Computer Vision

James L. Crowley

M2R MoSIG option GVR — Fall Semester
13 October 2010

Lesson 3

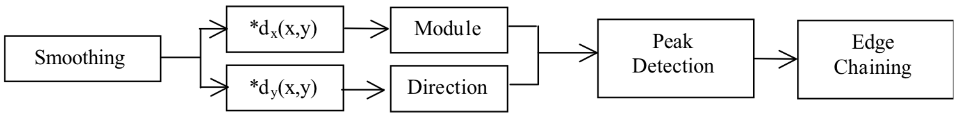
Describing Local Appearance

Lesson Outline:

1 Describing Contrast with Edges 2
1.1 Edge Detection using integer coefficient filters....2
1.2 Non-maximum suppression. 3
2 The Hough Transform....4
2.1 The Algorithm 4
2.2 Generalisation of the Hough Transform....5
3 Second Derivatives. 6
3.1 Integer Coefficient Second Derivatives....7
3.2 Zero Crossings in the second derivative....8
4 Beyond Edges - Describing Local Appearance 9
5 Image Description Using Gaussian Derivatives 10
5.1 Gaussian Derivatives Operators....10
5.2 The Sampled Gaussian Functions 10
5.3 Gaussian Digital Filters 11
5.4 Properties of Gaussian Receptive Fields (or Wavelets)....13
5.5 Using the Gaussian to compute image derivatives 16
5.6 Steerability of Gaussian Derivatives. 16

1 Describing Contrast with Edges

1.1 Edge Detection using integer coefficient filters



$$\nabla P(i,j) = \begin{bmatrix} \frac{\Delta p(i,j)}{\Delta i} \\ \frac{\Delta p(i,j)}{\Delta j} \end{bmatrix} = \begin{bmatrix} m_{1*} p(i,j) \\ m_{2*} p(i,j) \end{bmatrix} = \begin{bmatrix} E_1(i,j) \\ E_2(i,j) \end{bmatrix}$$

Gradient:

$$E(i,j) = \left\| \vec{E}(i,j) \right\| = \sqrt{E_1(i,j)^2 + E_2(i,j)^2}$$

Direction of maximum contrast

$$\varphi(i,j) = Tan^{-1}\left(\frac{E_2(i,j)}{E_1(i,j)} \right)$$

Steps:
1) Smoothing - Suppress high frequency noise
2) Gradient - Compute first derivatives in row and column
3) Detection - Non-maximum suppression with double threshold
4) Chaining - Assembly of connected points above threshold. Elimination of chains where one of the points is not above a second threshold.
5) Polygonal approximation (multiple algorithms exist).

1.2 Non-maximum suppression.

Contrast points are local maxima in E(i, j).

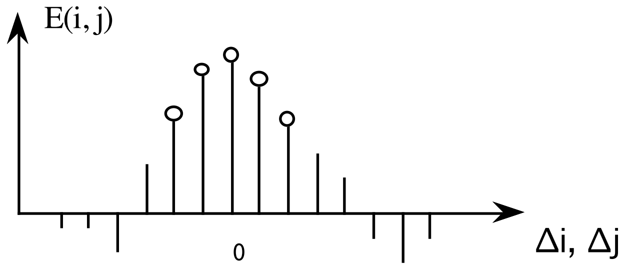
les points de contraste : C(i, j)
pour le gradient de la magnitude E(i, j) et orientation Φ(i, j)

For each point :

1) Determine the direction of maximum gradient:

$$\Delta i = \frac{\Delta_i P(i,j)}{\|\nabla P(i,j)\|} \qquad \Delta j = \frac{\Delta j P(i,j)}{\|\nabla P(i,j)\|}$$

2) Compare the gradient to its neighbors in this direction.



$$c(i,j) = \begin{cases} E(\mathrm{i,j}) \ \text{ if } \mathrm{E(i-\Delta i, j-\Delta j)} \le \mathrm{E(i,j)} \ge \mathrm{E(i+\Delta i, j+\Delta j)} \\ 0 \ \text{ Otherwise} \end{cases}$$

Construct a list of connected points for which E(i,j) $\neq$ 0.

Techniques:

1) Line scan edge chaining algorithm
2) Edge following
3) Hough Transform

2 The Hough Transform

The Hough transform is an "optimal" statistical detector for estimating parametric functions from discrete samples. This method was invented for interpreting bubble chamber images in particle physics. It is based on "voting" for possible parameters.

This transform was invented by
P.V.C. Hough, Machine Analysis of Bubble Chamber Pictures, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

It was patented in a crude form by IBM in 1962 using $y = mx+c$.

It was made popular by Duda and Hart :
Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, Vol. 15, pp. 11–15 (January, 1972)

2.1 The Algorithm

Consider the line equation

$$x \cos(\theta) + y \sin(\theta) + c = 0$$

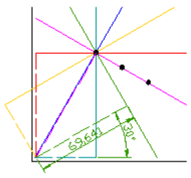
In the image, for each x,y (free parameters) we need to determine (c, θ)

In the Hough transform, we will create a dual space in which (c, θ) are free parameters.
We will estimate lines as peaks in this dual space. To find peaks we build an accumulator array : $h(c, \theta)$.

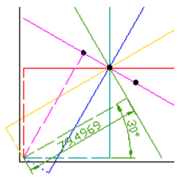
Let the c be an integer $c \in [0, D]$ where D is the "diagonal distance of the image.
Let θ be an integer $\theta \in [0, 179]$

Algorithm:
```
allocate a table  $h(c, \theta)$ initially set to 0.
For each x, y of the image
    for  $\theta$ from 0 to 179
        $c = -x \cos(\theta) - y\sin(\theta)$
        $h(c, \theta) = h(c, \theta) + E(x, y)$
    End
End
```

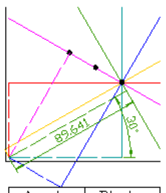
The resulting table accumulates contrast.
Peaks in $h(c, \theta)$ correspond to line segments in the image.



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

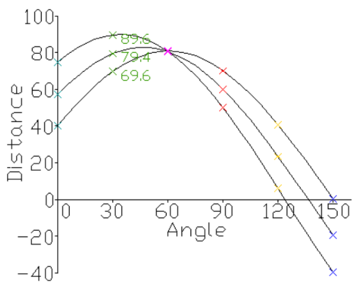


Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5



Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

=>



Because we know θ(x, y), we can limit the evaluation to θ(x, y)+/- Δθ

2.2 Generalisation of the Hough Transform

We can represent a circle with the equation:

$$(x - a)^2 + (y - b)^2 = r^2$$

We can use this to create a Hough space h(a, b, r) for limited ranges of r.

The ranges of a and b are the possible positions of circles.

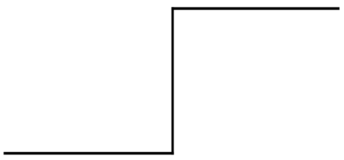
Algorithm:

```
allocate a table  h((a, b, r) initially set to 0.
For each x, y of the image
    for r from $r_{min}$ to $r_{max}$
        for a from 0 to $a_{max}$
            $b = -y - \mathrm{sqrt}( r^2 - (x - a)^2)$
            h(a,b,r) = h(a,b,r) + E(x,y).
        End
    End
End
```

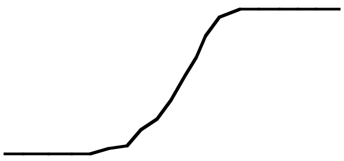
3 Second Derivatives.

An alternative to the gradient is to detect edges as zero crossings in the second derivative.

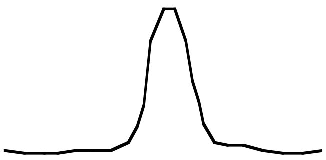
Contrast :



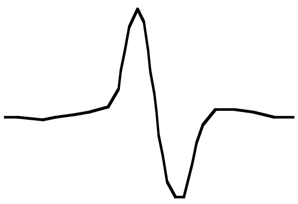
Smoothing :



1st derivative



Second derivative



The zero crossing of the second derivative gives a closed contour corresponding to edges. Unfortunately, many noise contours are commonly observed.

3.1 Integer Coefficient Second Derivatives

The second derivative is a form of Laplacian operator:

$$\text{Laplacien: } \nabla^2 p(i,j) = \frac{\Delta^2 P(i,j)}{\Delta i^2} + \frac{\Delta^2 P(i,j)}{\Delta j^2}$$

$$\Delta_i^2 = [\,1\ \ -1\,] * [\,1\ \ -1\,] = [\,-1\ \ 2\ \ -1\,]$$

$$\Delta_i^2 = \boxed{-1\ 2\ -1} \qquad \Delta_j^2 = \boxed{\begin{matrix} -1 \\ 2 \\ -1 \end{matrix}}$$

$$\text{Laplacian : } \nabla^2 p(i,j) = \Delta_i^2 * p(i,j) + \Delta_j^2 * p(i,j)$$

There are several possible discrete forms:

$$\boxed{\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}} = \boxed{\begin{matrix} -1 \\ 2 \\ -1 \end{matrix}} + \boxed{-1\ 2\ -1}$$

L(u,v) = 4 - 2cos(u) - 2cos(v)
The best is :

$$\boxed{\begin{matrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{matrix}} + 2\,\boxed{\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}} = \boxed{\begin{matrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{matrix}}$$

$$\text{Gradient : } \|\nabla p(i,j)\| = \sqrt{\left(\frac{\partial P(i,j)}{\partial i}\right)^2 + \left(\frac{\partial P(i,j)}{\partial j}\right)^2}$$

$$\text{Laplacien : } \nabla^2 p(i,j) = \frac{\partial^2 P(i,j)}{\partial i^2} + \frac{\partial^2 P(i,j)}{\partial j^2}$$

3.2 Zero Crossings in the second derivative.

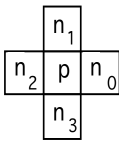
In theory

1) Zero crossings give closed contours
2) Zero crossings can be easily interpolated for high precision.

In practice

Zero crossings detect many small unstable contours.

Neighborhood test:



$$C(i, j) = \begin{cases} 1 & if(sign(n_0) \neq sign(n_2))\ and\ |n_0 n_2 | > 0 | \\ & or \text{ if } (signe(n_1) \neq signe(n_3))\ and\ |n_1 n_3 | > 0 \\ 0 & \text{Otherwise} \end{cases}$$

alternatively:

$$C(i, j) = \begin{cases} 1 & \text{if } (\text{sign}(n_0) \neq \text{sign}(n_2)) \\ & \quad \text{and } |n_0\ n_2| > 0 | \\ & \quad \text{and } |n_0 - n_2| > \text{Threshold} \\ & \text{or if } (\text{sign}(n_1) \neq \text{sign}(n_3)) \\ & \quad \text{and } |n_1 n_3 | > 0 \\ & \quad \text{and } |n_1 - n_3 | > \text{Threshold} \\ 0 & \text{Otherwise} \end{cases}$$

4 Beyond Edges - Describing Local Appearance

Appearance is the visual manifestation of a phenomena.

We seek a set of K local basis functions, $d_k(x,y)$ to describe "appearance" around a point in an image. Each basis function, $d_k(x,y)$ gives an image "feature", x_k, describing appearance at the pixel (x_o, y_o).

$$x_k = \sum_{x=-R}^{R} \sum_{y=-R}^{R} p(x_o + x, y_o + y) d_k(x,y)$$

Projection of the image neighborhood $P(x_o,y_o)$ onto this set of functions gives a "feature" vector for appearance, $\vec{X} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_K \end{pmatrix}$

Ideally these functions should be orthogonal

$$\sum_{x=-R}^{R} \sum_{y=-R}^{R} d_m(x_o + x, y_o + y) d_n(x,y) = 0 \quad \text{if n} \neq \text{m}$$

to minimize the number of features. This is not strictly necessary.
Such a basis can be obtained by a Taylor series representation of the image function P(x) around a point, a:

$$f(x) = f(a) + \frac{1}{1!} f_x (x-a) + \frac{1}{2!} f_{xx}(x-a)^2 + \frac{1}{3!} f_{xxx}(x-a)^3 + \dots$$

The basis set for a Taylor series is the series of local derivatives.
This is called the "local jet". The local jet is a basis for describing appearance.

Because the image is a sampled signal, we need a kernel filter to compute the local Jet. The Gaussian function provides a Kernel with many useful properties.

5 Image Description Using Gaussian Derivatives

5.1 Gaussian Derivatives Operators

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

The Gaussian function is invariant to affine transformations.

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

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:

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

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

5.2 The Sampled Gaussian Functions

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

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

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

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

The sampled Gaussian and its derivatives are:

$$G(n,\sigma) = e^{-\frac{n^2}{2\sigma^2}}$$

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

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

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

For the 2D Sampled Gaussian $G(i,j,\sigma) = e^{-\frac{i^2+j^2}{2\sigma^2}}$

(Attention: i, j are INTEGERS, not complex numbers!)

2D Gaussian is separable: $G(i,j,\sigma) = \frac{1}{A} e^{-\frac{(j^2+j^2)}{2\sigma^2}} = e^{-\frac{i^2}{2\sigma^2}} * e^{-\frac{j^2}{2\sigma^2}}$

The normalization factor $A = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} e^{-\frac{(i^2+j^2)}{2\sigma^2}} \approx 2\pi\sigma$

This factor will be slightly smaller if the Gaussian is evaluated over a finite window $W_N(i,j)$ of size NxN pixels.

This factor can be determined from the filter, and can be ignored in discussing the properties of the Gaussian.

5.3 Gaussian Digital Filters

Computers represent image as 2D sampled digitized signals. Because they are sampled, processing requires convolution with a sampled filter.

To obtain a 2D digital Gaussian filter we must sample the function at a rate of Δx, and Δy over some fine support of radius R.

G(x, σ)

Because we can control scale with σ, we can set the sample size to Δx=Δy=1.
This gives a sampled function

$$G(n,\sigma) = e^{-\frac{n^2}{2\sigma^2}}$$

To represent this in a computer we must also specify the finite support (number of samples), N.

We set N = 2R + 1.

This gives us 2 parameters to control:

1) The scale of the Gaussian $\sigma/\Delta x$
2) the size of the support $N = 2R+1$

Truncating a function to a finite support is equivalent to multiply by a window $W_N(n)$

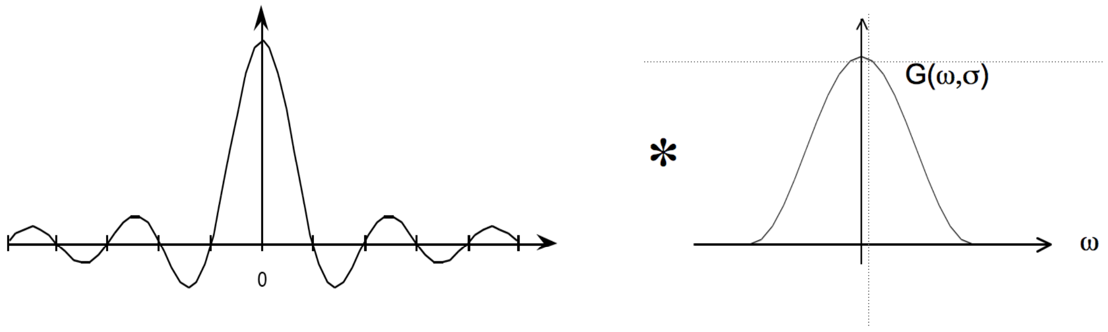
When we limit $G(x,\sigma)$ to a finite support, we multiply by a window

$$G(n, \sigma) = G(n, \sigma) \cdot w_N(n) \text{ where } w_N(n) = \begin{cases} 1 & \text{for } -R \le n \le R \\ 0 & \text{otherwise} \end{cases}$$

Multiplying by a finite window is equivalent to convolving with the Fourier transform of the finite window:

$$F\{G(n,\sigma)\cdot w_N(n)\} = G(\omega,\sigma) * W_N(\omega)$$

where $W_N(\omega) = \dfrac{\sin(\omega N/2)}{\sin(\omega/2)}$ and $G(\omega,\sigma) = \sqrt{2\pi}\,\sigma e^{-\frac{1}{2}\sigma^2\omega^2}$



For $N < 7$, the ripples in WN(w) dominate the spectrum and corrupt the resulting Gaussian.
At N=7 the effect is tolerable but significant.
At $N \geq 9$ the effect becomes minimal

In addition for $\sigma/\Delta x < 1$, the phenomenon of aliasing folds a significant amount of energy at the nyquist frequency, corrupting the quality (and the invariance) of the Gaussian function.

Finally, it is necessary to assure that the "gain" of the Gaussian filter is 1. This can be assured by normalizing so that the sum of the coefficients is 1. If the Gaussian were infinite in extent, then

$$\sum_{x=-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} = \sqrt{2\pi}\sigma$$

However, because we truncate the Gaussian to an extent n ±R, we must calculate the sum of the coefficients, A:

$$A = \sum_{n=-R}^{R} e^{-\frac{n^2}{2\sigma^2}}$$

The Gaussian filter is thus normalized by dividing by A to give a unit gain Receptive Field.

$$G(i,j,\sigma) = \frac{1}{A} e^{-\frac{(j^2+j^2)}{2\sigma^2}}$$

5.4 Properties of Gaussian Receptive Fields (or Wavelets)

Separability:

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

Sampled Gaussian: $W_N(i,j)\cdot G(i,j,\sigma) = W_N(i,j)\cdot e^{-\frac{(j^2+j^2)}{2\sigma^2}} = (W_N(i)\cdot e^{-\frac{i^2}{2\sigma^2}}) * (W_N(j)\cdot e^{-\frac{j^2}{2\sigma^2}})$

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

Derivative Filters:

$$G_x(i,j,\sigma) = -\frac{i}{\sigma^2} G(i,j,\sigma)$$
$$G_{xx}(i,j,\sigma) = \frac{i-\sigma^2}{\sigma^4} G(i,j,\sigma)$$
$$G_{xy}(i,j,\sigma) = \frac{ij}{\sigma^4} G(i,j,\sigma)$$
$$G_{xxx}(n,\sigma) = -\frac{i^3 - i\sigma^2}{\sigma^6} G(i,j,\sigma)$$
$$\nabla^2 G_x(i,j,\sigma) = G_{xx}(i,j,\sigma) + G_{yy}(i,j,\sigma)$$

Diffusion Equation: $\nabla^2 G_x(i,j,\sigma) = G_{xx}(i,j,\sigma) + G_{yy}(i,j,\sigma) = \frac{\partial G(i,j,\sigma)}{\partial \sigma}$

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

This typically requires $\sigma_1 \geq 1.4\, \sigma_2$

Thus it is common to use:

$$\nabla^2 G(x,y,\sigma) \approx G(x,y,\sqrt{2}\sigma) - G(x,y,\sigma)$$

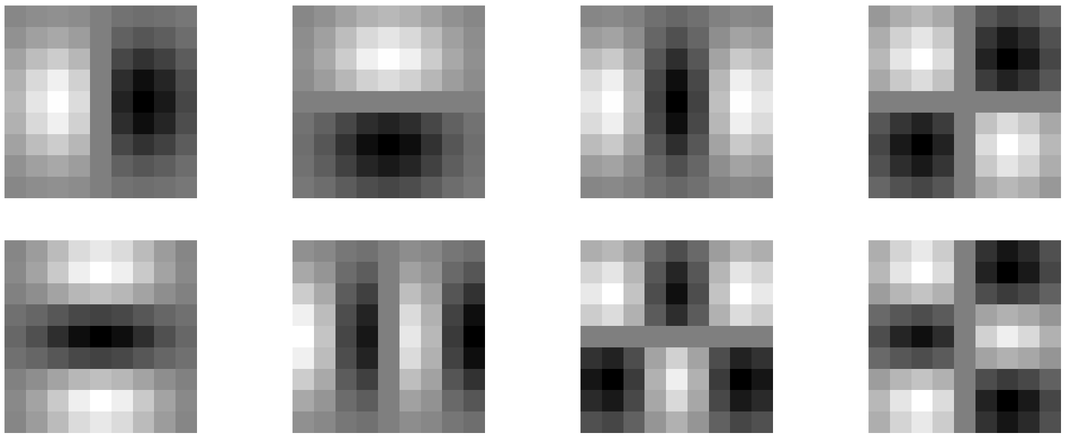
But note that :

$$G(x,y,\sqrt{2}\sigma) \approx G(x,y,\sigma) * G(x,y,\sigma)$$

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 sampled functions to create a basis set of receptive fields:

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



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

Note that there is only one parameter: σ. This determines the limit of the resolution for the position of a contrast point.

These can be used to compute a local description known as the "local jet".

$$\vec{L}(i,j,\sigma) = P * \vec{G}(i,j,\sigma)$$

The Local jet can be steered to a normalization angle θ.

Steerability:

$$G_\theta^1(x,y,\sigma) = \cos(\theta)\cdot G_x(x,y,\sigma) + \sin(\theta)\cdot G_y(x,y,\sigma)$$

$$G_\theta^2(x,y,\sigma) = Cos(\theta)^2 G_{xx}(x,y,\sigma) + Cos(\theta)Sin(\theta)G_{xy}(x,y,\sigma) + Sin(\theta)^2 G_{yy}(x,y,\sigma)$$

$$G_\theta^3(x,y,\sigma) = Cos(\theta)^3 G_{xxx}(x,y,\sigma) + Cos(\theta)^2 Sin(\theta)G_{xxy}(x,y,\sigma) + Cos(\theta)Sin(\theta)^2 G_{xxy}(x,y,\sigma) + Sin(\theta)^3 G_{yyy}(x,y,\sigma)$$

Note the scale parameter σ determines the "resolution" of the derivatives.
You MUST specify σ. The smallest σ is not always the best.
Many computer vision algorithms give unpredictable results because the researchers forget to specify the scale σ at which the algorithm was validated.

5.5 Using the Gaussian to compute image derivatives

Consider an image: P(i, j),

The Gradient $\vec{\nabla}P(i,j)$ is calculated by $P(i,j) * \vec{\nabla}G(i,j,\sigma)$

where $$\vec{\nabla}G(i,j,\sigma) = \begin{pmatrix} G_x(i,j,\sigma) \\ G_y(i,j,\sigma) \end{pmatrix}$$

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

Laplacien: $$\nabla^2 P(i,j) = P * \nabla^2 G(i,j,\sigma) = \begin{pmatrix} P * \nabla^2 G(i,j,\sigma) \\ P * \nabla^2 G(i,j,\sigma) \end{pmatrix} \approx P * \nabla^2 G(i,j,\sigma_1) - P * \nabla^2 G(i,j,\sigma_2)$$

where typically $\frac{\sigma_1}{\sigma_2} \geq \sqrt{2}$

5.6 Steerability of Gaussian Derivatives.

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

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.

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

By steering the Gaussian response to the local orientation, we obtain an "invariant" measure of local contrast:

Thus:
$$P_1(x,y;\theta,\sigma) = Cos(\theta) < G_x(x,y;\sigma) \cdot P(x,y) > + Sin(\theta) < G_y(x,y;\sigma) \cdot P(x,y) >$$

a similar measure can be obtained for the second and third derivatives:

$$G_2^\theta(x,y,\sigma) = Cos(\theta)^2 G_{xx}(x,y,\sigma) + Cos(\theta)Sin(\theta)G_{xy}(x,y,\sigma) + Sin(\theta)^2 G_{yy}(x,y,\sigma)$$
$$G_3^\theta(x,y,\sigma) = Cos(\theta)^3 G_{xxx}(x,y,\sigma) + Cos(\theta)^2 Sin(\theta)G_{xxy}(x,y,\sigma) + Cos(\theta)Sin(\theta)^2 G_{xxy}(x,y,\sigma) + Sin(\theta)^3 G_{yyy}(x,y,\sigma)$$