

Computer Vision

MoSIG M2

James L. Crowley

Fall Semester

14 November 2019

Lesson 5

Scale Invariant Pyramids, Interest Points and Descriptors

Lesson Outline:

1. Scale Space	2
1.1. Resolution, Scale, Sample Rate and Locality.....	2
1.2. Scale Space	2
1.3. Definition.....	3
1.4. Detecting Keypoints in Scale Space.....	4
1.5. Invariance to Changes in Position, Orientation and Scale.....	5
2. Image Pyramids	6
2.1. Spatial Resampling and Image Pyramids	6
2.2. Optimization of the Pyramid Algorithm.....	7
2.3. Scale Invariant Pyramids	8
2.4. Computing Image Derivatives with a Gaussian Pyramid.....	9
2.5. Color Opponent Receptive Fields.....	11
3. Scale Invariant Interest Points and the Laplacian profile	13
3.1. Natural Interest points from the Laplacian at half octave scales.....	14
3.2. Interest points from Gradient Magnitude	15
4. Histogram of Oriented Gradients (HOG)	16
5. Scale Invariant Feature Transform (SIFT)	17
6. Harris Corner Detector	18
7. Ridge Detection.....	20

1. Scale Space

1.1. Resolution, Scale, Sample Rate and Locality

In computer vision, the scale of an entity refers to its “spatial extent“ (or size). This can be formalized as the length of the largest line segment that can be contained within the entity. This is the physical locality over which an entity may be “seen”. The spatial extent (largest line segment) can be used to define the notion of locality. That is the spatial extent of the window that must be analyzed to recognize the entity.

However, this does not tell us how many pixels we need.

Resolution refers to the smallest distance for which two entities can be discriminated. The term refers to our ability to “resolve” separate entities.

Resolution determines the spatial sample rate required to represent the contents of the window. However, it is always possible to use more pixels than the required. The window can be oversampled. This does not necessarily add additional information.

Obviously, resolution and scale are related. But they are sometimes confused.

One way to think of this is to see scale as the physical size of the window that is required to see an entity, and resolution as the minimum number of pixels that must be used to represent the window.

1.2. Scale Space

In most natural images, information can be found at multiple scales. In general information at larger scales provides context for information at smaller scales.

To recognize information at multiple scales, we represent the image in a “scale space” and detect interest points that serve as anchor points for description and recognition.

Images at different scales can be represented with different resolutions (sample rates). The result is a multi-resolution pyramid representation of scale space. This can greatly accelerate recognition at larger scales.

1.3. Definition

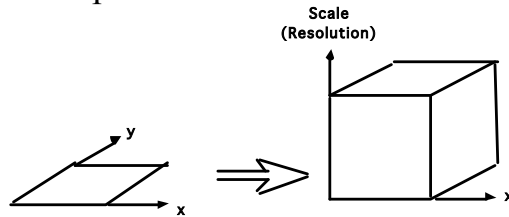
Let $P(x, y)$ be a 2-D image where (x, y) are real values,

Let $G(x, y, \sigma)$ be a Normalized Gaussian function of scale σ , with real valued σ .

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

The variable, σ , determines the “scale” of the Gaussian.

If we consider σ to be a free variable we can define a continuous 3D space $P(x, y, s)$. This is referred to as a “Scale Space”.



Formally, a scale space is the convolution of the image with Gaussians over a continuous range of scales, s .

$$P(x, y, s) = P(x, y) * G(x, y, \sigma_s) \text{ where } \sigma_s = \sigma_0^s$$

Recall that the definition of convolution for continuous x, y is:

$$P(x, y, s) = P(x, y) * G(x, y, \sigma_s) = \iint P(x-u, y-v) \cdot G(u, v, \sigma_s) du, dv$$

While scale space is formally defined for a continuous range of scales, it is typically computed at a discrete, exponential set of scales, σ_s

Invariance requires that these be sampled along a logarithmic scale.

That is, for some value of base scale, σ_0 , $\sigma_s = \sigma_0^s$

Typically the base value is $\sigma_0 = 2$, but any value greater than 1 may be used.

Values of $\sigma_0 = \sqrt{2}$ are also common.

1.4. Detecting Keypoints in Scale Space

Differential Geometry can be used to find invariant structures for matching and tracking in scale space. Such structures typically correspond to peaks and ridges in the first or second derivatives, but can also be found with zero crossings or with level crossing curves.

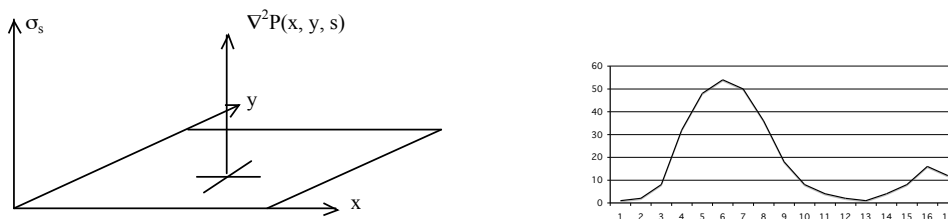
Local extrema (peaks) in the first or second derivative are commonly used to define “interest points” for tracking or matching. These can be computed using:

- 1) The Gradient Magnitude or
- 2) The Laplacian (second derivative).

The Laplacian of the image is $\nabla^2 P(x, y, s) = P * \nabla^2 G(x, y, \sigma_s) = P_{xx}(x, y, s) + P_{yy}(x, y, s)$

When evaluated over an exponential range of scales, the Laplacian gives a “laplacian profile. This is an invariant descriptor for appearance.

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



A change in viewing distance at x, y shifts the function $\nabla^2 P(x, y, s)$ in s . The form of the profile translates in scale but remains the same. Technically this is called a “covariant”.

A Laplacian interest point is found by $(x_i, y_i, s_i) = \underset{x, y, s}{\text{local-max}} \{ \nabla^2 P(x, y, s) \}$

Local-Max $\{ \}$ returns any point (x, y, s) for which the value of the function is larger than all neighbors within some distance ϵ .

1.5. Invariance to Changes in Position, Orientation and Scale

Landmarks, or keypoints, in scale space can be used to define structures. Keypoints are typically found as maximum in the Gradient (first derivative) or Laplacian (second derivative). Structures of keypoints can be used to prove a description of appearance that is invariant (or equi-variant) to changes in position, orientation and scale.

For example, translating a region of an image by Δx , Δy results in translation of the structural description of the region by Δx and Δy in scale space. This should be called equivariance. Similarly, rotation of the region by an angle results in a rotation of the structure of keypoints. Rotate the image of an object by θ in x , y and the structure rotates by θ in $P(x, y, s)$.

Structures of keypoints are also scale invariant. When an object in an image is made larger or smaller by a factor of $D = 2^d$

$$P(x,y) \rightarrow P(x2^d, y2^d)$$

Then the structure that describes the object is translated by d in the scale axis

$$P(x, y, s) \rightarrow P(x,y,s+d)$$

Note that the scale invariance requires that the scale axis be logarithmic. An exponential scale axis is necessary for scale invariant image description.

2. Image Pyramids

Scale space is an abstract mathematical construct. To compute the scale space representation of an image we must sample scale space in x , y and in s .

Let $P(x, y)$ be an image array of size $R \times C$ pixels, where (x, y) are integers instead of reals. A discrete scale space is:

$$P(x, y, k) = P(x, y) * G(x, y, \sigma_k)$$

where the convolution is defined using a sum:

$$P(x, y, \sigma_k) = P(x, y) * G(x, y, \sigma_k) = \sum_{u,v} P(x-u, y-v) \cdot G(u, v, \sigma_k)$$

For example, we can use a step size of $\sigma_0 = 2$ so that

$$\sigma_k = 2^k \quad \text{For } k=0 \text{ to } K.$$

At $k=0$: $\sigma_k = 2^0 = 1$. $\sigma=1$ is the smallest scale that we can represent.

The largest value, K , is determined by the size of the image.

Let $M = \min(R, C)$. $K = \text{Log}_2(M)$

For $k > K$ the scale parameter σ is larger than the image and the scale space signal rapidly converges to a constant value.

2.1. Spatial Resampling and Image Pyramids

Recall that $P(x, y, k)$ is defined for integer x, y, k .

The Gaussian, $G(x, y, \sigma)$, is a low pass filter.

As σ grows it becomes possible to resample the image with a larger step size without loss of information.

We can replace (x, y) with $(i \cdot \Delta x, j \cdot \Delta y)$.

$$G(x, y, \sigma_k) \rightarrow G(i \cdot \Delta x_k, j \cdot \Delta y_k, \sigma_k)$$

Where Δx_k and Δy_k are determined by σ_k

The images in the discrete scale space can be re-sampled using a sampling operation $S_{\Delta x}\{\}$

$$P(i, j, k) = S_{\Delta x}\{P(x, y, k)\} = P(i \Delta x_k, j \Delta y_k, k)$$

A resampled scale space is known as a pyramid, and can be written as

$$P(i, j, k) = P(i\Delta x_k, j\Delta y_k, k)$$

Shannon's sampling theory shows that the sample size at each $\Delta x_k, \Delta y_k$ can grow in proportion to σ_k . In the previous lecture we saw that $\sigma \geq 1$. This implies that for any value of k , $\sigma_k \geq \Delta x_k$.

It is common to use $\Delta x_k = \sigma_k = 2^k$. However, while this keeps the algebra simple, using, $\Delta x_k = \sigma_k$ results in substantial noise from aliasing.

This can be remedied by using $\sigma_k = 2^{k+1}$ and $\Delta x_k = 2^k$.

2.2. Optimization of the Pyramid Algorithm

A direct computation of scale space requires convolving the image with a series of exponentially larger Gaussian Filters:

$$P(x, y, k) = \sum_{u=-R_k}^{R_k} \sum_{v=-R_k}^{R_k} P(u-x, v-y)G(u, v, \sigma_k)$$

Where $R_k = 3\sigma_k$ giving an $N \times N$ computation where $N = 2R_k + 1 = 6\sigma_k + 1$. Each convolutions costs $O(N^2)$ operations.

We can gain some improvement by noting that the Gaussian is separable, giving a convolution that costs $2N$ rather than N^2 .

$$P(x, y, k) = P(x, y) * G(x, y, \sigma_k) = P(x, y) * G(x, \sigma_k) * G(y, \sigma_k)$$

However, for $\sigma_k = 2^{k+1}$ we still have an exponential growth $\sigma_k = (2, 4, 8, 16, \dots)$. For an image of $R \times C$ pixels, where $C > R$ this still requires $O(R^2)$ operations. For a 2^{10} by 2^{10} image, this is $O(2^{20})$ operations.

To overcome this exponential growth, we can use each level in the pyramid to compute the next level. This yields a form of recursive algorithm referred to as "cascade convolution". To develop this algorithm, we call the scale property of Gaussian Filters from lecture 4.

$$G(x, \sqrt{2}\sigma) = G(x, \sigma) * G(x, \sigma) = G(x, \sigma)^{*2}$$

Where the exponent " $*2$ " represents the convolution of $G(x, \sigma)$ with itself. To double the scale of a Gaussian we need a cascade of 4 convolutions:

$$G(x, 2\sigma) = G(x, \sigma) * G(x, \sigma) * G(x, \sigma) * G(x, \sigma) = G(x, \sigma)^{*4}$$

This says that we can use level k of the pyramid to compute level $k+1$.

$$P(x, y, k+1) = P(x, y, k) * G(x, y, \sigma_k)^{*4} = P(x, y) * [G(x, \sigma_k)^{*4}] * [G(y, \sigma_k)^{*4}]$$

However, we still have an exponential growth in σ_k . To reduce this we note that there is no need to compute the convolution at image positions that are removed by resampling.

This yields the optimized pyramid algorithm: Cascade convolution with resampling:

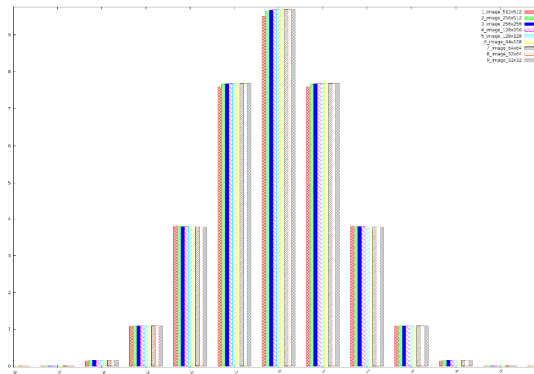
$$P(i, j, k+1) = P(2i, 2j, k) * G(x, y, \sigma_0)^{*4}$$

The computational cost of this algorithm is linear with the number of rows and columns.

2.3. Scale Invariant Pyramids

Resampling $P(x, y, k)$ at $\Delta x_k \sim \sigma_k$ results an identical impulse response at each level.

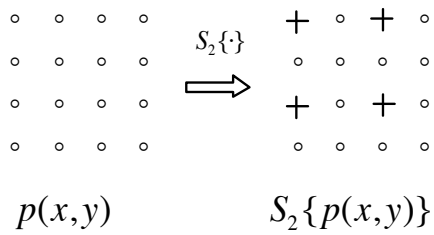
An impulse response is the output of a convolution when the input is a single impulse (Dirac Delta function). The following figure show the “impulse response” for a Scale Invariant pyramid at 6 different scales, computed using a fast $O(N)$ algorithm.



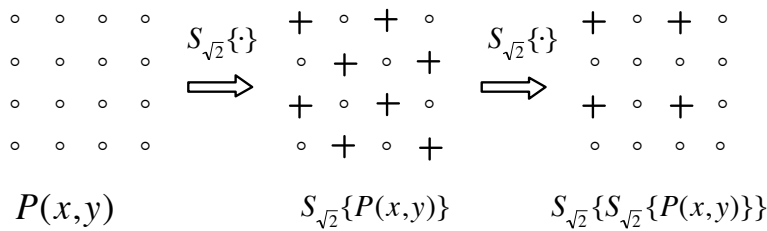
Pyramid samples are at discrete positions $(i\Delta x_k, j\Delta x_k)$ for integer values of i, j :

$$P(i, j, k) = P(i\Delta x_k, j\Delta x_k, k)$$

The position in the original image of a sample from level k is $x = i\Delta x_k$ and $y = j\Delta y_k$. If we sampling at a scale step of $\Delta x_k = 2^k$ this gives a "full octave" pyramid.



It is also possible to build a scale invariant pyramid with a step size of $\Delta x_k = 2^{k/2}$ using $\Delta x_k = 2^{k/2}$. This is known as a “half-octave” pyramid.



The Half-octave pyramid requires a re-sampling algorithm is somewhat complex and beyond the scope of this class. For today we will use a full octave pyramid to explain the principles of pyramids and scale space.

Note that for an image of size $N=R \times C$ pixels, direct computation of an image pyramid costs $O(N \log N)$ operations because the pyramid is composed of $\log(N)$ resampled image, and each costs $O(N)$ to compute. However a very fast recursive algorithm with cost of $O(N)$ exists. This is also beyond the scope of today’s class.

2.4. Computing Image Derivatives with a Gaussian Pyramid

It is possible to use the Gaussian Pyramid to compute image derivatives at scale using sums and differences. These are very similar to the receptive fields observed in the visual cortex of mammals.

Let $P(x, y)$ be an image array of size $R \times C$ pixels, where (x, y) are integers,

A full octave Gaussian pyramid of the image is a resampled set of images resulting from the convolution of image with Gaussians at an exponential set of scales.

$$P(x, y, k) = P * G(x, y, \sigma_k)$$

$$P(i, j, k) = P(i\Delta x_k, j\Delta y_k, k)$$

$$\text{where } \Delta x_k = \Delta y_k = 2^k \text{ and } \sigma_k = 2^{k+1} \text{ and } x = i\Delta x_k \text{ and } y = j\Delta y_k$$

For any sample $P(i, j, k)$ the position in the original image is $x = i\Delta x_k$ and $y = j\Delta y_k$

A pyramid of image derivatives can be defined as the convolution of image with derivatives of Gaussians

$$P_x(x,y,k) = P * G_x(x,y,\sigma_k)$$

With the Gaussian Pyramid, we can obtain a fast approximation for Gaussian derivatives by sum and difference of the samples of the Gaussian pyramid.

Then can approximate the Gaussian Derivatives at each scale using sums and differences of samples of the pyramid:

$$P_x(i,j,k) \approx P(i+1,j,k) - P(i-1,j,k) = P(i,j,k) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

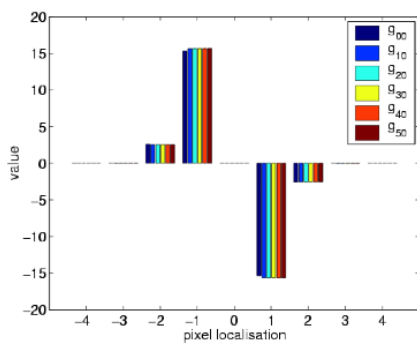
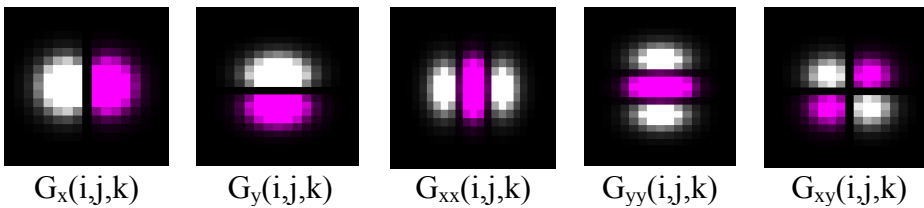
$$P_y(i,j,k) \approx P(i,j+1,k) - P(i,j-1,k) = P(i,j,k) * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

$$P_{xx}(i,j,k) \approx P(i+1,j,k) - 2P(i,j,k) + P(i-1,j,k) = P(i,j,k) * \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

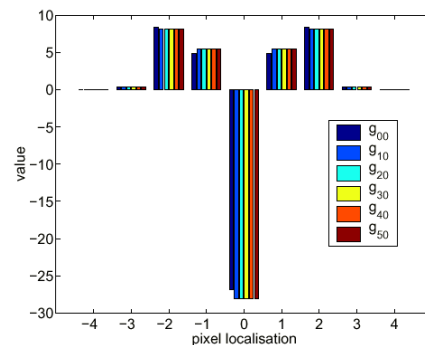
$$P_{yy}(i,j,k) \approx P(i,j+1,k) - 2P(i,j,k) + P(i,j-1,k) = P(i,j,k) * \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$P_{xy}(i,j,k) \approx P(i+1,j+1,k) - P(i-1,j+1,k) - P(i+1,j-1,k) + P(i-1,j-1,k) = P(i,j,k) * \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

The following are some numerically evaluated examples published in the Scale Space Conference of 2003. [Crowley-Riff 2003]



Impulse Response for $G_x(i,j,k)$



Impulse Response for $G_{xx}(i,j,k)$

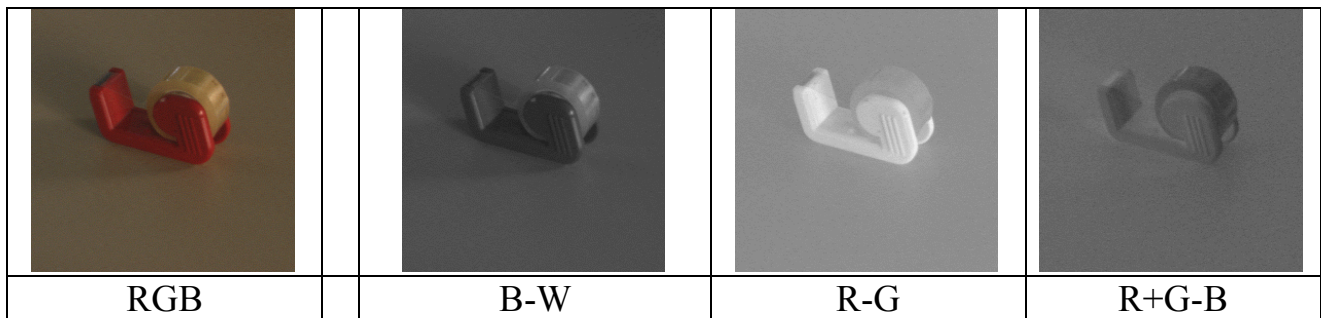
Impulse response for Gaussian derivatives for pyramid levels $k=0,1,2,3,4,5$.

2.5. Color Opponent Receptive Fields

A color opponent space is useful for illumination invariance

$$(R, G, B) \Rightarrow (L, C_1, C_2) \quad \begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 0.33 & 0.33 & 0.33 \\ -0.5 & -0.5 & 1 \\ 0.5 & -0.5 & 0 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

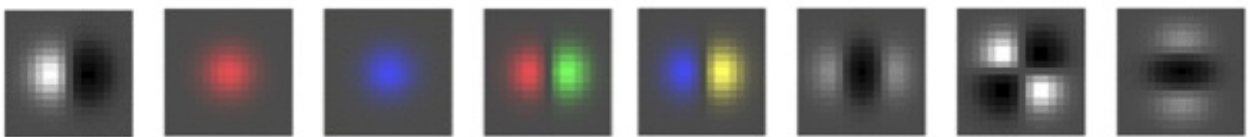
This representation separates luminance and chrominance.



Color opponent space can be used to build color opponent receptive fields.

$$\begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 0.33 & 0.33 & 0.33 \\ -0.5 & -0.5 & 1 \\ 0.5 & -0.5 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 R \\ \alpha_2 G \\ \alpha_3 B \end{pmatrix}$$

We then compute 3 pyramids: $L(i, j, k)$, $C_1(i, j, k)$, and $C_2(i, j, k)$,



Examples of color opponent receptive fields.

Color opponent receptive fields can be steered in color to provide color invariance. This gives us a feature vector for local appearance:

$$\vec{A}(x, y, k) = \begin{bmatrix} G_x^{L\sigma_k} \\ G^{C_1\sigma_k} \\ G^{C_2\sigma_k} \\ G_x^{C_1\sigma_k} \\ G_x^{C_2\sigma_k} \\ G_{xx}^{L\sigma_k} \\ G_{xy}^{L\sigma_k} \\ G_{yy}^{L\sigma_k} \end{bmatrix}$$

This can be generalized to include multiple scales and higher order derivatives.

The result is tensor who dimensions as $x, y, c, \sigma, \theta, d$

where c represents the color channel (L, C_1 or C_2) and d represents the derivative order.

3. Scale Invariant Interest Points and the Laplacian profile

Scale invariant interest points (or feature points) can be computed within a Pyramid with a Gradient (First derivative) or a Laplacian (Second derivative).

Recall that the Laplacian of the image is

$$\nabla^2 P(x, y, s) = P * \nabla^2 G(x, y, \sigma_s) = P_{xx}(x, y, s) + P_{yy}(x, y, s)$$

As described above, a Laplacian profile for an image point is the Laplacian of the image computed over a continuous (exponential) range of scales.

A Laplacian interest point is $(x_i, y_i, s_i) = \underset{x, y, s}{local} - \max\{\nabla^2 P(x, y, s)\}$

Such interest points can be computed directly from a difference of levels in the Gaussian pyramid. For a Gaussian Scale Space, we can show that:

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

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

This is called a "Difference of Gaussians" (DoG) and requires $\sigma_1 \geq \sqrt{2} \sigma_2$. Thus the Laplacian of the image can be approximated as the difference at adjacent pyramid levels from a Gaussian Pyramid.

$$\nabla^2 P(x, y, k) = P(x, y, k) - P(x, y, k-1)$$

We can detect scale invariant interest points local maxima in the Laplacian using the DoG:

$$x_i, y_i, s_i = \underset{x, y, k}{local} - \max\{P(x, y, k) - P(x, y, k-1)\}$$

Note that the points must be at the same image position (x,y). If the pyramid is resampled then we cannot simply use (i,j). We must use $x = i\Delta x_k$ and $y = j\Delta y_k$ to assure that the sample are at the same image position.

A difference in scale in the pyramid levels is $\Delta\sigma=2$ is usable but sensitive to noise. A value of $\Delta\sigma=\sqrt{2}$ is more reliable.

3.1. Natural Interest points from the Laplacian at half octave scales.

David Lowe used the scale invariant pyramid to define “Natural” interest points. These are used in the SIFT image descriptor.
(SIFT = Scale Invariant Feature Transform)

To obtain a natural interest point with a scale precision of less than $\Delta\sigma=2$ we can use cascade convolution within each pyramid level.

Consider a pyramid image at level k : $P(i, j, k)$ with $\sigma_k : \Delta\sigma=2$ and $\sigma_k=2^k$

The resolution at $P(i, j, k)$ is $\sigma_k = 1$ (relative to i, j)
we and compute:

$$\begin{aligned} P(i, j, k) & & \sigma_k &= 1 \\ P_1(i, j, k) &= P * G(i, j, 1) & \sigma_{k1} &= \sqrt{2} \\ P_2(i, j, k) &= P_1 * G(i, j, \sqrt{2}) = P_1 * G(i, j, 1) * G(i, j, 1) & \sigma_{k2} &= 2 \\ P_3(i, j, k) &= P_2 * G(i, j, 2) & \sigma_{k2} &= 2\sqrt{2} \\ P_4(i, j, k) &= P_3 * G(i, j, 2\sqrt{2}) = P_3 * G(i, j, 2) * G(i, j, 2) & \sigma_{k2} &= 4 \end{aligned}$$

We can then calculate 4 Laplacian values:

$$\begin{aligned} L_{k0} &= P_1(i, j, k) - P(i, j, k) \\ L_{k1} &= P_2(i, j, k) - P_1(i, j, k) \\ L_{k2} &= P_3(i, j, k) - P_2(i, j, k) \\ L_{k3} &= P_4(i, j, k) - P_3(i, j, k) \end{aligned}$$

If $P(i, j, k)$ is a local max in i and j :

$$P(i-1, j, k) < P(i, j, k) > P(i+1, j, k) \text{ and } P(i, j-1, k) < P(i, j, k) > P(i, j+1, k)$$

Then

If $L_{k0} < L_{k1} > L_{k2}$ then the point $P(i, j, k)$ is a natural interest point at with $\sigma = \sqrt{2} \sigma_k$

If $L_{k1} < L_{k2} > L_{k3}$ then the point $P(i, j, k)$ is a natural interest point with $\sigma = 2 \sigma_k$

The position of the natural interest point is $x = i \cdot 2^k, y = j \cdot 2^k$

This is similar to the method used to find natural interest points in the SIFT detector.

3.2. Interest points from Gradient Magnitude

We can also compute an intrinsic scale for the Gradient magnitude.

$$\text{The Gradient } \vec{\nabla}P(x,y,s) = \begin{pmatrix} P_x(x,y,s) \\ P_y(x,y,s) \end{pmatrix} = \begin{pmatrix} P * G_x(x,y,s) \\ P * G_y(x,y,s) \end{pmatrix}$$

For any image point (x,y) the intrinsic scale can be computed from

$$s_i = \text{local} - \max_s \{ \|\vec{\nabla}P(x,y,s)\| \}$$

These are positions in the image that can serve as landmarks for tracking or recognition.

In a scale-invariant pyramid, the gradient is available at any sample in the pyramid as

$$\vec{\nabla}P(i,j,k) = \begin{pmatrix} P_x(i,j,k) \\ P_y(i,j,k) \end{pmatrix} = \begin{pmatrix} P(i+1,j,k) - P(i-1,j,k) \\ P(i,j+1,k) - P(i,j-1,k) \end{pmatrix}$$

For the image gradient, a scale invariant interest point is

$$i_i, j_i, k_i = \text{Local} - \max_{i,j,k} \{ \|\vec{\nabla}P(i,j,k)\| \}$$

4. Histogram of Oriented Gradients (HOG)

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 (currently at UGA Labo LJK).

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

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

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

$$a(i,j,k) = \text{Round} \left\{ N \cdot \frac{\theta(i,j,k)}{\pi} \right\}$$

We can then build a local histogram for a window of size $W \times H$, 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:

$$\sum_{i=1}^W \sum_{j=1}^H h(a(i+i_o, j+j_o, k)) = h(a(i+i_o, j+j_o, k)) + 1$$

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:

$$\begin{aligned} P_0(i,j,k) &= P(i+1,j,k) - P(i-1,j,k) \\ P_{\frac{\pi}{4}}(i,j,k) &= P(i+1,j+1,k) - P(i-1,j-1,k) \\ P_{\frac{\pi}{2}}(i,j,k) &= P(i,j+1,k) - P(i,j-1,k) \\ P_{\frac{3\pi}{4}}(i,j,k) &= P(i+1,j-1,k) - P(i-1,j+1,k) \end{aligned}$$

To determine $a(i,j,k)$ simply choose the maximum of $P_0, P_{\frac{\pi}{4}}, P_{\frac{\pi}{2}}, P_{\frac{3\pi}{4}}$

5. Scale Invariant Feature Transform (SIFT)

SIFT uses a scale invariant pyramid to compute scale invariant interest points as shown above.

$$L_{k0} = p_1(i, j, k) - p(i, j, k)$$

$$L_{k1} = p_2(i, j, k) - p_1(i, j, k)$$

$$L_{k2} = p_3(i, j, k) - p_2(i, j, k)$$

$$L_{k3} = p_4(i, j, k) - p_3(i, j, k)$$

If $L_{k0} < L_{k1} > L_{k2}$ then the point $p(i, j, k)$ is a natural interest point at with $\sigma = 2^{k+1/2}$

If $L_{k1} < L_{k2} > L_{k3}$ then the point $p(i, j, k)$ is a natural interest point with $\sigma = 2^{k+1}$

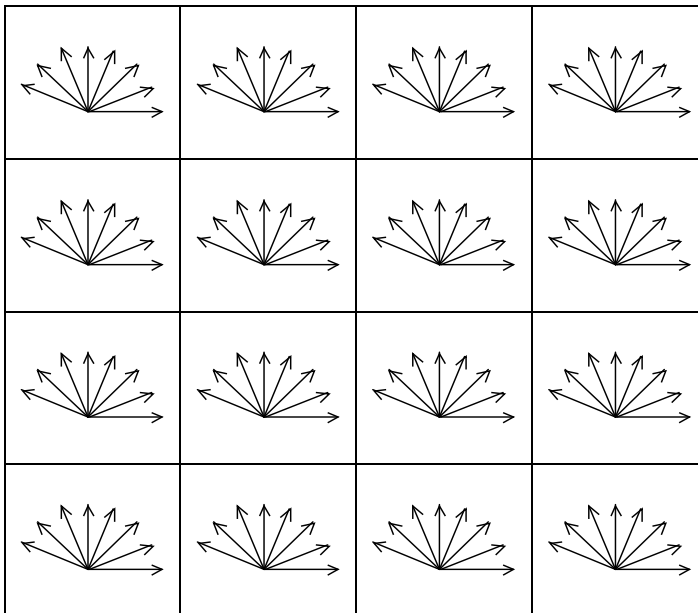
For each interest point, it then computes a $U \times V$ grid of HOG detectors with $N=8$, $W=4$, $H=4$ at the level k

Typically $U=V=4$.

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

This gives $16 \times 8 = 128$ 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 .

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

16 histograms of 4 numbers = 64.

6. Harris Corner Detector

Harris, Chris, and Mike Stephens. "A combined corner and edge detector." Alvey vision conference. Vol. 15. 1988.

The Harris-Stevens Corner detector is inspired from the Moravec Interest Point detector proposed in 1973 by Hans Moravec for stereo matching. Moravec used the Sum of Squared Difference (SSD) between adjacent small patches to detect interest points. In 1988, Harris and Stevens observed that this is equivalent to an auto-correlation of the image.

$$S(x,y) = \sum_{u,v} w(u,v) (I(u+x,v+y) - I(u,v))^2$$

where $I(x,y)$ is the image,
 $w(x,y)$ is some window function, typically Gaussian.

$I(u+x,v+y)$ can be approximated as a local Taylor Series:

$$I(u+x,v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

where $I_x(x,y)$ and $I_y(x,y)$ are the local x and y derivatives

Giving
$$S(x,y) = \sum_{u,v} w(u,v) (I_x(u,v)x + I_y(u,v)y)^2$$

Which can be written in Matrix form as: $S(x,y) \approx (x \ y) A \begin{pmatrix} x \\ y \end{pmatrix}$ where A is the "Structure Tensor"

$$A = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

With our Gaussian pyramid this is simply: $A = \begin{bmatrix} P_x^2 & P_x P_y \\ P_x P_y & P_y^2 \end{bmatrix}$

Compute the Eigenvectors of A: $\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} = R A R^T$

where λ_1 is the maximum gradient, λ_2 is the minimum gradient.

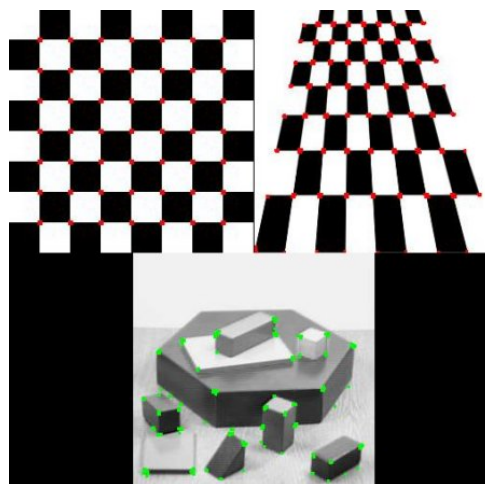
if $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$ then the point is of no interest
 if $\lambda_1 \approx 0$ and $\lambda_2 \gg 0$ then the point is a horizontal edge
 if $\lambda_1 \gg 0$ and $\lambda_2 \approx 0$ then the point is a vertical edge
 if $\lambda_1 \approx \lambda_2 \gg 0$ then the point is corner

To avoid computing the eigenvalues (requires a square root), we can define a measure for “corner-ness”:

$$M_c = \det(A) - \kappa \cdot \text{Trace}^2(A) = \lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2$$

where κ is a tunable sensitivity parameter.

Examples of Harris-Stevens Corners:



7. Ridge Detection.

The Eigenvalues of the Hessian provide a popular ridge detector.

The Hessian at scale s is $H(x,y,s) = \begin{pmatrix} P_{xx}(x,y,s) & P_{xy}(x,y,s) \\ P_{xy}(x,y,s) & P_{yy}(x,y,s) \end{pmatrix}$

The Eigenvalues are found by diagonalizing the Hessian.

For any point in scale space (x, y, s)

$$\begin{pmatrix} P_{rr} & 0 \\ 0 & P_{ss} \end{pmatrix} = R H R^T \quad \text{where} \quad R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

P_{ss} is the largest value in second derivative, while P_{rr} is the smallest.

On a ridge point, P_{rr} will be the second derivative along the ridge (close to zero) while P_{ss} will be the 2nd derivative perpendicular to the ridge.

For any 2D Matrix, the principal directions can be computed directly as

$$\cos(\theta) = \sqrt{\frac{1}{2} \left(1 + \frac{P_{xx} - P_{yy}}{\sqrt{(P_{xx} - P_{yy})^2 + 4P_{xy}^2}} \right)}, \quad \sin(\theta) = \text{sgn}(P_{xy}) \sqrt{\frac{1}{2} \left(1 - \frac{P_{xx} - P_{yy}}{\sqrt{(P_{xx} - P_{yy})^2 + 4P_{xy}^2}} \right)}$$

Recall that the gradient is $\vec{\nabla}P(x,y,s) = \begin{pmatrix} P_x(x,y,s) \\ P_y(x,y,s) \end{pmatrix} = \begin{pmatrix} P^* G_x(x,y,s) \\ P^* G_y(x,y,s) \end{pmatrix}$

for any point (x, y, s) , the Gradient can be aligned with the ridge using

$$\begin{aligned} P_r &= \cos(\theta)P_x - \sin(\theta)P_y \\ P_s &= \sin(\theta)P_x + \cos(\theta)P_y \end{aligned}$$

A positive ridge point is any point, $R(x,y,s)$ that satisfies:

$$P_r = 0 \text{ and } P_{rr} \leq 0 \text{ and } |P_{rr}| \geq |P_{ss}|$$

A negative ridge is any point for which

$$P_r = 0 \text{ and } P_{rr} \geq 0 \text{ and } |P_{rr}| \leq |P_{ss}|$$

of course, P_r will rarely be exactly zero, so we use form of approximation $P_r \approx 0$

The ridge direction at (x, y, s) is: $\cos(\theta) = \frac{P_x}{\sqrt{P_x^2 + P_y^2}}$ $\sin(\theta) = \frac{P_y}{\sqrt{P_x^2 + P_y^2}}$

A Maximal ridge is a ridge point $R(x, y, s)$ for which $\text{local}_s - \max\{\nabla^2 P(x, y, s)\}$

Examples of Maximal Ridge points:

