Computer Vision

James L. Crowley

M2R MoSIG option GVR

Fall Semester
6 November 2014

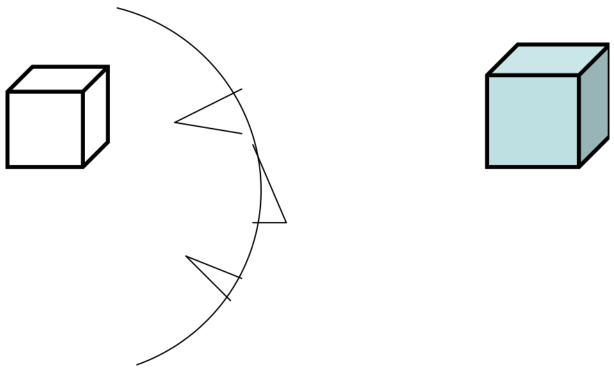
Lesson 4

Describing Images with Contrast Contours (Edges)

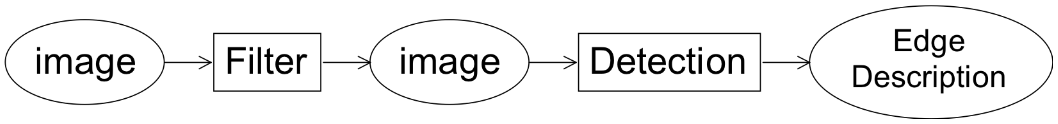
<u>Lesson Outline</u>:

1 Describing Images with Edges

Objects composed of planar surfaces (polyhedric objects) have straight line edges. Such edges are visual invariants to view angle. For this reason, edges (straight edge contours) became popular as an invariant image description during the 1960s and 1970s.



Edge detection is typically organized in two steps
1) contrast filtering
2) edge point detection, segmentation and description.



A classic contrast detection operator is the Sobel edge detector.
Modern approaches use Gaussian Derivatives.

1.1 The Sobel Edge Detector

Invented by Irwin Sobel in his 1964 Doctoral thesis, this edge detector was made famous by the classic text book of R. Duda and P. Hart published in 1972.

$$m_r(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad m_c(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$m_c(i,j)$: detects contrast in column direction. *$m_r(i,j)$* : detects contrast in row direction

Convolution (or filtering) gives an edge vector: for n = r, c (row or column)

$$E_n(i,j) = m_n * p(i,j) \sum_{k=-1}^{1}\sum_{l=-1}^{1} m_n(k,l)p(i-k,j-l) \qquad \vec{E}(i,j) = \begin{pmatrix} E_c(i,j) \\ E_r(i,j) \end{pmatrix} = \begin{pmatrix} m_c(i,j) \\ m_r(i,j) \end{pmatrix}$$

The contrast is the edge energy: $E(i,j) = \|\vec{E}(i,j)\| = \sqrt{E_r(i,j)^2 + E_c(i,j)^2}$

The direction of maximum contrast is the phase angle $\varphi(i,j) = Tan^{-1}\left(\frac{E_c(i,j)}{E_r(i,j)}\right)$

Sobel's edge filters can be seen as a composition of an image derivative and a binomial smoothing filter.

$$m_c(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = b_c(i,j) * d_r(i,j) = \sum_{k=-1}^{1}\sum_{l=-1}^{1} b_c(k,l)d_r(i-k,j-l)$$

Attn: The "*" is NOT vector multiplication!! This is convolution.

$$m_r(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} = b_r(i,j) * d_c(i,j) = \sum_{k=-1}^{1}\sum_{l=-1}^{1} b_r(k,l)d_c(i-k,j-l)$$

The filter $d_c(i,j) = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ is a form of image derivative in the column direction (along each row).

The filter $b_c(i,j) = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ is a binomial smoothing filter in the row direction (along each column).

1.2 Difference Operators: Derivatives for Sampled Signals

For continuous x, the derivative of the function s(x) can be defined as a one sided derivative or a two sided derivative:

One sided derivative: $\frac{\partial s(x)}{\partial x} = \lim_{\Delta x \to 0} \left\{ \frac{s(x+\Delta x) - s(x)}{\Delta x} \right\}$

For a sampled signal, s(n), an the equivalent is $\frac{\Delta s(n)}{\Delta n}$

A two sided (symmetric) derivative is $\frac{\partial s(x)}{\partial x} = \lim_{\Delta x \to 0} \left\{ \frac{s(x+\Delta x) - s(x-\Delta x)}{2\Delta x} \right\}$

For a sampled signal, the two sided derivative is :

$$\frac{\Delta s(n)}{\Delta n} = \frac{s(n+1) - s(n-1)}{2} = s(n) * \begin{bmatrix} -1/2 & 0 & 1/2 \end{bmatrix}$$

The ½ term is a scalar multiple that can be neglected or normalize away.
The result is the edge operator used by Sobel.

$\Delta n = 1$: $\frac{\Delta s(n)}{\Delta n} = s(n) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$

1.3 Fourier Analysis of Derivative Operators

Note that a derivative is equivalent to a convolution!

We can define derivation in the Fourier domain as follows:

$$F\left\{ \frac{\partial s(x)}{\partial x} \right\} = -j\omega \cdot F\{s(x)\}$$

and thus

$$\frac{\partial s(x)}{\partial x} = F^{-1}\{-j\omega\} * s(x)$$

If we can determine $d(x) = F^{-1}\{-j\omega\}$ then we have our derivative operator.
If we "sample" d(x) to produce d(n) we have a sampled derivative operator.

Unfortunately, $F^{-1}\{-j\omega\}$ has an infinite duration in x, and thus d(n) is an infinite series. However, the first term of d(n) is [-1 0 1].

The Fourier Transform of a discrete signal is periodic in frequency with period π.

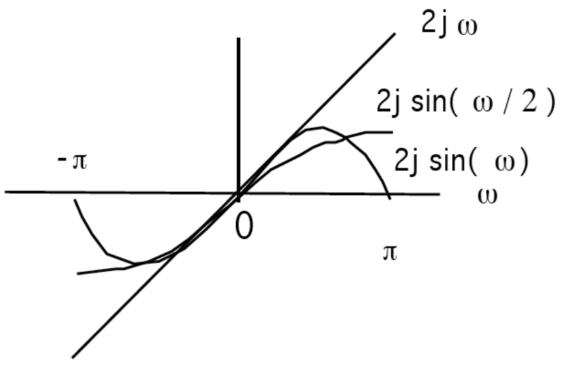
$$F\{s(n)\} = S(\omega) = \sum_{n=-\infty}^{\infty} s(n)e^{-j\omega n}$$

The first difference filter $d_1(n)$ = [1, 0, -1] has a Fourier transform:

$$\begin{aligned}
D(\omega) &= \sum_{n=-1}^{1} d(n)e^{-j\omega n} \\
D(\omega) &= 1e^{-j\omega(-1)} + 0e^{-j\omega 0} + (-1)e^{-j\omega(1)} \\
D(\omega) &= e^{j\omega} - e^{-j\omega} \\
D(\omega) &= -2j\sin(\omega)
\end{aligned}$$

Calculation of a derivative is the same as convolution with the filter [1, 0, -1], which is the same as multiplication of the spectrums.

$$d(n) * s(n) \quad \Leftrightarrow D(\omega) \cdot S(\omega)$$



The derivative used by Sobel attenuates aliasing noise in high frequencies.

1.4 Smoothing: The Binomial Low pass filter.

Sobel uses a filter b(m) = [1, 2, 1] to smooth.
It is part of a family of filters generated by the binomial series.

The binomial series is the series of coefficients of the polynomial:

$$(x+y)^n = \sum_{m=0}^{n} b_{m,n} x^{n-m} y^m$$

The coefficients can be computed as $b_{m,n} = b_n(m) = [1, 1]^n$

These are the coefficients of Pascal's Triangle.

Les coefficients du suite binomial sont générés par le triangle de Pascal :

n	sum = 2^n	μ= n/2	σ^2 = n/4	$\sigma = \sqrt{n/2}$	Coefficients
0	1	0	0	0	1
1	2	0.5	0.25		1 1
2	4	1	0/5		1 2 1
3	8	1.5	0.75		1 3 3 1
4	16	2	1	1	1 4 6 4 1
5	32	2.5	1.25		1 5 10 10 5 1
6	64	3	1.5		1 6 15 20 15 1
7	128	3.5	1.75		1 7 21 35 35 21 7 1
8	256	4	2	$\sqrt{2}$	1 8 29 56 70 56 29 8 1

These coefficients provide a family of low pass filters with remarkable properties.
Notably, these are the best approximation for a Gaussian filter of finite extent.
They also happen to have integer coefficients.

$$b_n(m) = b_1(m)^{*n} = [1, 1]^{*n} = \text{n convolutions of } [1, 1]$$

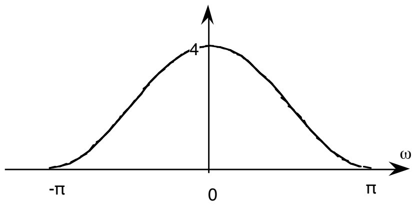
Gain : $$s_n = \sum_{m=1}^{n} b_n(m) = 2^n$$

Center of gravity is $$\mu_n = \frac{1}{s_n}\sum_{m=1}^{n} b_n(m)\cdot m = \frac{n}{2}$$

The variance is: $$\sigma_n^2 = \frac{1}{s_n}\sum_{m=1}^{n} b_n(m)\cdot(m-\mu)^2 = \frac{n}{4}$$

The Fourier transform for $b_2(m)$ = [1, 2, 1] is

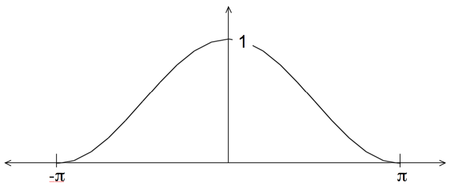
$$B_2(\omega) = \sum_{m=-1}^{1} b_2(m) e^{-j\omega m}$$
$$B_2(\omega) = 1e^{-j\omega(-1)} + 2e^{-j\omega 0} + 1e^{-j\omega(1)}$$
$$B_2(\omega) = 2 + e^{j\omega} + e^{-j\omega}$$
$$B_2(\omega) = 2 + 2\cos(\omega)$$



If we normalize the gain: $b_2(m) = (1/4)[1, 2, 1]$

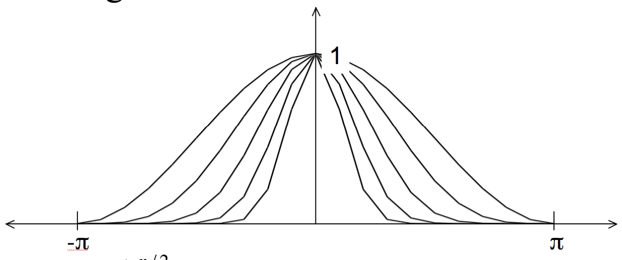
$$B_2(\omega) = \frac{1}{2} + \frac{1}{2}\cos(\omega)$$

Which is a cosine on a platform



$$B_2(\omega) = \frac{1}{2} + \frac{1}{2}\cos(\omega)$$

Repeated convolution makes generates



$$B_n(\omega) = \left(\frac{1}{2} + \frac{1}{2}\cos(\omega)\right)^{n/2}$$

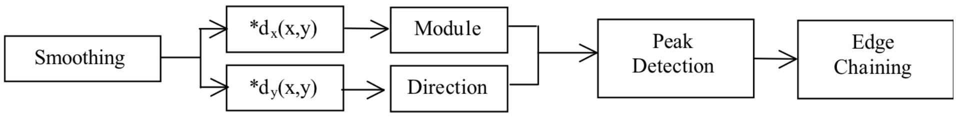
The binomial coefficients provide a series of low pass filters with no ripples.
In 2D, the filters provide separable filters that are nearly circularly symmetric

2-D $b_2(i, j) = \boxed{\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}} = \boxed{\begin{matrix} 1 \\ 2 \\ 1 \end{matrix}} * \boxed{1\ 2\ 1}$

2 Describing Contrast with Edges

Contrast is "change in image intensity".
Classically, contrast was described by detecting "edges".

2.1 Edge Detection using integer coefficient filters



Gradient of the image is a vector: $\vec{\nabla} P(i,j) = \begin{pmatrix} p_i(i,j) \\ p_j(i,j) \end{pmatrix}$

where $p_i(i,j) = \frac{\Delta p(i,j)}{\Delta i} = p(i,j) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ $\quad p_j(i,j) = \frac{\Delta p(i,j)}{\Delta j} = p(i,j) * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

Gradient Magnitude indicates edge strength $E(i,j) = \|\vec{\nabla} P(i,j)\|$

Gradient direction is the direction of maximum contrast

As an angle: $\vartheta(i,j) = Tan^{-1}\left(\frac{p_j(i,j)}{p_i(i,j)}\right)$

As a vector: $\vec{d}(i,j) = \begin{pmatrix} Sin(\vartheta(i,j)) \\ Cos(\vartheta(i,j)) \end{pmatrix} = \frac{1}{\|\nabla P(i,j)\|}\begin{pmatrix} p_i(i,j) \\ p_j(i,j) \end{pmatrix} = \begin{pmatrix} \frac{p_i(i,j)}{\|\nabla P(i,j)\|} \\ \frac{p_j(i,j)}{\|\nabla P(i,j)\|} \end{pmatrix}$

We can get the direction of direction of maximum gradient, $\vartheta(i,j)$, by normalizing the derivatives.

2.2 Non-maximum suppression.

Contrast points C(i, j) are local maxima in $\vec{\nabla}P(i,j)$ in the direction of maximum gradient. $\vartheta(i,j)$

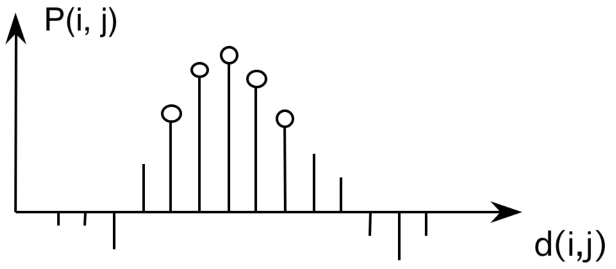
$$\vec{\nabla}P(i,j) = \begin{pmatrix} p_i(i,j) \\ p_j(i,j) \end{pmatrix} \qquad \vartheta(i,j) = Tan^{-1}\left(\frac{\Delta_j p(i,j)}{\Delta_i p(i,j)}\right) = Tan^{-1}\left(\frac{E_j(i,j)}{E_i(i,j)}\right)$$

For each edge pixel :

1) Determine the direction of maximum gradient:

$$\Delta i = \cos(\vartheta(i,j)) = \frac{P_i(i,j)}{\|\vec{\nabla}P(i,j)\|} \qquad \Delta j = \sin(\vartheta(i,j)) = \frac{P_j(i,j)}{\|\nabla P(i,j)\|}$$

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



$$c(i,j) = \begin{cases} 1 & \text{IF } E(i-\Delta i, j-\Delta j) \le E(i,j) \ge E(i+\Delta i, j+\Delta j) \\ & \text{and IF } E(i-2\Delta i, j-2\Delta j) < E(i,j) > E(i+2\Delta i, j+2\Delta j) \\ 0 & \text{Otherwise} \end{cases}$$

Alternatively, we can consider $E(i,j) = \|\vec{\nabla}P(i,j)\|$ as "evidence" for contrast at all points

3 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)

3.1 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)

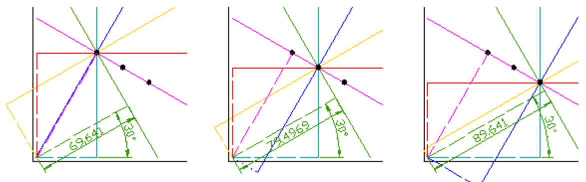
In the Hough transform, we will create a dual space in which (c, θ) are free parameters.
We will estimate lines as peaks in this dual space. To find peaks we build an accumulator array : $h(c, \theta)$.

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, θ) correspond to line segments in the image.

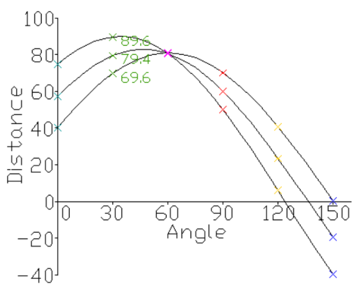


Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

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

3.2 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

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 $a_{max}$
            $b = -y - \text{sqrt}( r^2 - (x - a)^2)$
            h(a,b,r) = h(a,b,r) + E(x,y).
        End
    End
End
```