Computer Vision

James L. Crowley

M2R MoSIG option GVR

Fall Semester
20 October 2011

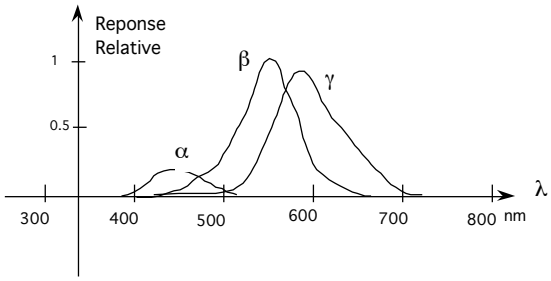
Lesson 3

Color Perception in Man and Machine

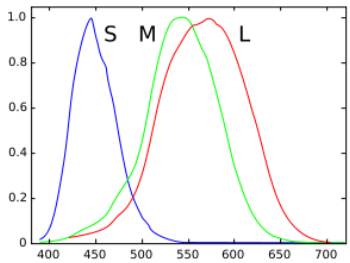
Lesson Outline:

1 Color Perception in Man and Machine

Recall from the last lecture that the human visual system uses three chromatic pigments in “cones” to perceive color.



Relative Sensitivities



Normalised Sensitivities

Cones provide our chromatique "day vision". Human Cones employ 3 pigments :

Short Wavelength: cyanolabe α 400–500 nm peak at 420–440 nm
Medium Wavelength: chlorolabe β 450–630 nm peak at 534–545 nm
Long Wavelength: erythrolabe γ 500–700 nm peak at 564–580 nm

The three pigments have different sensitivities, leading to a much stronger sensitivity for green-yellow light.

1.1 Bayer Matrix Retina

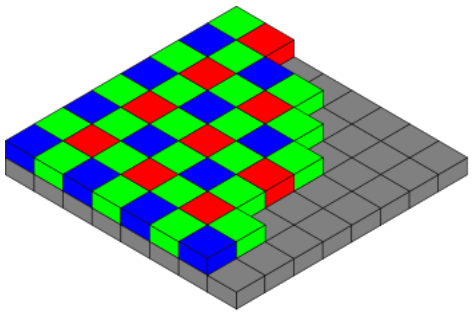
Silicon semiconductors respond to light by emitting photons (Einstein effect), thus generating a charge. A silicon retina is composed of a matrix of individual photocells cells (sensels) that convert photons to positive voltage.

Note that silicon is sensitive to light out into the near infrared (< 1500 Nm). Color filters are used to limit the spectrum of light reaching each photo-cell.

Most modern digital cameras employ a Bayer Mosaic Retina, named after its inventor, Bryce E. Bayer of Eastman Kodak who patented the design in 1976.

A Bayer filter mosaic is a color filter array (CFA) for arranging RGB color filters on a square grid of photosensors. The filter pattern is 50% green, 25% red and 25% blue, hence is also called RGBG, GRGB, or RGGB.

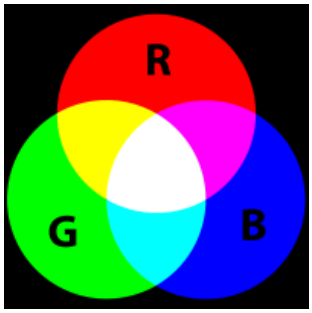
The Bayer mosaic uses twice as many green elements as red or blue to mimic the pigments of the human eye. These elements are referred to as sensor elements, sensels, pixel sensors, or simply pixels;



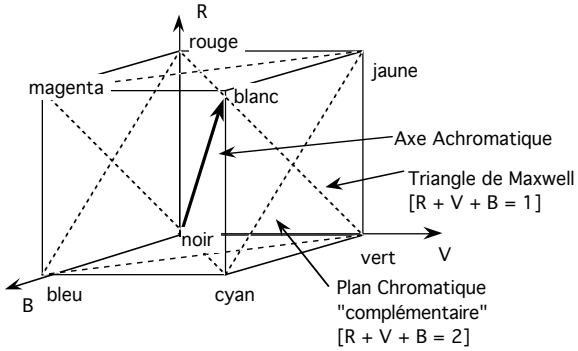
The voltage values on each sensel are converted to numeric values, interpolated and processed to provide image pixels. This step is sometimes called “image reconstruction” in the image processing community, and is generally carried out on the retina. The result is generally an image with colors coded as independent components: RGB.

1.2 The RGB Color Model

RGB is one of the oldest color models, originally proposed by Isaac Newton. This is the model used by most color cameras.



The RGB model "pretends" that Red, Green and Blue are orthogonal (independent) axes of a Cartesian space.



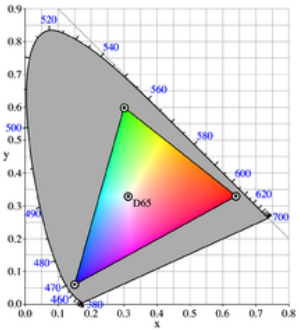
The achromatic axis is R=G=B.
Maxwell's triangle is the surface defined when R+G+B = 1.
A complementary triangle exists when R+G+B = 2.

For printers (subtractive color) this is converted to CMY (Cyan, Magenta, Yellow).

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} R_{\max} \\ G_{\max} \\ B_{\max} \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

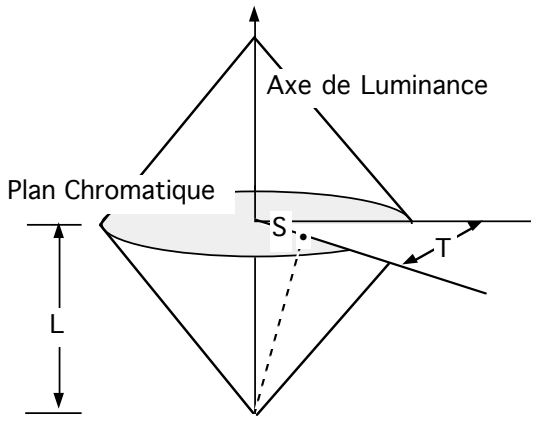
1.3 The HLS color model

The RGB model only captures a small part of visible colors:



Painters and artists generally use the HLS: Hue Luminance Saturation model.

HLS is a polar coordinate model for and hue (perceived color) and saturation.
The polar space is placed on a third axis. The size of the disc corresponds to the range of saturation values available.



One (of many possible) mappings from RGB:

Luminance : L = (R + B + B)

Saturation : 1 - 3*min(R, G, B)/L

Hue : $x = \cos^{-1}\left(\frac{\frac{1}{2}(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right)$

if B>G then H = x else H = 2π–x.

1.4 Color Opponent Model

Color Constancy: The subjective perception of color is independent of the spectrum of the ambient illumination.

Subjective color perception is provide by "Relative" color and not "absolute" measurements.

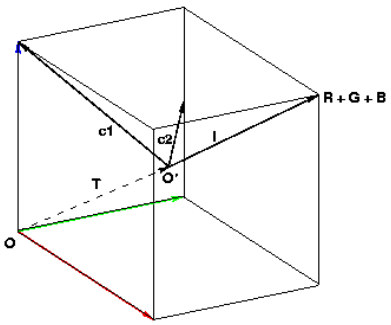
This is commonly modeled using a Color Opponent space.

The opponent color theory suggests that there are three opponent channels: red versus green, blue versus yellow, and black versus white (the latter type is achromatic and detects light-dark variation, or luminance).

This can be computed from RGB by the following transformation:
Luminance : L = R+G+B
Chrominance: C1 = (R-G)/2
C2 = B – (R+G)/2

as a matrix :

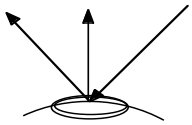
$$\begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ -0.5 & -0.5 & 1 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$



Such a vector can be "steered" to accommodate changes in ambient illumination.

1.5 Separating Specular and Lambertian Reflection.

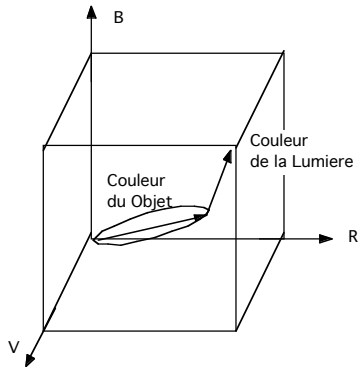
Consider what happens at a specular reflection.



The specularity has the same spectrum as the illumination.
The rest of the object has a spectrum that is the product of illumination and pigments.

This scan be seen in a histogram of color:

$$\forall \vec{C}(i,j) \quad : \quad H(\vec{C}(i,j)) = H(\vec{C}(i,j)) + 1$$



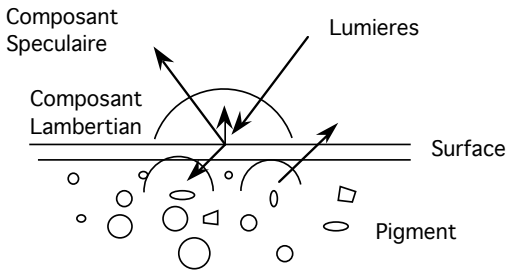
Two clear axes emerge:
One axis from the origin to the RGB of the product of the illumination and the source.
The other axis towards the RGB representing the illumination.

2 Detection and Tracking using Color

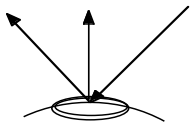
2.1 Object detection by pigment color

Recall the Bichromatic reflection function :

$$R(i, e, g, \lambda) = \alpha R_S(i, e, g, \lambda) + (1 - \alpha) R_L(i, \lambda)$$



For Lambertian reflection, the intensity || P(i,j) || is generally determined by changes in surface orientation, while color is determined by Pigment.



Thus Luminance captures surface orientation (3D shape) while
Chrominance is a signature for object pigment (identity)

Thus it is often convenient to transform the (RGB) color pixels into a color space that separates Luminance from Chrominance.

$$\begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} \Leftarrow \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Ones such space is the color opponent model:
Luminance : L = R+G+B
Chrominance: c_1 = (R-G)/2
c_2 = B – (R+G)/2

Another, popular alternative is normalized R, G.

Luminance: L= R+G+B

Chrominance : $c_1 = r = \frac{R}{R+G+B}$ $c_2 = g = \frac{G}{R+G+B}$

Suppose that these are coded with N values between 0 and N – 1

$$c_1 = \text{trunc}(N \cdot \frac{R}{R+G+B}) \qquad c_2 = \text{trun}(N \cdot \frac{G}{R+G+B})$$

Luminance normalized RG is often used for skin detection.
Skin pigment is generally always the same color. Luminance can change with pigment density, and skin surface orientation. Chrominance will remain invariant.

Thus we can use $\vec{c} = \begin{pmatrix} r \\ g \end{pmatrix}$ as a "signature for detecting skin.

2.2 Histograms

A histogram is a table of frequency of occurrence. We can use histograms to estimate probability densities for integer valued features.

Assume integer x from a bounded set of values, such that $x \in [x_{min}, x_{max}]$,
the probability that a random observation X takes on x is

$$P(X=x) = \frac{1}{M} h(x)$$

The validity of this depends on the ratio of the number of sample observations M and the number of cells in the histogram Q=N

This is true for vectors as well as values.
For a vector of D values $\vec{x}$ the table has D dimensions. $h(x_1, x_2, \ldots, x_D) = h(\vec{x})$

The average error depends on the ration $Q=N^D$ and M. : $E_{ms} \sim O(\frac{Q}{M})$

We need to assure that $M >> Q = N^d$
As a general rule : M should be greater than $8N^d$

2.3 Color Skin Detection

We can use statistics chrominance to build a very simple skin detector.

To use a Bayesian approach we need to represent the probability for each possible chrominance. We can estimate probability for chrominance with a histogram calculated from a set of training images.

Suppose the training images are composed of M pixels $\{P_m\}$.
Suppose we project these into a set of M chrominance pixels. $S = \{\vec{c}_m\}$

We then allocate a 2D table : $h(\vec{c})$ of size N x N.

For example, for skin chrominance, N=32 seems to work well.
Let $h(\vec{c})$ be a 32 x 32 table. Q = 32 x 32 = 1024 cellules

For each pixel, (i,j) possibly from S,

$$\forall_{\vec{c}_m \in S} \; h(\vec{c}_m) = h(\vec{c}_m) + 1$$

For M pixels in the training data, the histogram $h(\vec{c})$ of chrominance gives an estimate of the probability for a chrominance value within the data (or in similar data).

$$p(\vec{c}) = \frac{1}{M} h(\vec{c})$$

Important. The number of pixels, M, should be much larger than $Q = N^2$

We also can apply this to learn the probability chrominance for a target.

Suppose that we mark all pixels that belong to the target in the training data to obtain a subset $T \subset S$ composed of M_k target pixels.

We can learn a second histogram:

$$\forall_{\vec{c}_m \in T} \; h_k(\vec{c}_m) = h_k(\vec{c}_m) + 1$$

then the probability of observing a chrominance value $\vec{c}$ given the target is

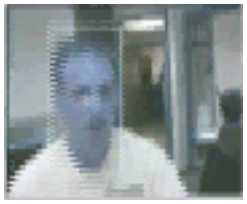
$$p(\vec{c} \mid \text{target}) = \frac{1}{M_k} h_k(\vec{c})$$

Because the target samples are a subset of the training data, the probability of a target pixel is

$$p(\text{target}) = \frac{M_k}{M}$$

From Bayes rule:

$$p(\text{target} \mid \vec{c}(i,j)) = \frac{p(\vec{c}(i,j) \mid \text{target})p(\text{target})}{p(\vec{c}(i,j))} = \frac{\frac{1}{M_k} h_k(\vec{c}(i,j)) \frac{M_k}{M}}{\frac{1}{M} h(\vec{c}(i,j))} = \frac{h_k(\vec{c}(i,j))}{h(\vec{c}(i,j))}$$



We can use this to convert a color image to a "probability image", t(i,j), by table lookup.

$$t(i,j) = \frac{h_k(\vec{c}(i,j))}{h(\vec{c}(i,j))}$$

2.4 Bayesian Tracking Process

A Bayesian Tracker is a cyclic process composed of the cycles predict, detect and update.

Tracking maintains object constancy across images. It also
1) Focusing computing resources
2) Improves reliability of detection.
3) Makes it possible to estimate motion derivatives.

2.5 Gaussian Blob Tracking

To construct a Bayesian tracker, we need to represent clouds of high target probability using Gaussian Blobs.

Gaussian blobs express a region in terms of moments.
Confidence is the sum (mass) of the detection probability pixels, t(i,j).
Position is center of gravity.
Size is the second moment (covariance).

We use some form of "a priori" estimation to estimate a Region of Interest (ROI) for the blob. Let us represent the ROI as a rectangle : (t,l,b,r)

t - "top" - first row of the ROI.
l - "left" - first column of the ROI.
b - "bottom" - last row of the ROI
r - "right" -last column of the ROI.

(t,l,b,r) can be seen as a bounding box, expressed by opposite corners (l,t), (r,b)

2.6 Moment Calculations for Blobs

Given a target probability image t(i,j) and a ROI (t,l,b,r):

Sum: $$S = \sum_{i=l}^{r} \sum_{j=t}^{b} t(i,j)$$

We can estimate the "confidence" as the average detection probability:

Confidence: $$CF = \frac{S}{(b-t)(r-l)}$$

First moments: $$\mu_i = \frac{1}{S} \sum_{i=l}^{r} \sum_{j=t}^{b} t(i,j) \cdot i \qquad \mu_j = \frac{1}{S} \sum_{i=l}^{r} \sum_{j=t}^{b} t(i,j) \cdot j$$

Position is the center of gravity: (μ_i, μ_j)

Second Moments:

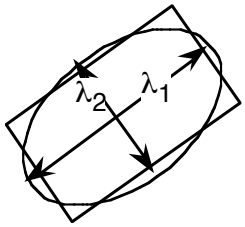
$$\sigma_i^2 = \frac{1}{S} \sum_{i=l}^{r} \sum_{j=t}^{b} t(i,j) \cdot (i-\mu_i)^2$$

$$\sigma_j^2 = \frac{1}{S} \sum_{i=l}^{r} \sum_{j=t}^{b} t(i,j) \cdot (j-\mu_j)^2$$

$$\sigma_{ij}^2 = \frac{1}{S} \sum_{i=l}^{r} \sum_{j=t}^{b} t(i,j) \cdot (i-\mu_i) \cdot (j-\mu_j)$$

These compose the covariance matrix: $$C = \begin{pmatrix} \sigma_i^2 & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_j^2 \end{pmatrix}$$

The principle components (λ_1, λ_2) determine the length and width.
The principle direction determines the orientation of the length.
We can discover these by principle components analysis.



$$\Phi C \Phi^T = \Lambda = \begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix}$$

where

$$\Phi = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

The length to width ratio, λ_1/λ_2, is an invariant for shape.

This suggests a "feature vector" for the blob: $\begin{pmatrix} x \\ y \\ w \\ h \\ \theta \end{pmatrix}$

where x= μ_i, $y = \mu_j$, $w=\lambda_1$, $h=\lambda_2$

and

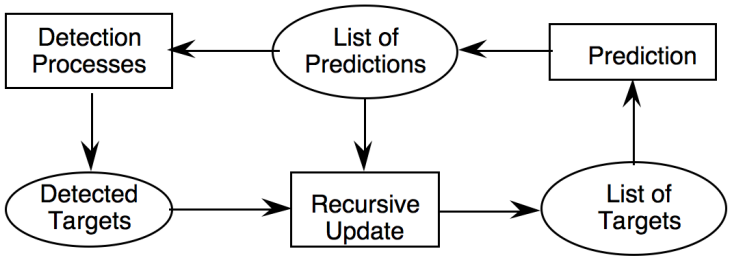
$$CF = \frac{S}{(b-t)(r-l)}$$

However, for tracking we need to keep explicit the center of gravity and covariance. Thus we will track:

Position: $\vec{\mu}_t = \begin{pmatrix} \mu_i \\ \mu_j \end{pmatrix}$ Size : $C_t = \begin{pmatrix} \sigma_i^2 & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_j^2 \end{pmatrix}$ along with CF_t.

2.7 Bayesian Estimation

A Bayesian tracker is a recursive estimator, composed of the phases:
Predict, Detect, estimate.
Having "detected a blob", next we need estimate the parameters.



The detection process can contain errors due to missed detection and false detection. To minimize the influence of errors we use the idea of a Gaussain window.

The Gaussian window is the previous covariance for the blob, enlarged by some "uncertainty" covariance. The uncertainty captures the possible loss of information during the time from the most recent observation.

Our Gaussian blob is

Position: $\vec{\mu}_t = \begin{pmatrix} \mu_i \\ \mu_j \end{pmatrix}$ Size : $C_t = \begin{pmatrix} \sigma_i^2 & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_j^2 \end{pmatrix}$ along with CF_t.

Let us represent the estimated blob at time t as: $\hat{\mu}_t,\ \hat{C}_t$
Let us estimate the predicted feature vector at time t as: $\vec{\mu}_t^*,\ C_t^*$

We will compute the estimated blob from by multiplying the detected pixels by a Gaussian mask determined from the predicted blob. The Covariance is multiplied by 2 to offset the fact that we will use mask to estimate a new covariance.

Gaussian Mask: $G(\vec{\mu}_t^*, 2C_t^*)$

Detected target pixels:

$$t(i,j) \leftarrow \frac{h_t(c(i,j))}{h(c(i,j))} \cdot e^{-\frac{1}{2}\left(\begin{pmatrix} i \\ j \end{pmatrix}-\begin{pmatrix} \mu_i \\ \mu_i \end{pmatrix}\right)^T 2C_t^{*-1}\left(\begin{pmatrix} i \\ j \end{pmatrix}-\begin{pmatrix} \mu_i \\ \mu_i \end{pmatrix}\right)}$$

We then estimate the new position and covariance as before:

First moments: $\mu_i = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j)\cdot i \qquad \mu_j = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j)\cdot j$

Second Moments:

$$\sigma_i^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j)\cdot (i-\mu_i)^2$$

$$\sigma_j^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j)\cdot (j-\mu_j)^2$$

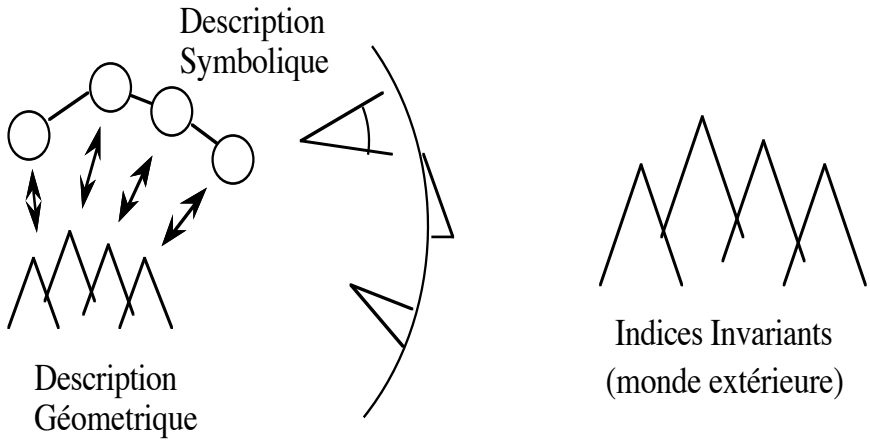
$$\sigma_{ij}^2 = \frac{1}{S}\sum_{i=l}^{r}\sum_{j=t}^{b} t(i,j)\cdot (i-\mu_i)\cdot (j-\mu_j)$$

Position: $\hat{\vec{\mu}}_t = \begin{pmatrix} \mu_i \\ \mu_j \end{pmatrix}$ Size : $\hat{C}_t = \begin{pmatrix} \sigma_i^2 & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_j^2 \end{pmatrix}$

3 Describing Image Contrast with Derivatives

An image is simply a large table of numerical values (pixels).
The "information" in the image may be found in the colors of regions of pixels, and the variations in intensity of pixels (contrast).

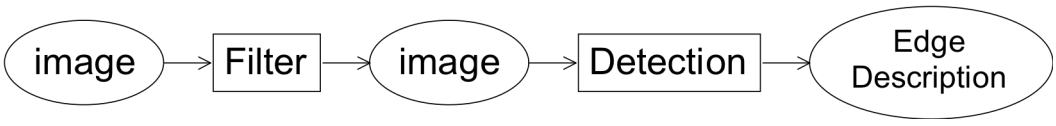
Extracting information from an image requires organizing these values into patterns that are "invariant" to changes in illumination and viewing direction.



Color provides information about regions of constant pigment.
Contrast provides information about 3D shape, as well as surface markings.

Contours of high contrast are referred to as "edges".

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



Two classic contrast detection operators are:

1) Roberts Cross Operator, and
2) The Sobel edge detector.

A more modern approach is to use Gaussian Derivatives

3.1 Roberts Cross Edge Detector

One of the earliest methods fore detecting image contrast (edges) was proposed by Larry Roberts in his 1962 Stanford Thesis (directed by Thomas Binford).

In this same thesis, Roberts introduced 3 fundamental techniques in Comptuer Vision:
1) the use of homogeneous coordinates for camera models,
2) wire frame scene models.
3) The Roberts Cross edge detector.

Roberts subsequently went to work for DARPA where he invented the "packet switching" communication technique and then managed the program that created the Arpanet (now known as the internet).

Roberts Cross operator employs two simple image filters:

$$m_1(i,j) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad m_2(i,j) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

These two operators are used as filters. They are convolved with the image.

Convolution (or filtering) : for n = 1, 2

$$E_n(i, j) = m_n * p(i,j) = \sum_{k=0}^{1}\sum_{l=0}^{1} m_n(k,l)\,p(i-k, j-l)$$

The contrast is the module of each pixel :

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

The direction of maximum contrast is the phase

$$\varphi(i,j) = Tan^{-1}\left(\frac{E_2(i,j)}{E_1(i,j)}\right) + \frac{\pi}{4}$$

Because of its small size and simplicity, the Roberts detector is VERY sensitive to high spatial-frequency noise. This is exactly the noise that is most present in images.

To reduce such noise, it is necessary to "smooth" the image with a low pass filter.
We can better understand the Roberts operators by looking at their Fourier Transform.

$$M_n(u,v) = \sum_{k=0}^{1}\sum_{l=0}^{1} m_n(k,l)e^{-j(ku+lv)}$$

$M_1(u,v) = (+1)\cdot e^{-j(-(0)u+-(0)v)} + (-1)\cdot e^{-j(u+v)} = 2j\ Sin(0.5u+0.5v)$

$M_2(u,v) = (+1)\cdot e^{-j(-(0)u+-(1)v)} + (-1)\cdot e^{-j(-(1)u+-(0)v)} = 2j\ Sin(0.5u-0.5v)$

3.2 The Sobel Detector

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

It is perhaps the most famous and widely used edge detector:

$$m_1(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad m_2(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Convolution (or filtering) : for n = 1, 2

$$E_n(i, j) = m_n * p(i,j) = \sum_{k=-1}^{1}\sum_{l=-1}^{1} m_n(k,l)p(i-k, j-l)$$

The contrast is the module of each pixel :

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

The direction of maximum contrast is the phase

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

Sobel's Edge Filters can be seen as a composition of a image derivative and a smoothing filter.

$$m_1(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

$$m_2(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

The filter $\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ is a form of image derivative.
The filter $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ is a binomial smoothing filter.

3.3 Difference Operators: Derivatives for Sampled Signals

For the function, s(x) the derivative can be defined as :

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

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

the limit does not exist, however we can observe

$$\Delta n = 1: \quad \frac{\Delta s(n)}{\Delta n} = \frac{s(n+1) - s(n)}{1} = s(n) * [-1 \quad 1]$$

$$\Delta n = 0: \quad \frac{\Delta s(n)}{\Delta n} = \frac{0}{0}$$

This is the operator used by Roberts.

If we use a Symmetric definition for the derivative:

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

then

$$\Delta n = 1: \quad \frac{\Delta s(n)}{\Delta n} = \frac{s(n+1) - s(n-1)}{1} = s(n) * [-1 \quad 0 \quad 1]$$

This is the operator used by Sobel.

Note that a derivative is equivalent to 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].