum_{x=-R}^{R} \sum_{y=-R}^{R} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \approx 2\pi\sigma$

2.5. Using the Gaussian to compute image derivatives

For an image *p(i,j)*, the derivatives can be approximated by convolution with Derivatives of Gaussians.

$$\frac{\partial p(i,j)}{\partial x} * G(x,y) = \frac{\partial}{\partial x} * p(i,j) * G(x,y) = \frac{\partial}{\partial x} * G(x,y) * p(i,j) = \frac{\partial G(x,y)}{\partial x} * p(i,j)$$

Thus we can approximate an image derivative as $P_x(i,j) \approx G_x * P(i,j)$
However to compute G_x, it is NECESSARY to specify σ.
Small σ is not necessarily best.

$$p_x(i,j,\sigma) \approx G_x(x,y,\sigma) * p(i,j)$$

or more simply $p_x(i,j,\sigma) \approx G_x(\sigma) * p(i,j)$

Simarly:

$$p_y(i,j,\sigma) \approx G_y(\sigma) * p(i,j)$$
$$p_{xx}(i,j,\sigma) \approx G_{xx}(\sigma) * p(i,j)$$
$$p_{xy}(i,j,\sigma) \approx G_{xy}(\sigma) * p(i,j)$$
$$p_{yy}(i,j,\sigma) \approx G_{yy}(\sigma) * p(i,j)$$

The Gradient of the image $\vec{\nabla} p(i,j)$ is calculated by $\vec{\nabla} G(\sigma) * p(i,j)$

where $\vec{\nabla} G(\sigma) = \begin{pmatrix} G_x(\sigma) \\ G_y(\sigma) \end{pmatrix}$ This gives:

Gradient: $\vec{\nabla} p(i,j,\sigma) = \begin{pmatrix} p_x(i,j,\sigma) \\ p_y(i,j,\sigma) \end{pmatrix} \approx \vec{\nabla} G(\sigma) * p(i,j) = \begin{pmatrix} G_x(\sigma) * p(i,j) \\ G_y(\sigma) * p(i,j) \end{pmatrix}$

Laplacien:

$$\nabla^2 p(i,j,\sigma) = \nabla^2 G(\sigma) * p(i,j) = p_{xx}(i,j,\sigma) + p_{yy}(i,j,\sigma) \approx G_{xx}(\sigma) * p(i,j) + G_{yy}(\sigma) * p(i,j)$$

2.6. Steerability of Gaussian Derivatives.

It is possible to synthesize an oriented derivative at any point as a weighted sum of derivatives in perpendicular directions. The weights are given by sine and cosine functions. The weights are given by sine and cosine functions.

$$G_1^\theta(x,y,\sigma) = \cos(\theta) \cdot G_x(x,y,\sigma) + \sin(\theta) \cdot G_y(x,y,\sigma)$$

Higher order derivatives can also be steered.

Thus:

1st order $p_1^\theta(i,j,\sigma) = Cos(\theta) p_x(i,j,\sigma) + Sin(\theta) p_y(i,j,\sigma)$
2nd order $p_1^\theta(i,j,\sigma) = Cos(\theta)^2 p_{xx}(i,j,\sigma) + Sin(\theta)^2 p_{yy}(i,j,\sigma) + 2Cos(\theta)Sin(\theta) p_{xy}(i,j,\sigma)$
3rd order

$$p_3^\theta(i,j,\sigma) = Cos(\theta)^3 p_{xxx}(i,j,\sigma) + Cos(\theta)^2 Sin(\theta) p_{xxy}(i,j,\sigma) + Cos(\theta) Sin(\theta)^2 p_{xyy}(i,j,\sigma) + Sin(\theta)^3 p_{yyy}(i,j,\sigma)$$

By steering the derivatives to the local orientation, we obtain an "invariant" measure of local contrast. We can also "steer" in scale to obtain invariance to size.

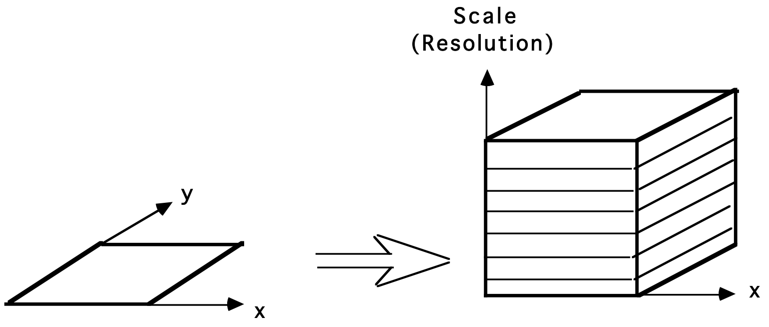
Note, we can NOT steer the mixed derivatives, i.e $p_{xy}(i, j, \sigma)$

2.7. Intrinsic Orientation:

For each pixel, one can calculate the orientation of maximal gradient. This orientation is equivariant with rotation. One can use this as an "intrinsic" orientation to normalize the receptive fields at any point in the image.

Local orientation: $\theta_i(x,y,\sigma) = Tan^{-1}\left(\frac{G_y \cdot P(x,y,\sigma)}{G_x \cdot P(x,y,\sigma)}\right)$

3. Image Scale Space



Our objective is to describe the appearance at each pixel of the image in a manner that does not change (or changes very little) with position, illumination color, distance, or viewing direction.
We will obtain invariance to position by using shift invariant filters
We will obtain invariance to illumination color by estimating the illumination color and correcting the image description in LC_1C_2.
We will obtain invariance to distance by working in a Scale Space

3.1. Continuous Scale Case.

Let P(x,y) be the image.
Let G(x, y, σ) by a Gaussian function of scale $\sigma=2^s$

The image Scale Space is a 3D continuous space *p(x, y, s)*

$$p(x, y, s) = p(x,y) * G(x, y, 2^s)$$

Note that the scale (s) axis is logarithmic. $s = \mathrm{Log}_2(\sigma)$

3.2. Discrete Scale Space

Suppose *p(x, y)* is an image array of size M x M pixels.

We will sample the scale axis with a step size of $\Delta\sigma = 2^{1/2}$: $\sigma_k \approx 2^{k/2}$

Because we use a low pass filter, $G(x, y, \sigma_k)$, as σ_k grows it becomes possible to resample the image with a larger step size without loss of information. Such resampling has the benefit of assuring an invariance of the impulse response of each image. The sample size Δx_k, Δy_k can grow exactly as σ_k.

$p(i, j, k) = p(x\,\Delta x_k, y\,\Delta x_k, k)$

What sample size is possible? It is possible to show that the sample step must be smaller than $\frac{1}{\sqrt{2}}\sigma$. $\Delta x \leq 2^{-1/2}\sigma$

If we choose k such that $\Delta x_k = 2^{k/2}$, then $\sigma_k = 2^{(k+1)/2}$

This defines a resampled scale space (a pyramid)

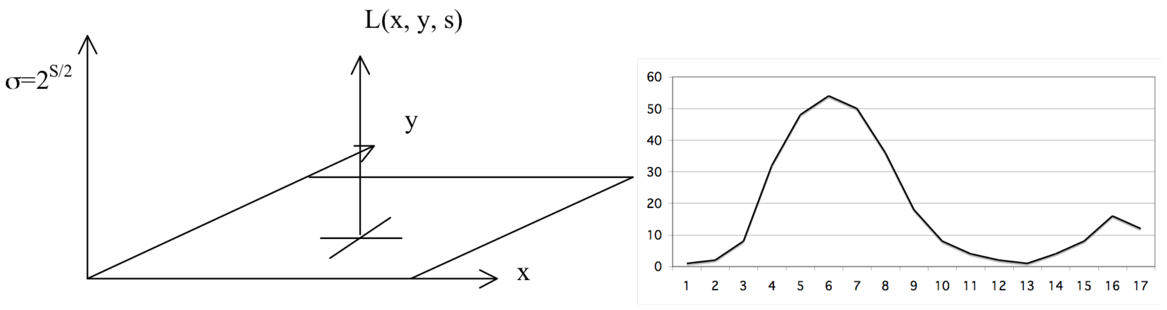
$$p(i, j, k) = p(x\,\Delta x_k, y\,\Delta x_k, k) \quad \text{such that } \Delta x_k = 2^{k/2} \text{ and } \sigma_k = 2^{(k+1)/2}$$

3.3. Laplacian Profile

At every image point, the Laplacian profile is the Laplacian of the image computed over a continuous (exponential) range of scales.

$$L(x, y, s) = P(x, y) * \nabla^2 G(x, y, 2^{s/2})$$

The Laplacian profile is invariant to rotation and translation and equivariant to changes in scale. Since scale is proportional to distance, the profile is invariant to viewing distance.



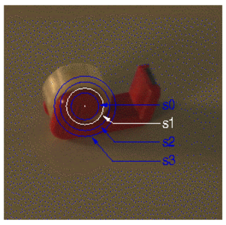
A change in viewing distance at x, y shifts the function L(x,y,s) in s.
The function remains the same. Thus the maximum is a local invariant.

The "intrinsic" scale at a point (x, y) is $\sigma_i = 2^{s_i/2}$

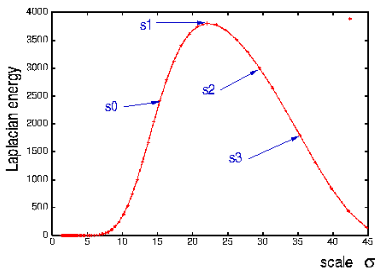
where:

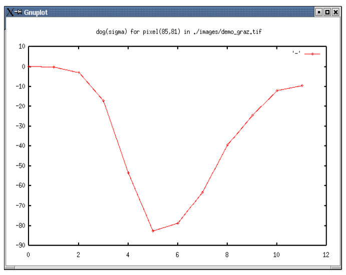
$$s_i = \arg\text{-}\max_s\{L(x,y,s)\}$$

Examples:



zero crossing of Laplacian at si





The scale of the maximal Laplacian is an invariant at ALL image points.

3.4. Scale Invariant Interest Points

Maximal points in the image derivatives provide keypoints.
In an image scale space, these points are scale invariant.

Example: maxima in the Lapacian as invariant "interest points"

Recall the Laplacian of the image :

$$\nabla^2 p(x,y,s) = \nabla^2 G(\sigma) * p(x,y) = G_{xx}(\sigma) * p(x,y) + G_{yy}(\sigma) * p(x,y)$$

from the diffusion equation, we know that for a Gaussian

$$\nabla^2 G(\sigma) = \frac{\partial^2 G(\sigma)}{\partial x^2} + \frac{\partial^2 G(\sigma)}{\partial y^2} = \frac{\partial G(\sigma)}{\partial \sigma} \approx \left(G(\sigma_1) - G(\sigma_2)\right)$$

This is referred to as a "Difference of Gaussian" or DoG.

The approximation works best at $\sigma_1 = \sqrt{2}\sigma_2$

Thus we can compute the Laplacian as a difference of Gaussians:

$$\nabla^2 G(\sigma) * p(x,y) \approx \left(G(\sigma_1) - G(\sigma_2)\right) * p(x,y) = G(\sigma_1) * p(x,y) - G(\sigma_2) * p(x,y)$$

and we can compute the Laplacian of the image as a difference of Gaussian smoothed images.

$$\nabla^2 p(i,j,\sigma) = \nabla^2 G(\sigma) * p(i,j) \approx G(\sqrt{2}\sigma) * p(i,j) + G(\sigma) * p(i,j)$$

Using an image Pyramid, the Laplacian is simply the difference at adjactent levels.

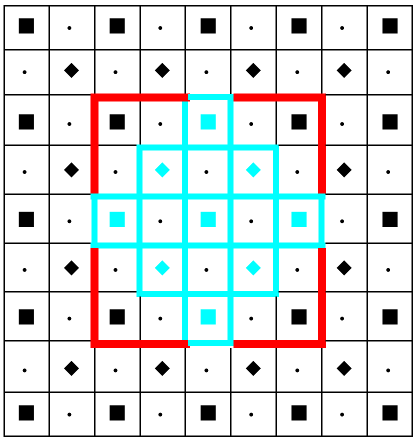
DoG: $L(i, j, k) = \nabla^2 p(i, j, k) = p(i, j, k) - p(i, j, k-1)$

We can detect scale invariant interest points as

$$X(i,j,k) = \underset{i,j,k,R}{local-\max}\{L(i,j,k)\}$$

with R=1 at k, and R=2 at k-1, and k+1.

Note that because of resampling, $\Delta x = 2^{k/2}$, the neighborhood grows larger as k increases.



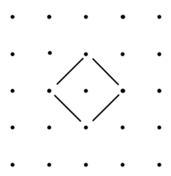
Level k-1 - Red arg-max over i, j for R=2
Level k - Blue (Note neighborhood, R=$\sqrt{2}$ is smaller than level k-1!
level k+1 black. arg-max over i, j for R=2.

Such points are used for tracking, for image registration, and as feature points for recognition.

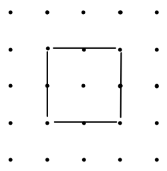
3.5. Neighborhoods in a Pyramid

Computing a variable Radius Local-Max operator over a $\sqrt{2}$ image pyramid can be somewhat complex.

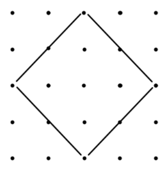
R=1 4 Nearest Neighbors - Odd (Cartesian) levels :



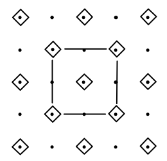
R=$\sqrt{2}$ 8 Nearest Neighbors - Odd (Cartesian) levels :



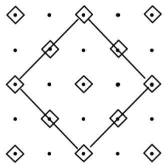
R=2 12 Nearest Neighbors Odd Levels



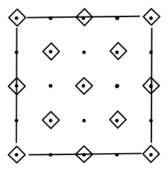
R=1 4 Nearest Neighbors - Even ($\sqrt{2}$)levels



R=$\sqrt{2}$ 8 Nearest Neighbors - Even ($\sqrt{2}$)levels



R=2 12 Nearest Neighbors Even ($\sqrt{2}$)levels



3.6. Other popular interest point detectors.

Other popular detectors for scale invariant interest points include:

Gradient Magnitude: $A(i,j,k) = \underset{i,j,k}{Local-}\max\{\| P_x(i,j,k), P_y(i,j,k)\|\}$

and Determinant of the Hessian: $A(i,j,k) = \underset{i,j,k}{Local-}\max\left\{\det\begin{pmatrix} P_{xx}(i,j,k) & P_{xy}(i,j,k) \\ P_{xy}(i,j,k) & P_{yy}(i,j,k)\end{pmatrix}\right\}$

$$A(i,j,k) = \underset{i,j,k}{Local-}\max\left\{P_{xx}(i,j,k)P_{yy}(i,j,k) - P_{xy}(i,j,k)^2\right\}$$

and the Harris-Laplace.

let $b_2(i,j) = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

$$H_x^2 = b_2 * P_x^2$$

$$H_{xy} = b_2 * P_{xy}$$
$$H_y^2 = b_2 * P_y^2$$

$$H = \begin{pmatrix} H_x^{\ 2} & H_{xy} \\ H_{xy} & H_y^{\ 2} \end{pmatrix}$$

Harris interest points h(i,j,k) = arg-max{det(H)-Trace(H)}

4. HoG: Histogram of Gradients

A local histogram of gradient orientation provides a vector of features image appearance that is relatively robust to changes in orientation and illumination.

HoG gained popularity because of its use in the SIFT feature point detector (described next). It was subsequently explored and made popular by Navneet Dalal (M2R GVR 2003) and Bill Triggs.

Recall: The orientation of a gradient at pyramid sample (i,j,k) is:

$$\theta(i,j,k) = Tan^{-1}\left\{\frac{p_y(i,j,k)}{p_x(i,j,k)}\right\}$$

This is a number between 0 and π. We can quantize it to a value between 1 and N value by

$$a(i,j,k) = N \cdot Trunc\left\{\frac{\theta(i,j,k)}{\pi}\right\}+1$$

We can then build a local histogram for a window of size WxH, with upper left corner at i_o, j_o, k. We allocate a table of N cells: h(a). Then for each pixel i,j in our window:

$$\overset{W}{\underset{i=1}{\forall}}\ \overset{H}{\underset{j=1}{\forall}} h(a(i+i_o, j+j_o, k)) = h(a(i+i_o, j+j_o, k))+1$$

The result is a local feature composed of N values.
Recall that with histograms, we need around 8 samples per bin to have a low RMS error. Thus a good practice is to have N=W=H. For example N=4, W=4 and H=4. Many authors ignore this and use values such as N=8, W=4, H=4, resulting in a sparse histogram.

Remark: A fast version when N=4 replaces the inverse tangent by computing the diagonal derivatives with differences:

$$P_{\frac{\pi}{4}}(i,j,k) = P(i+1,j+1,k) - P(i-1,j-1,k)$$
$$P_{\frac{\pi}{2}}(i,j,k) = P(i,j+1,k) - P(i,j-1,k)$$
$$P_{\frac{3\pi}{4}}(i,j,k) = P(i+1,j-1,k) - P(i-1,j+1,k)$$
$$P_{\pi}(i,j,k) = P(i+1,j,k) - P(i-1,j,k)$$

To determine a(i,j,k) simply choose the maximum.

5. Scale Invariant Feature Transform (SIFT)

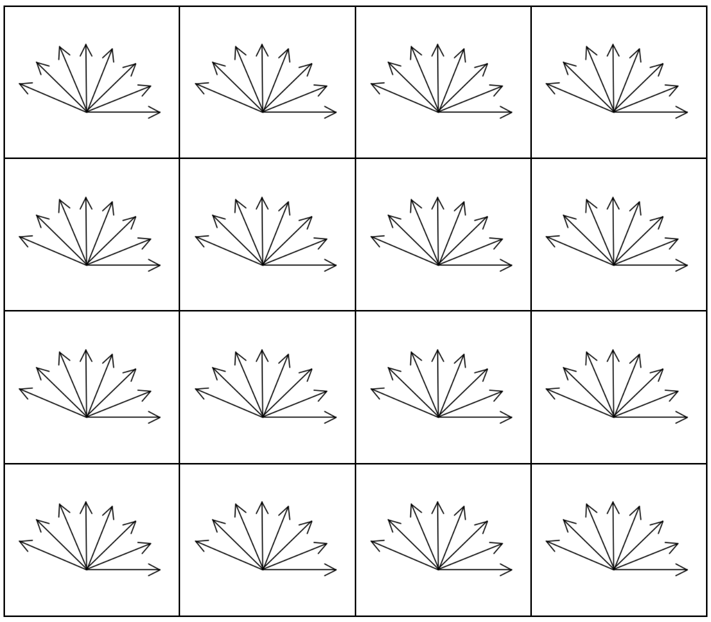
SIFT uses a scale invariant pyramid to compute a scale invariant DoG interest point detector to detect local scale-invariant interest points.

$$X(i,j,k) = \underset{i,j,k,R=2}{Local-\max}\{P(i,j,k) - P(i,j,k-1)\}$$

It then computes a U x V grid of HoG detectors with N=8, W=4, H=4 at the level k
Typically U=V=4.

At level k, $\Delta i = \Delta j = 2^{k/2}$

This gives 16 x 16 = 128 features at each interest point.
This feature vector is invariant to changes in position and scale and very robust with changes in image plane rotation and illumination intensity.



Various authors experiment with other grid sizes.

For example, let the grid size be G.

G=4, W=4, H=4, N=4

Gives 64 features.

6. Integral Images

An integral image, *ii(x,y)* of a window *p(x,y)* is the sum of all pixels from the upper left corner (1,1) to the current pixel (x,y).

$$ii(x,y) = \sum_{u=1,v=1}^{x,y} p(u,v)$$

A recurrence formula may be used to compute the integral image in 2 operations (memory access and additions) per pixel. This operation starts by computing an intermediate sum for each row:

$$s(x,y) = s(x,y-1) + i(x,y)$$

This intermediate sum is then used to compute the sum of rectangles.

$$ii(x,y) = ii(x-1,y)+s(x,y)$$

Thus the total cost of computing ii(x,y) for a window of size W·H is 2WH adds.

Any sum of the image within any rectangle from (x_1,y_1) to (x_2,y_2) can be computed as the sum or difference of 4 values:

$$r(x_1, y_1, x_2, y_2) = ii(x,y)–ii(x_1,y_2)–ii(x_2,y_1)+ii(x_1,y_1)$$

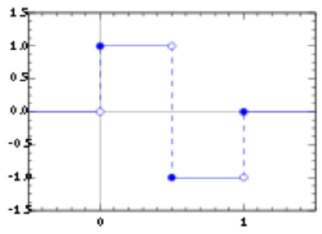
Thus integral images can be used to provide VERY fast computation of "box" features. These can be used to compute Haar wavelets.

6.1.1 Haar Wavelets

Haar A. Zur Theorie der orthogonalen Funktionensysteme, Mathematische Annalen, 69, pp 331–371, 1910.

A Haar wavelet is a difference of rectangular Windows.
In 1-D the definition is:

$$h(t) = \begin{cases} 1 & \text{for } 0 \le t < 0.5 \\ -1 & \text{for } 0.5 \le t < 1 \\ 0 & \text{for } t < 0 \text{ and } t \ge 1 \end{cases}$$



The Haar wavelet may be shifted by d and scaled by s

$$h(t;s,d) = h(t/s - d)$$

Note that the Haar Wavelet is zero gain (zero sum).

$$G = \int_{-\infty}^{\infty} h(t)dt = 0$$

The Digital (discrete sampled) form of Haar wavelet is

$$h(n;d,k) = \begin{cases} 1 & \text{for d} \le n < d + k/2 \\ -1 & \text{for d} + \text{k/2} \le n < d + k \\ 0 & \text{for } n < d \text{ and } n \ge d + k \end{cases}$$

Haar wavelets can be used to define an orthogonal transform analogous to the Fourier basis. This can be used to define an orthogonal transform (the Walsh-Hadamard Transform). In the 1980s the Wavelet community re-baptized the Haar functions as "wavelets" and demonstrated that the Walsh-Hadamard transform is the simplest form of wavelet transform. A 2-D form of Walsh-Hadamard transform may be defined using DoB features.