ECE 661
Purdue U.
Avi Kak
Lecture 9

Extracting Interest Points and Their Descriptors (with Harris, SIFT, and SURF) in Image Pairs and Establishing Point-to-Point Correspondences Between the Images

- My goal is to first introduce you to the notion of scale space that plays an important role in the more modern operators like SIFT and SURF for extracting interest points that are significantly invariant to scale, orientation, and illumination changes.

- Recall from Lecture 8 the LoG operator for edge detection. What the operator does amounts to first smoothing the image with a Gaussian function $\boxed{g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}}$ and then applying the Laplacian $\boxed{\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}}$ to the smoothed image. Obviously, the larger the value of σ, the greater the smoothing effect, and the greater the suppression of small and fast changes in image gray levels. In other words, the larger the value of σ, the larger the scale at which the gray levels must change in order to be detectable by the LoG operator.

- When any form of image representation depends on σ, the set of the representations that can be calculated for all possible values of σ constitutes the scale space of the image.

- Here is an important result from the scale-space theory for images: For a given image $f(x,y)$, let $ff(x,y,\sigma)$ represent its σ-smoothed version. That is, $\boxed{ff(x,y,\sigma) = \iint_{-\infty}^{\infty} f(x',y')\, g(x-x',y-y')\, dx'dy'}$ where $g(x,y)$ is the Gaussian that was shown previously. The important result is $\boxed{\frac{\partial}{\partial\sigma} ff(x,y,\sigma) = \sigma \nabla^2 ff(x,y,\sigma)}$. Since $\nabla^2 ff(x,y,\sigma)$ is the LoG of $f(x,y)$, we can write $\boxed{LoG(f(x,y)) = \frac{\partial}{\partial\sigma} ff(x,y,\sigma)}$

- The above result implies that the LoG of an image $f(x,y)$ can be approximated by subtracting the $(\sigma+\delta\sigma)$-smoothed version of $f(x,y)$ from the σ-smoothed version. The difference of two Gaussian-smoothed versions of $f(x,y)$ for two different values of σ is referred to as the Difference-of-Gaussian (DoG). The fact that LoG can be approximated by DoG gives you a clue to the power of scale-space analysis.

- To prove $\boxed{\frac{\partial}{\partial\sigma} ff(x,y,\sigma) = \sigma \nabla^2 ff(x,y,\sigma)}$, first recall from Lecture 8 that $\boxed{\nabla^2 ff(x,y,\sigma) = f(x,y) * h(x,y,\sigma)}$ where $\boxed{h(x,y,\sigma) = -\frac{1}{2\pi\sigma^4}\left(2 - \frac{x^2+y^2}{\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}}$.
Let's now examine $\frac{\partial}{\partial\sigma} ff(x,y,\sigma)$. We can write $\boxed{\frac{\partial}{\partial\sigma} ff(x,y,\sigma) = \iint f(x',y') \frac{\partial}{\partial\sigma}\left\{\frac{1}{2\pi\sigma^2} e^{-\frac{(x-x')^2+(y-y')^2}{2\sigma^2}}\right\} dx'dy'}$
The right hand side can be re-expressed as
$$\boxed{\frac{\partial}{\partial\sigma} ff(x,y,\sigma) = \iint f(x',y') \left[\frac{-1}{\pi\sigma^3} + \frac{1}{2\pi\sigma^2}\left(-[(x-x')^2+(y-y')^2]\right)\frac{-2}{2\sigma^3}\right] e^{-\frac{(x-x')^2+(y-y')^2}{2\sigma^2}} dx'dy'}$$
which simplifies to
$$\boxed{\frac{\partial}{\partial\sigma} ff(x,y,\sigma) = -\frac{\sigma}{2\pi\sigma^4}\iint f(x',y')\left[2 - \frac{(x-x')^2+(y-y')^2}{\sigma^2}\right] e^{-\frac{(x-x')^2+(y-y')^2}{2\sigma^2}} dx'dy'} = \sigma f(x,y) * h(x,y)$$

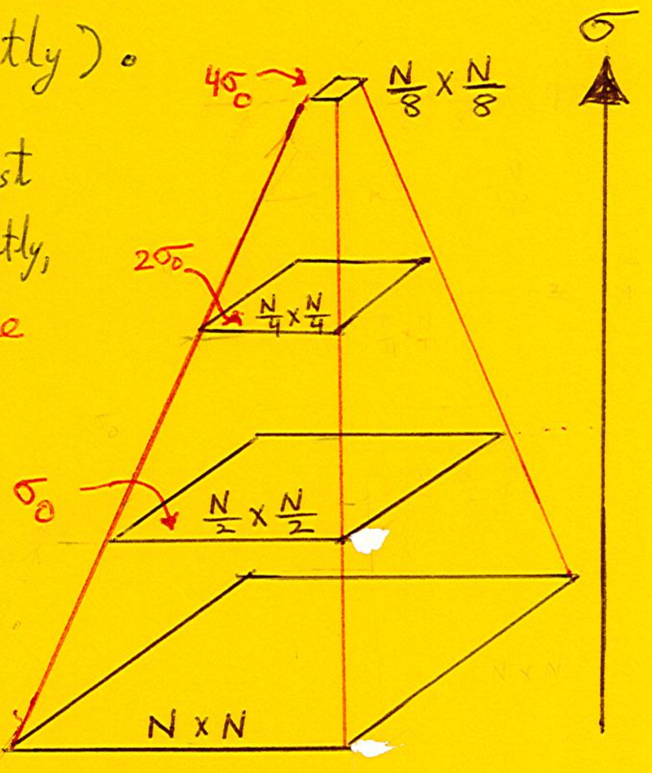
- The fact that we can approximate LoG[f(x,y)] by DoG[f(x,y)], while giving us a window into the power of scale-space analysis, has some pretty amazing computational consequences: It is much faster to calculate DoG with σ_1 and σ_2 for the two Gaussian smoothings than to calculate LoG for any given σ. As for the reason, since the Gaussian $g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$ is separable in x and y, each of the 2-D smoothings for DoG can be carried out by two applications of 1-D smoothings, first along x and then the result along y. This cannot be done for the LoG operator $h(x,y)$ since it is not separable.

- Additionally, for digital implementation, you are likely to use a smaller sized operator for DoG than for LoG. Assuming a unit distance for the pixel sampling interval, let $\sigma = \sqrt{2}$. As you saw in Lecture 8, the digital version of LoG requires a 13x13 operator. With a DoG based implementation, you can make do with just a 9-element 1-D smoothing window. [Even for the same calculation, the width of a 1-D operator is smaller than the width of a 2-D operator. For example, for a 1-D LoG, you'll use $g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$. For this, we have $h(x,\sigma) = \boxed{\frac{\partial^2 g}{\partial x^2} = -\frac{1}{\sqrt{2\pi}\sigma^3}\left[1 - \frac{x^2}{\sigma^2}\right] e^{-x^2/2\sigma^2}}$, which has a half-width of σ for the central lobe. (Compare that to the radius $\sqrt{2}\,\sigma$ for the central lobe for 2D.) Again using the criterion that the half-width of the overall operator be three times the half-width of the central lobe, we get 3σ for the half-width of the operator. When $\sigma = \sqrt{2}$, we get $2 \times 3 \times \sqrt{2} \approx 9$ pixels for the full width of the 1D LoG operator as opposed to 13x13 for the size of the 2-D LoG for the same σ.]

Image Pyramids for Scale-Space Operators

- Now that you know what a scale space is, let's talk about different kinds of pyramidal representations for an image. Pyramidal representations are at the heart of operators such as SIFT and SURF.

- The simplest such representations is the Gaussian Pyramid. One generates a Gaussian pyramid for an image by smoothing it with successively larger Gaussian functions (meaning, Gaussians with larger σ) and arranging the sequence of smoothed images in the form of a stack that looks like a pyramid (for reasons that will be clear shortly).



$4\sigma_0$
$\frac{N}{8} \times \frac{N}{8}$
σ
$2\sigma_0$
$\frac{N}{4} \times \frac{N}{4}$
σ_0
$\frac{N}{2} \times \frac{N}{2}$
N x N

- Let's say that, at the bottom of the pyramid, the first application of the Gaussian is with $\sigma = \sigma_0$. Subsequently, as you go from one level to the next, you double the size of σ. Additionally, after smoothing you downsample the resulting image by a factor of 2.

- An important practical reason for down sampling is that you can use the same MxM digital approximation to the Gaussian for all level-to-level transitions in the pyramid. The factor-of-2 downsampling can be achieved by retaining every other row and column in the image.

- The downsampling of each level by a factor of 2 compared to the level below can be theoretically justified by the Nyquist Theorem which implies that the highest frequency in a digital signal obtained by using a sampling interval of τ units is $\frac{1}{2\tau}$ cycles per unit distance. [If the original analog signal has frequencies greater than $\frac{1}{2\tau}$ cycles per unit length, then the digital signal will contain aliasing distortion within the $(-\frac{1}{2\tau}, \frac{1}{2\tau})$ frequency band it can support.]

- In order to appreciate the role played by the Nyquist Theorem in the justification for downsampling, note that the Fourier transform of the Gaussian $\boxed{g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}}$ is $\boxed{G(u,v) = \sqrt{2\pi\sigma^2}\, e^{-2\pi^2\sigma^2(u^2+v^2)}}$, where $\boxed{G(u,v) = \iint_{-\infty}^{\infty} g(x,y)\, e^{-j2\pi(ux+vy)}\, dx\, dy}$. Recognizing that the Fourier transform of a Gaussian is another Gaussian, let's rewrite the Fourier transform of $g(x,y)$ as $\boxed{G(u,v) \simeq e^{-\frac{u^2+v^2}{2\alpha^2}}}$ with the 'standard deviation' $\boxed{\alpha = \frac{1}{2\pi\sigma}}$. Truncating the radius of $G(u,v)$ at five times the 'standard deviation', we get for the highest frequency retained after smoothing as $\boxed{W = \frac{5}{2\pi\sigma}}$.

- Therefore, if we double the value of σ for smoothing, we halve the highest frequency in the sampling points if we maintain the same resolution as what existed before smoothing. Downsampling the smoothed image by a factor of 2 would re-establish the relationship (that existed prior to smoothing) between the highest frequency and the total number of samples. That would allow for the same Gaussian operator to be used to create the next level of the pyramid.

- The fact that each level of the pyramid is characterized by a σ that is twice the value used in the previous level can also be expressed by saying that each level of the Gaussian pyramid is one octave above the level below. [In music, if two notes are one octave apart, the pitch at the higher note is twice the pitch at the lower note.]

The Laplacian (Or the DoG) Pyramid

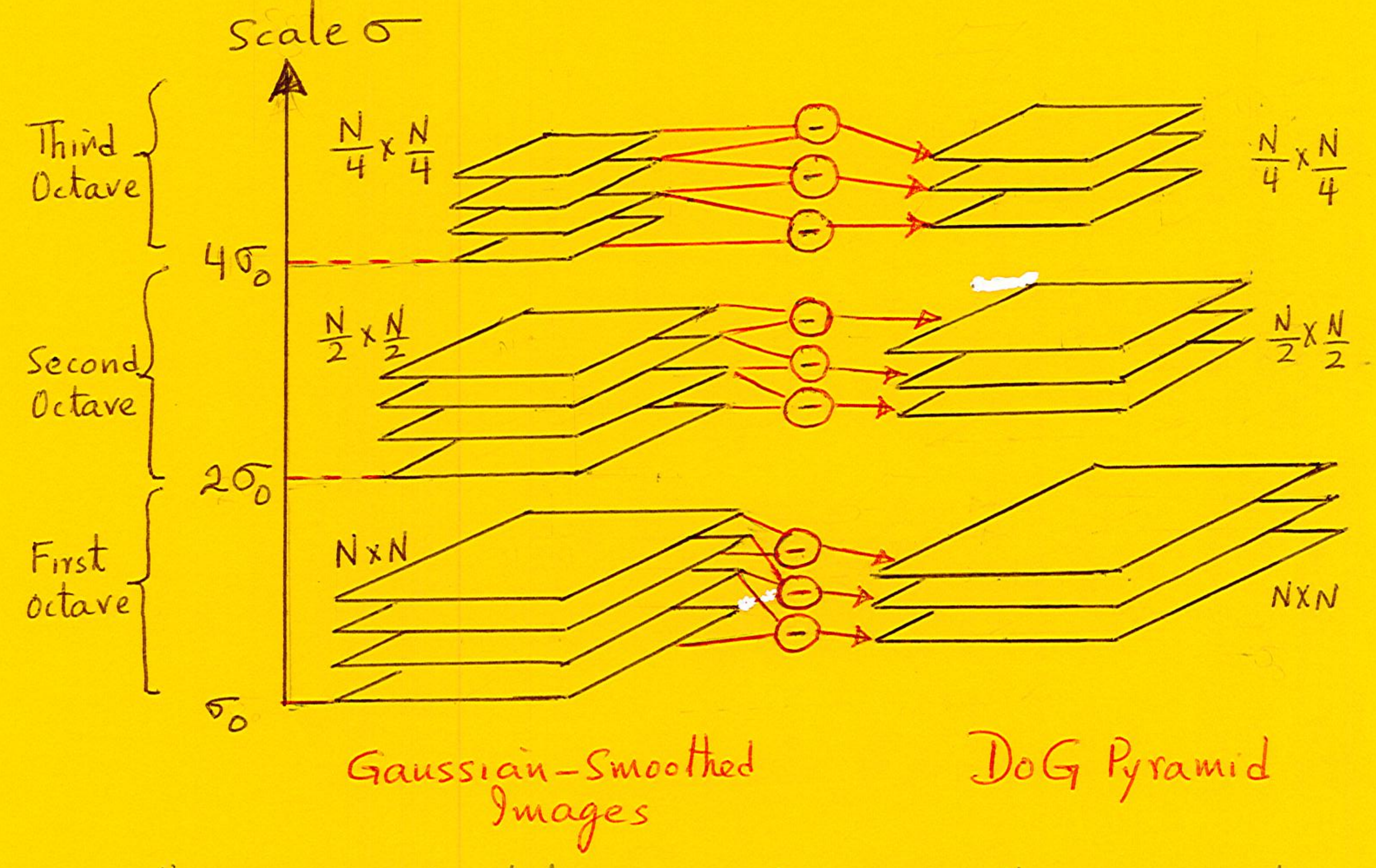
- The smoothed images that reside at the different levels of a Gaussian pyramid are digital versions of $ff(x,y,\sigma)$ we defined earlier for $\sigma = \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0, \ldots$

- Recall that a direct consequence of $\boxed{\frac{\partial}{\partial\sigma} ff(x,y,\sigma) = \sigma \nabla^2 ff(x,y,\sigma)}$ is that we can calculate the Laplacian of the image at scale σ by the DoG $\boxed{ff(x,y,\sigma_1) - ff(x,y,\sigma_2)}$ for two closely spaced scales σ_1 and σ_2. The Gaussian pyramid does not lend itself directly to the calculation of the Laplacians in this manner because the one-octave jump between two successive levels is much too large for the approximation we need for $\frac{\partial}{\partial\sigma} ff(x,y,\sigma)$.

- Therefore, for calculating the Laplacians through DoG on the pyramid, we must first calculate the Gaussian-smoothed images at additional scale values within each octave.

- Since the sampling points along σ constitute a geometric progression (the different levels of the Gaussian pyramid are for scale values $\sigma_0 \cdot 2^i$, $i = 1, 2, 3, \ldots$, which constitutes a geometric series), we would like to maintain the same form for the additional within-octave sampling of the scale space. This we can do by defining a within-octave scale sampling interval $\boxed{\Delta\sigma = \frac{\sigma_b}{2^s}}$, where σ_b is the value of σ at the base of the octave and where s is a user-specified integer. We now place the additional sampling points at $\boxed{\Delta\sigma \cdot 2^i \text{ for } i = 0, 1, 2, \ldots, s-1}$.

- Since the DoG calculation $\boxed{ff(x,y,\sigma_1) - ff(x,y,\sigma_2)}$ requires that the (x,y) coordinates at the point at which the difference is being computed be the same in the two Gaussian-smoothed images, we do not change the image resolution in the within-octave samplings of the scale space. Therefore, the scale-space would now look like as shown at left below:


Scale σ
Third Octave
Second Octave
First Octave
$4\sigma_0$
$2\sigma_0$
σ_0
$\frac{N}{4} \times \frac{N}{4}$
$\frac{N}{2} \times \frac{N}{2}$
N x N
$\frac{N}{4} \times \frac{N}{4}$
$\frac{N}{2} \times \frac{N}{2}$
N x N
Gaussian-Smoothed Images
DoG Pyramid

- If you want to use the same digital Gaussian smoothing operator at the within-octave levels that you do at the octave boundaries, you must first downsample the image in proportion to the scale factor, apply the operator, and, finally, upsample the result so that its resolution is restored. The downsampling (and upsampling) will involve calculating image gray levels at fractional coordinates — something that can easily be accomplished with, say, bilinear interpolation.

- As shown at right in the figure above, subtracting the adjacent Gaussian-smoothed images within the same octave gives us the Laplacian at the scale that corresponds to the Gaussian-smoothed images.

SIFT for Extracting Interest Points and Their Descriptors

Referring to the DoG pyramid, let $\boxed{D(x,y,\sigma)}$ denote the DoG values at scale σ. Shown below are the steps that go into the extraction of the SIFT features:

SIFT stands for Scale Invariant Feature Transform

STEP 1:

- Find all the local extrema in the DoG pyramid. That is, we find points that are either locally maximum or locally minimum in the three dimensional space (x,y,σ). Each point in the DoG pyramid is compared with: (a) the 8 points in the immediate 3x3 neighborhood at the same scale; (b) the 9 points in the 3x3 neighborhood in the DoG that is at the next level up in the scale space; and (c) the 9 points in the 3x3 neighborhood that is at the level just below. Therefore, for a point in the pyramid to pass the local extremum test, it must be compared with 26 neighbors in a 3x3x3 "volumetric" neighborhood.

- Obviously, $D(x,y,\sigma)$ must be known for at least 3 values of σ in an octave for the extrema to be identified. It is recommended that you find the extrema for at least two different values of scale σ in each octave. For that we need $D(x,y,\sigma)$ for at least 4 values of σ. That in turn implies that we need Gaussian smoothed images at at least five different scales in each octave.

- A good question to ask at this point is: What gray level distributions in the original image give rise to extrema in the DoG pyramid? For a general answer, points in the original image where the gray levels change rapidly in several directions are likely to give rise to DoG extrema. For a more specific answer, consider an image whose gray levels are all 0 except for a circle of diameter 32 pixels where the gray levels are all 1. Now consider convolving this image with the LoG operator at scale σ. As you'll recall, the central lobe of this operator is of diameter $2\sqrt{2}\sigma$. So for the scale value $\sigma = 32/2\sqrt{2}$, the central part of the operator will match the size of the circle and the result of the convolution will be the largest value as the operator is right on top of the circle part of the image.

STEP 2:

- As σ increases, and especially when you go from a lower octave to a higher octave, the individual discrete points of DoG represent increasingly coarse sampling of the original image. An extremum located at a scale that is coarse may not point directly to a pixel in the original image that should represent a SIFT point. We can achieve greater precision in the localization of the extremum vis-a-vis the original image if we improve the location of the extremum in the DoG pyramid. As it turns out, if you can estimate the second-order derivatives of D at the sampling points in the DoG pyramid, you can localize the extremum with "sub-pixel" accuracy in the vicinity of where the extremum was found in Step 1. This is how you do it: The Taylor series expansion (for a scalar function of multiple variables) of $D(x, y, \sigma)$ in the vicinity of $\boxed{\vec{x}_0 = (x_0, y_0, \sigma_0)^T}$, which was found to be the location of an extremum in step 1, can be expressed as

$$\boxed{D(\vec{x}) \simeq D(\vec{x}_0) + J^T(\vec{x}_0)\vec{x} + \frac{1}{2}\vec{x}^T H(\vec{x}_0)\vec{x}}$$

where $\vec{x}$ is the incremental deviation from $\vec{x}_0$, and where J is the gradient vector estimated at $\vec{x}_0$: $\boxed{J(\vec{x}_0) = \left(\frac{\partial D}{\partial x}, \frac{\partial D}{\partial y}, \frac{\partial D}{\partial \sigma}\right)^T_{\vec{x}_0}}$, and where H is the Hessian at $\vec{x}_0$:

$$\boxed{H(\vec{x}_0) = \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial x \partial \sigma} \\ \frac{\partial^2 D}{\partial y \partial x} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial y \partial \sigma} \\ \frac{\partial^2 D}{\partial \sigma \partial x} & \frac{\partial^2 D}{\partial \sigma \partial y} & \frac{\partial^2 D}{\partial \sigma^2} \end{bmatrix} \text{at } \vec{x}_0}$$

At the true extremum, it must be the case that $\boxed{\frac{\partial D(\vec{x})}{\partial \vec{x}} = 0}$.

Taking a derivative of both sides of the Taylor series expansion with respect to the vector variable $\vec{x}$, we get $\boxed{0 \simeq J(\vec{x}_0) + H(\vec{x}_0)\vec{x}}$. This implies that the true location of the extremum is given by $\boxed{\vec{x} = -H^{-1}(\vec{x}_0) \cdot J(x_0)}$

STEP 3:

- Weed out the weak extrema by thresholding $|D(\vec{x})|$ at the locations $(x, y, \sigma)^T = \vec{x}$ of the extremums. Typically, an extremum is rejected if $\boxed{|D(\vec{x})| < 0.03}$. We also reject those extrema in the scale space that draw their support from an edge in the image. (Edges parallel to the sampling axes are unlikely to give rise to DoG pyramid extrema on account of the logic in Step 1, but slanted edges might.) This elimination of extrema uses the same logic that you will see later for the Harris corner detector. The Harris corner detection logic is applied at the same scale at which the extremum was discovered.

STEP 4:

- We now associate a 'dominant local orientation' with each extremum that survives the previous step. Note that our ultimate goal is to associate a "descriptor" with each retained extremum. To make these descriptors invariant to in-plane rotations of the recorded image, we must calculate them with respect to the dominant local orientation.
- To find the dominant local orientation, we calculate the gradient vector of the Gaussian-smoothed image $ff(x, y, \sigma)$ at the scale σ of the extremum. At each point in a K×K neighborhood around the extremum, we compute the gradient magnitude $\boxed{m(x,y) = \sqrt{|ff(x+1, y, \sigma) - ff(x, y, \sigma)|^2 + |ff(x, y+1, \sigma) - ff(x, y, \sigma)|^2}}$ and the gradient orientation $\boxed{\theta(x,y) = \arctan \frac{ff(x, y+1, \sigma) - ff(x, y, \sigma)}{ff(x+1, y, \sigma) - ff(x, y, \sigma)}}$

- After weighting $\theta(x,y)$ with $m(x,y)$, we construct a histogram of $\theta(x,y)$ values using 36 bins spanning the full 360° range. The bin where the histogram peak occurs gives us the dominant local orientation. For better accuracy, we can fit a parabola to the peak and its immediate neighbors.

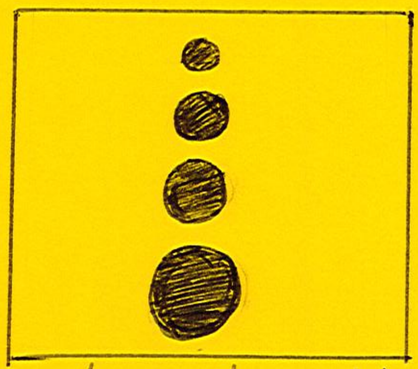
Step 5:

- This is where we create the SIFT descriptor — a 128-dimensional vector — at each retained extremum in the DoG pyramid. We use the same gradient information as was calculated in the previous step, except that the gradient directions $\theta(x,y)$ are now measured relative to the dominant local orientation.
- At the scale of the extremum, the extremum point is surrounded by a 16x16 neighborhood of points that is divided into 4x4 cells, each cell consisting of a 4x4 block of points. The magnitudes of the gradients in the 16x16 neighborhood are weighted by a Gaussian whose σ is half the width of the neighborhood in order to reduce the importance of the points that are relatively far from the extremum.
- For each of the 16 cells, an 8-bin orientation histogram is calculated from the gradient-magnitude-weighted values of $\theta(x,y)$ at the 16 pixels in the cell.
- Stringing together the 16 8-bin histograms yields a 128-element descriptor at each retained extremum in the DoG pyramid.
- Considering the 128-element descriptor as a vector in a 128-dimensional space, its length is normalized to unity to make it invariant to changes in illumination

SURF for Extracting Interest Points and Their Descriptors

- The goal is again to locate high-variance interest points in an image and to represent each such point with a vector attribute of local gray level variations. As before, we want both the interest points and their vector attributes to exhibit significant invariance to changes in scale, viewpoint, and illumination. (For both SIFT and SURF, the algorithms are designed explicitly for scale, in-plane rotation, and illumination invariances. The viewpoint invariance is a carry-over benefit of the other invariances.)

- Whereas scale invariance generally involves comparing two different images recorded at two different distances from a scene, the notion applies to a single image also. The image at right contains four circular objects at four different scales. A scale-space operator like SIFT or SURF would automatically represent this image by four interest points, one each at the center of each of the four circular blobs. Each interest point would be automatically extracted at a different scale in the scale space.



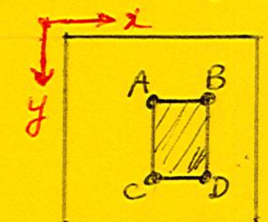
- Whereas SIFT places the interest points at the extrema of the values of $\boxed{\nabla^2 ff(x,y,\sigma) = \frac{\partial^2}{\partial x^2}ff(x,y,\sigma) + \frac{\partial^2}{\partial y^2}ff(x,y,\sigma)}$. SURF places the interest points at locations where the determinant of the Hessian

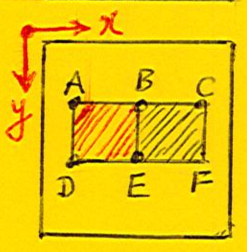
$$\boxed{H(x,y,\sigma) = \begin{bmatrix} \frac{\partial^2}{\partial x^2}ff(x,y,\sigma) & \frac{\partial^2}{\partial x \partial y}ff(x,y,\sigma) \\ \frac{\partial^2}{\partial x \partial y}ff(x,y,\sigma) & \frac{\partial^2}{\partial y^2}ff(x,y,\sigma) \end{bmatrix}}$$

is maximized. Think of $ff(x,y)$ as a surface over the xy-plane. Our goal is to find those (x,y) points where the curvature of this surface is high in ALL directions. At each point (x,y), the determinant of the Hessian measures the Gaussian curvature of the surface above that point. Since Gaussian Curvature is a product of the two principal curvatures, when Gaussian curvature is high, that means the surface has a strong variation in every direction. [SIFT tries to do the same thing using $\nabla^2 ff(x,y)$, but that is theoretically flawed.] That $\nabla^2 ff$ is the trace of the Hessian is beside the point.]

- What's surely the most significant difference between SIFT and SURF relates to how the derivatives required for $H(x,y,\sigma)$ are computed in the latter. SURF uses the notion of integral images for very fast calculation of these derivatives.

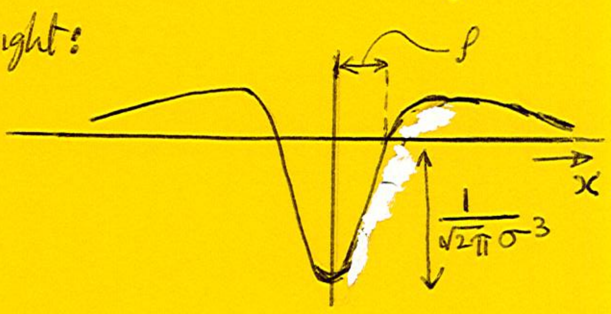
- The integral image of a given image $f(x,y)$ is obtained by summing all the pixels to the left of and above a given pixel. If $I(x,y)$ denotes the Integral Image of $f(x,y)$, we write $\boxed{I(x,y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} f(i,j)}$. With $I(x,y)$, the sum of $f(x,y)$ over any rectangular block of pixels in the image plane can be carried out with just 4 memory accesses and 3 additions. That is, to compute the sum of gray levels in the rectangular area ABCD of the image plane, all we need do is to calculate $I(D) - I(B) - I(C) + I(A)$. Let's extend this argument to the calculation of a discrete approximation to a derivative that involves convolving the image $f(x,y)$ with the operator ABCDEF shown at right in which the pixel values in the ABDE rectangle are -1 and those in the BCEF rectangle are $+1$. At any position of this operator over the image $f(x,y)$, and, even more importantly, for any size of the operator, the derivative can be calculated with just six additions of the $I(x,y)$ values.



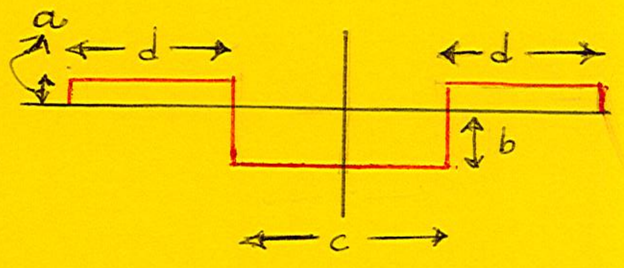


- Next we'll talk about the discretization of the derivatives in $H(x,y,\sigma)$

- Focusing on the discretization of $\boxed{\frac{\partial^2}{\partial x^2} f(x,y)}$ in the Hessian, this derivative amounts to convolving the image $f(x,y)$ with $\frac{\partial^2 g}{\partial x^2}$ where $\boxed{g(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-x^2/2\sigma^2}}$. As pointed out in the second bullet of page 9-2 of this scroll, this implies that we need to convolve the image $f(x,y)$ with $\boxed{h(x) = -\frac{1}{\sqrt{2\pi}\,\sigma^3}\left[1 - \frac{x^2}{\sigma^2}\right] e^{-x^2/2\sigma^2}}$, as shown at right:



To convert this operator into a discrete version that would allow us to take advantage of the integral image $I(x,y)$ associated with $f(x,y)$, we require that the operator look like what is shown in red at right. The parameters a, b, c, and d of the discrete version are determined from the following stipulations (assuming that the x-axis is discretized to represent the pixels with a sampling interval of unity): (1) The smallest value a must equal 1; (2) the overall width of the operator, c+2d, must be an odd integer to make it symmetric with respect to the image pixel on which it is centered during convolution; (3) the width c of the central 'lobe' be as close as possible to 2ρ subject to other conditions being satisfied; and, finally, (4) b be set to an integer so that all the operator elements add up to zero. For σ = 1.2, which is the lowest scale at which the Hessian is calculated by SURF, we end up with a = 1, b = −2, c = d = 3, which is a 9-element 1-D operator. From this we construct a 2D 9x9 operator that includes some smoothing by averaging the results over 5 rows of the image, as shown at right.



$\frac{\partial^2}{\partial x^2}$ for σ = 1.2

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

- Similar reasoning applied to the discretization of the $\frac{\partial^2}{\partial x \partial y}$ derivative in the Hessian gives us for σ = 1.2 the 9x9 operator shown at right here. As for the $\frac{\partial^2}{\partial y^2}$ derivative in the Hessian, it is simply a 90° rotated version of the discretized version of $\frac{\partial^2}{\partial x^2}$.

$\frac{\partial^2}{\partial x \partial y}$ for σ = 1.2

0	0	0	0	0	0	0	0	0
0	-1	-1	-1	0	1	1	1	0
0	-1	-1	-1	0	1	1	1	0
0	-1	-1	-1	0	1	1	1	0
0	0	0	0	0	0	0	0	0
0	1	1	1	0	-1	-1	-1	0
0	1	1	1	0	-1	-1	-1	0
0	1	1	1	0	-1	-1	-1	0
0	0	0	0	0	0	0	0	0

- For another important difference between SIFT and SURF, there is no need to downsample the images in SURF as you go up the scale-space pyramid. SIFT required us to downsample the images because we wanted to use the same operator for Gaussian smoothing at different scales. In SURF, the use of integral images and the "box filter" for the discretized operators make the cost of convolution independent of the size of the operator. So, in a SURF pyramid, the image stays at the same resolution at all scales.

- So, whereas the image resolution remains the same along the scale dimension in a SURF pyramid, the operators for calculating the derivatives in the Hessian must become larger with scale. To see the issues involved here, let's consider the next larger operator after the 9x9 operator shown previously. For the 9-element operator for $\frac{\partial^2}{\partial x^2}$ for $\sigma = 1.2$, the central lobe was 3 elements wide. In order to maintain symmetry around the central pixel, the central lobe in the next larger operator must be 5 pixels wide. However, that implies that the widths of the two side-lobes must each be increased by 2. This results in a 15-element operator for estimating $\frac{\partial^2}{\partial x^2}$, implying a 15x15 2D operator. The scale change in going from a 9x9 operator to a 15x15 operator is $\frac{15}{9} \simeq 1.7$, which is too large a jump along σ when you are looking for extrema in the determinant of the Hessian. You can get around this problem either by interpolation in the scale space, or by starting with an upsampled version of the image (upsampled by, say, a factor of 2).

- So far we have focused on the calculation of the second-order derivatives required for the Hessian. The interest points are located at the pixels where the determinant is a maximum with respect x, y, and σ. The rest of the SURF algorithm requires the calculation of first-oder derivatives $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ at different scales. As with SIFT, we need to associate a local dominant direction with each interest point and then compute a descriptor with respect to the dominant direction. The first-order derivatives for both these are calculated with the "Haar wavelet filters" that allow us to take advantage of the Integral Image $I(x,y)$ in the same manner as explained earlier. The basic form of the Haar wavelet is $\begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array}$ along x and $\begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline \end{array}$ along y. These forms are scaled up to an MxM operator where M is the smallest even integer greater than 4xσ. Shown at right are the 6x6 operators for σ = 1.2. Note that σ in the size of the operators refers to the scale at which the interest point was detected. We use $\boxed{d_x}$ to denote the output of the $\frac{\partial}{\partial x}$ operator and $\boxed{d_y}$ to denote the output for ∂/∂y.

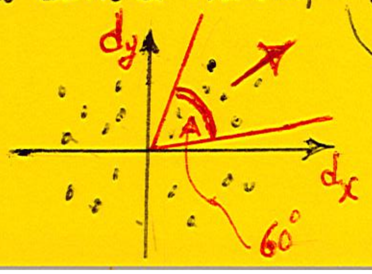
$\frac{\partial}{\partial x}$ for $\sigma = 1.2$

-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1

$\frac{\partial}{\partial y}$ for $\sigma = 1.2$

1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1

- At an interest point detected at scale σ, the local dominant direction is found by examining the d_x and d_y values in a 6σ X 6σ neighborhood around the

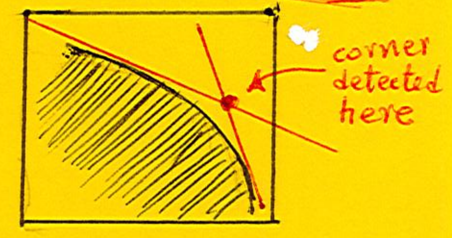


interest point. The (d_x, d_y) values are weighted with a 2σ-Gaussian centered at the interest point to decrease the importance of the points relatively far from the interest point. The calculation of The dominant direction amounts to scanning a scatter plot of the weighted (d_x, d_y) values with a 60° cone, as shown in the figure, to find the cone that yields the largest resultant vector when all the weighted (d_x, d_y) values are added vectorially in the cone. The direction of the resultant vector is the local dominant direction.

- The descriptor at an interest point detected at scale σ is calculated from a $20\sigma \times 20\sigma$ neighborhood around the point. This neighborhood is oriented along the local dominant direction found in the previous step; that is, one side of the square neighborhood is parallel to the dominant direction, and the other side perpendicular. The $20\sigma \times 20\sigma$ neighborhood is divided into a pattern of 4x4 squares, and each of the 16 squares thus created sampled over an array of $5\sigma \times 5\sigma$ points. Let d_x' and d_y' be the outputs of the two Haar operators relative to the local dominant direction.
- For each of the 16 squares mentioned above, we construct a 4-vector $\boxed{(\Sigma d_x', \Sigma d_y', \Sigma |d_x'|, \Sigma |d_y'|)}$ where the summations are over the 5x5 array of points in the square. Summing the actual d_x' and d_y' values and just their magnitudes separately helps us retain information regarding the polarity of the first-order gray level variations in each square separately. Stringing together the 4-vectors for the 16 squares gives us a 64-element descriptor vector at each interest point. We normalize the length of the vector to unity to make it invariant to contrast changes.

The Harris Corner Detector

- Before SIFT and SURF came along, this was probably the world's most popular interest point detector. It serves a useful purpose even today if your application does not require scale-space analysis or if the value of scale is fixed and known in advance.
- By a corner we mean any pixel in the vicinity of which the gray levels show significant variations in at least two different directions.



- The secret of a good corner detector is that its characterization of a pixel as a corner should be invariant to in-plane rotations of the image as long as there does not exist a direction with no gray level variations inside the window to which the operator is applied. The Harris corner detector possesses this invariance.
- Let d_x and d_y be the outputs of the x- and y-oriented Haar filters at scale σ as in our presentation of SURF. So d_x estimates the x-derivative and d_y the y-derivative at a pixel at scale σ. If scale is not important, you can also use for d_x and d_y the x- and the y-components of the local gradient as measured by the Sobel operator.
- We now construct the following matrix in a $5\sigma \times 5\sigma$ neighborhood of the pixel where we wish to determine the presence or the absence of a corner:

$$\boxed{C = \begin{bmatrix} \sum d_x^2 & \sum d_x d_y \\ \sum d_x d_y & \sum d_y^2 \end{bmatrix}}$$

where the summations are over all the pixels in the $5\sigma \times 5\sigma$ neighborhood.

This 2x2 matrix has full rank at a genuine corner. However, if the pixel on which the $5\sigma \times 5\sigma$ window is centered is on a straight edge (with all other pixels possessing uniform gray levels), its rank will reduce to 1. [Just imagine when the edge is parallel to the y-axis, in this case all d_y's will be zero. For an edge at an arbitrary orientation, d_y will be a constant times d_x.]

- Let λ_1 and λ_2 be the two eigenvalues of C. We assume $\lambda_1 \geqslant \lambda_2$. We place a threshold on the ratio $r = \lambda_2/\lambda_1$ and typically require $r \gtrsim 0.1$ for the pixel at the center of the window to be called a corner.
- The ratio r can be computed very efficiently without an explicit eigendecomposition of C by recognizing $\boxed{Tr(C) = \sum d_x^2 + \sum d_y^2 = \lambda_1 + \lambda_2}$ and $\boxed{det(C) = \sum d_x^2 \sum d_y^2 - (\sum d_x d_y)^2 = \lambda_1 \lambda_2}$. This implies $\boxed{\frac{det(C)}{[Tr(C)]^2} = \frac{\lambda_1 \lambda_2}{(\lambda_1+\lambda_2)^2} = \frac{r}{(1+r)^2}}$. Therefore, the ratio $\frac{det(C)}{[Tr(C)]^2}$ can be directly thresholded for corner detection.

Establishing Correspondences Between Image Pairs

- When scale-space operators like SIFT and SURF are used to extract interest points from the two images of the same scene (recorded from two different viewpoints), the interest points in the two images can be compared directly by measuring the Euclidean distance between their descriptor vectors
- On the other hand, when the low-level features extracted from the two images are corners, we can establish correspondences by directly comparing the gray levels in, say, a $(M+1) \times (M+1)$ window around the corner pixel in one image with the gray levels in a similar window around the corresponding pixel in the other image. We can use the following two metrics for such comparisons:

f_1: Image 1
f_2: Image 2
m_i: mean of f_i in

SSD: Sum of Squared Differences

$$\boxed{SSD = \sum_i \sum_j |f_1(i,j) - f_2(i,j)|^2}$$

Normalized Cross Correlation

$$\boxed{NCC = \frac{\sum\sum (f_1(i,j)-m_1)(f_2(i,j)-m_2)}{\sqrt{[\sum\sum(f_1(i,j)-m_1)^2][\sum\sum(f_2(i,j)-m_2)^2}}}$$