ECE5554 – Computer Vision
Lecture 2c – Edge Detection

Creed Jones, PhD

Today's Objectives

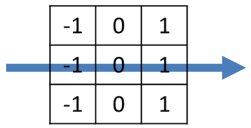
Edge detection

- Difference kernels
- Edges and their properties
- Edge detection kernels
- Canny edge detection
- Edge detection in color images

Consider a kernel with some positive and some negative values – what's the effect on the image?

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

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	20	20	20	0	0
0	0	20	20	20	0	0
0	0	20	20	20	20	20
0	0	20	20	20	20	20
0	0	20	20	20	20	20

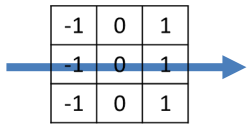


20	20	0	0	-20
40	40	0	0	-40
60	60	0	0	-40
60	60	0	0	-20
60	60	0	0	0

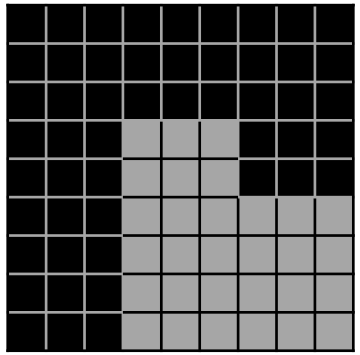
Consider a kernel with some positive and some negative values – what's the effect on the image?

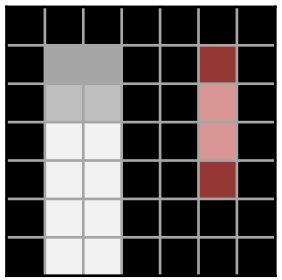
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	20	20	20	0	0	0
0	0	0	20	20	20	0	0	0
0	0	0	20	20	20	20	20	20
0	0	0	20	20	20	20	20	20
0	0	0	20	20	20	20	20	20
0	0	0	20	20	20	20	20	20

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

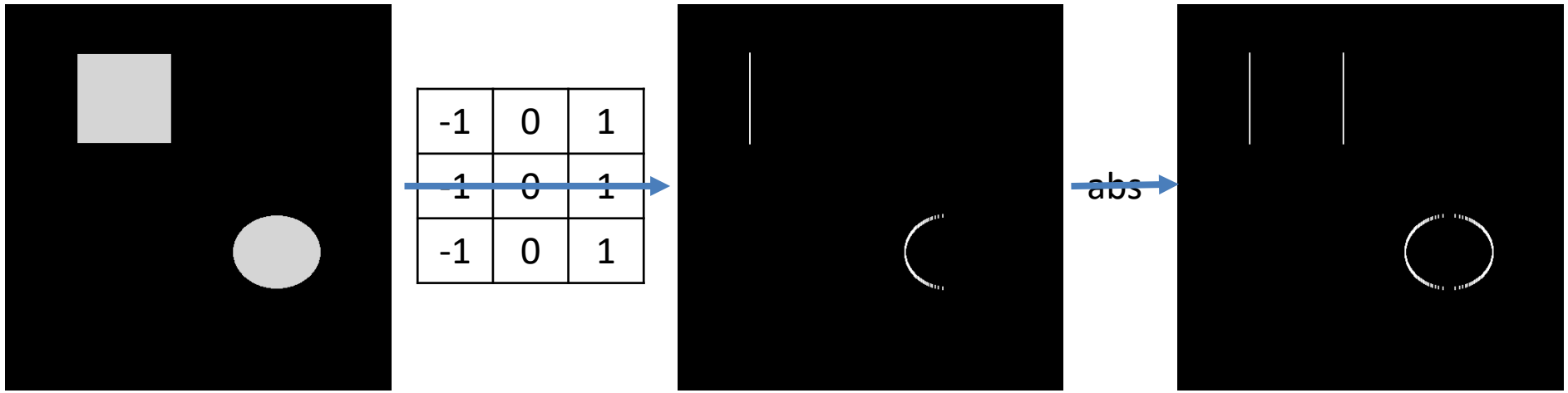


0	0	0	0	0	0	0
0	20	20	0	0	-20	0
0	40	40	0	0	-40	0
0	60	60	0	0	-40	0
0	60	60	0	0	-20	0
0	60	60	0	0	0	0
0	60	60	0	0	0	0

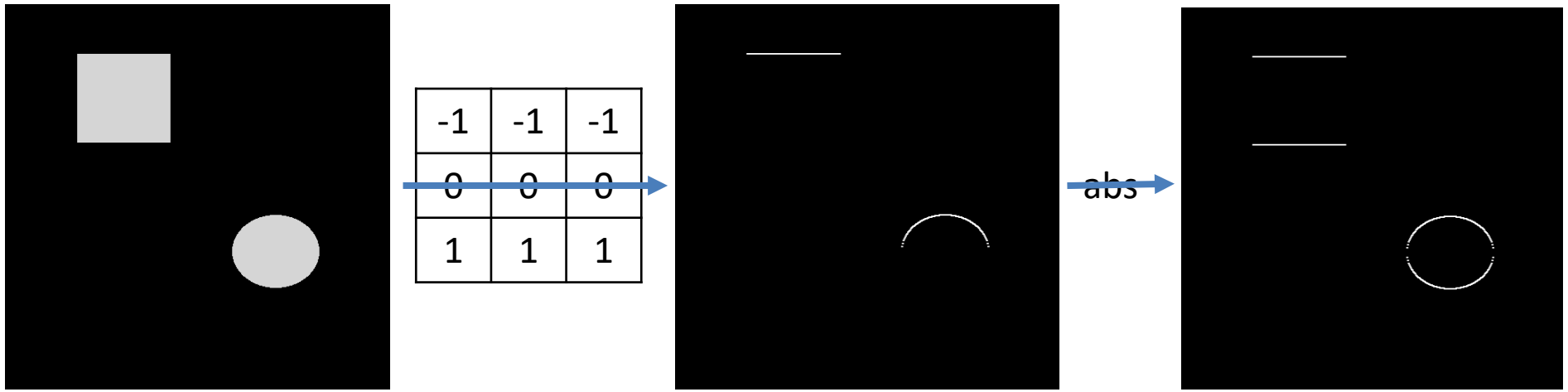




This 3x3 kernel highlights vertical *edges* – note that the result has both positive (dark to light) and negative (light to dark) values



The corresponding 3x3 vertical edge kernel has -1s on top and 1s on bottom – again, we can take the absolute value to show all transitions as light on a dark background

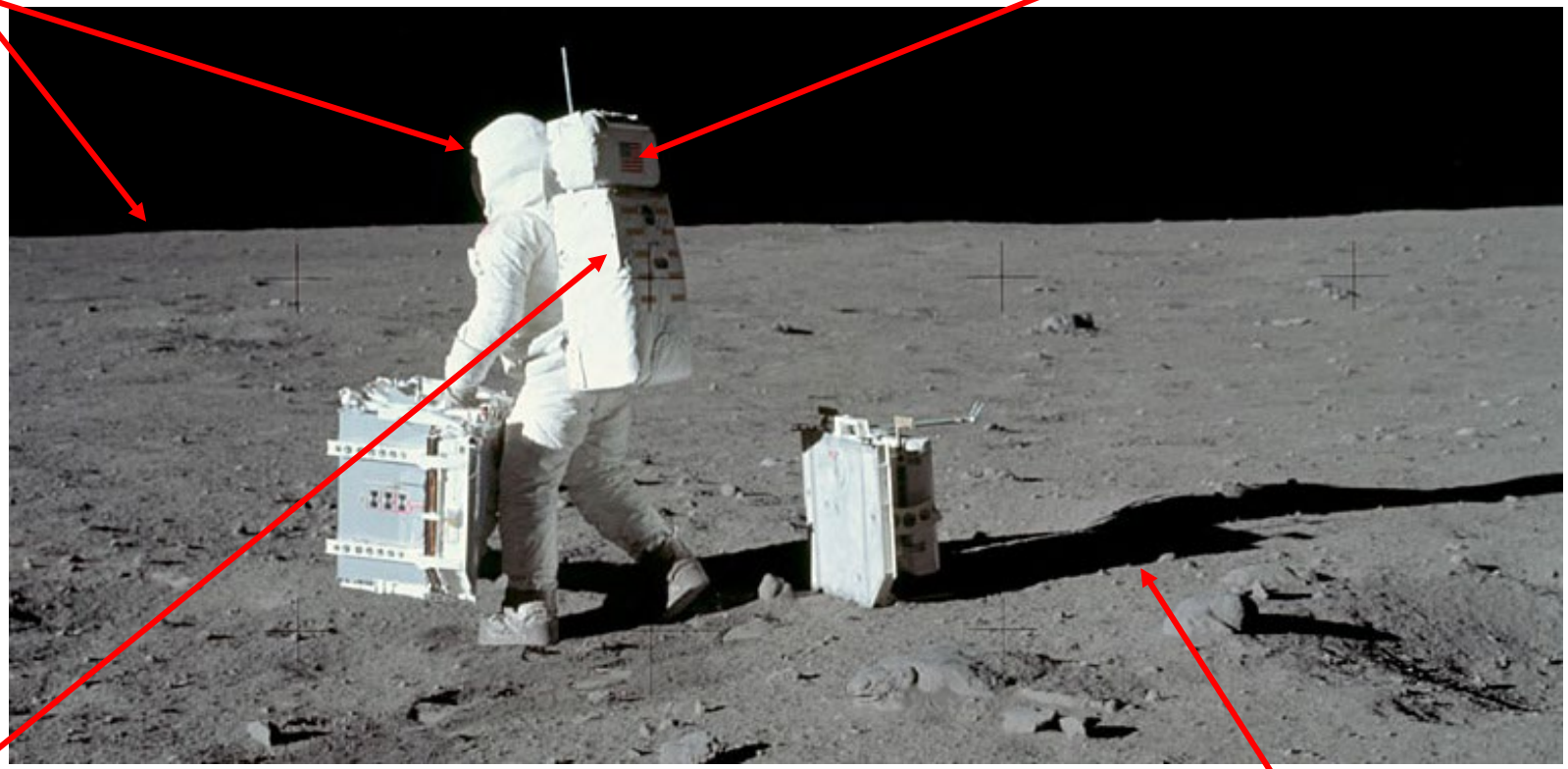


An **edge** is a sudden change in image intensity; Edge detection is very common in image analysis

What can cause an intensity edge?

Depth discontinuity

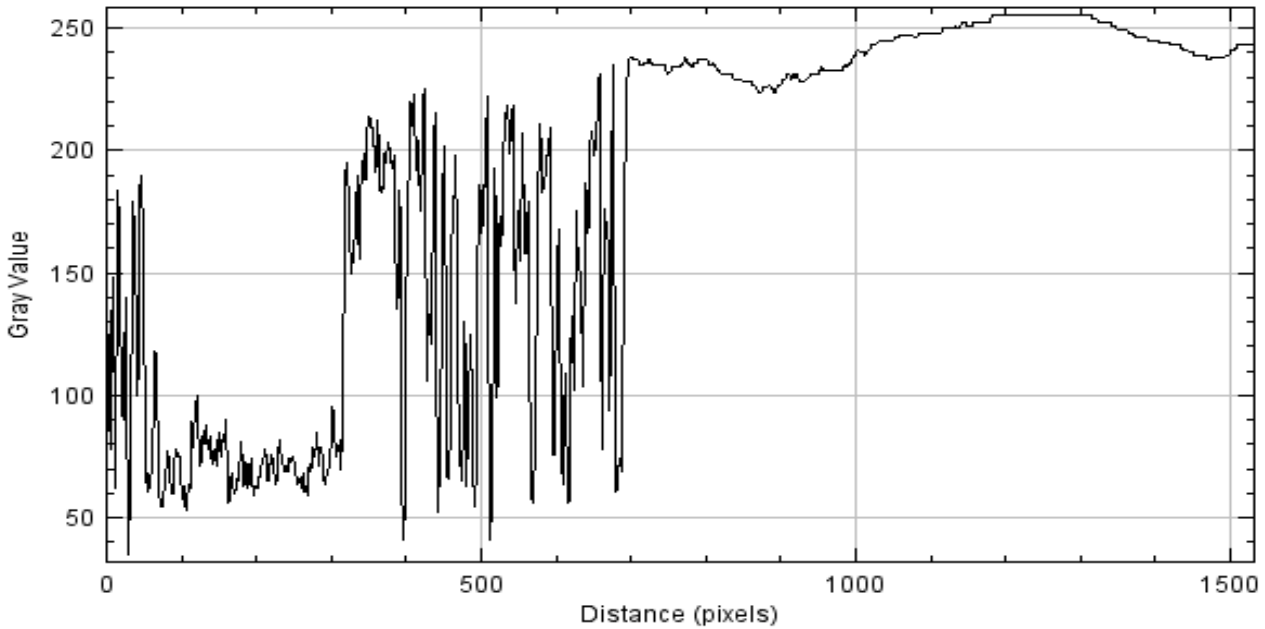
Sudden change in reflectance: here, paint on object's surface



Sudden change in 3D surface orientation

Sudden change in illumination: here, harsh shadows

- Consider a "profile" plot of intensity values, taken approximately from the row indicated
- This information is 1-dimensional
- We can let x represent the horizontal direction, and let $f(x)$ represent the image intensity
- How do you normally detect *change* in some function $f(x)$?
- By computing (or approximating) the *derivative*



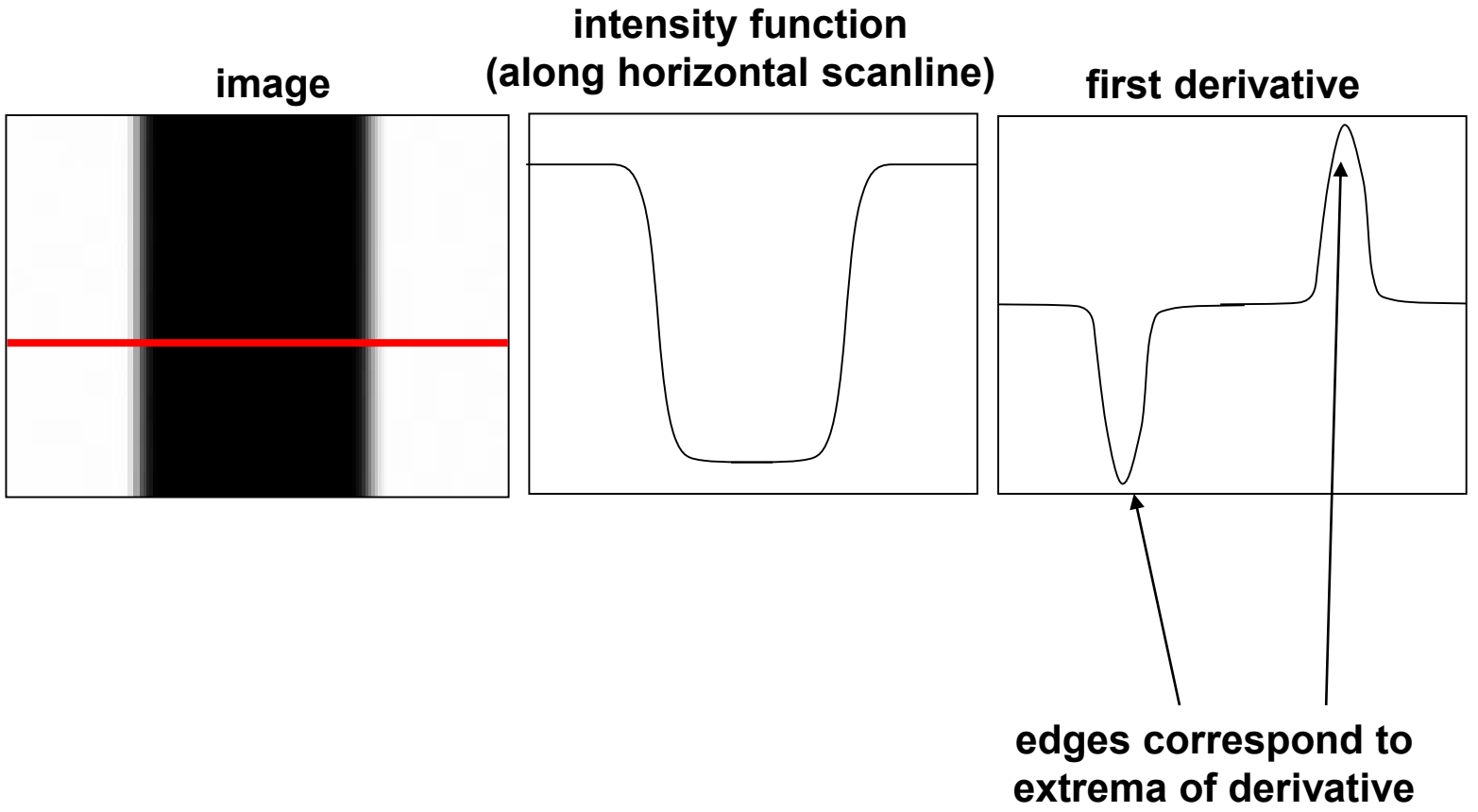
An edge is a place of rapid change in the image intensity function; we compute the spatial derivative to quantify (measure) the edge strength

- Ideal case:

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

- Discrete approximation:

$$\frac{df}{dx} \approx \frac{f(x + \Delta x) - f(x)}{\Delta x}$$



Now extend this idea to a 2-dimensional image I(x, y), and let Δx be related to the pixel spacing

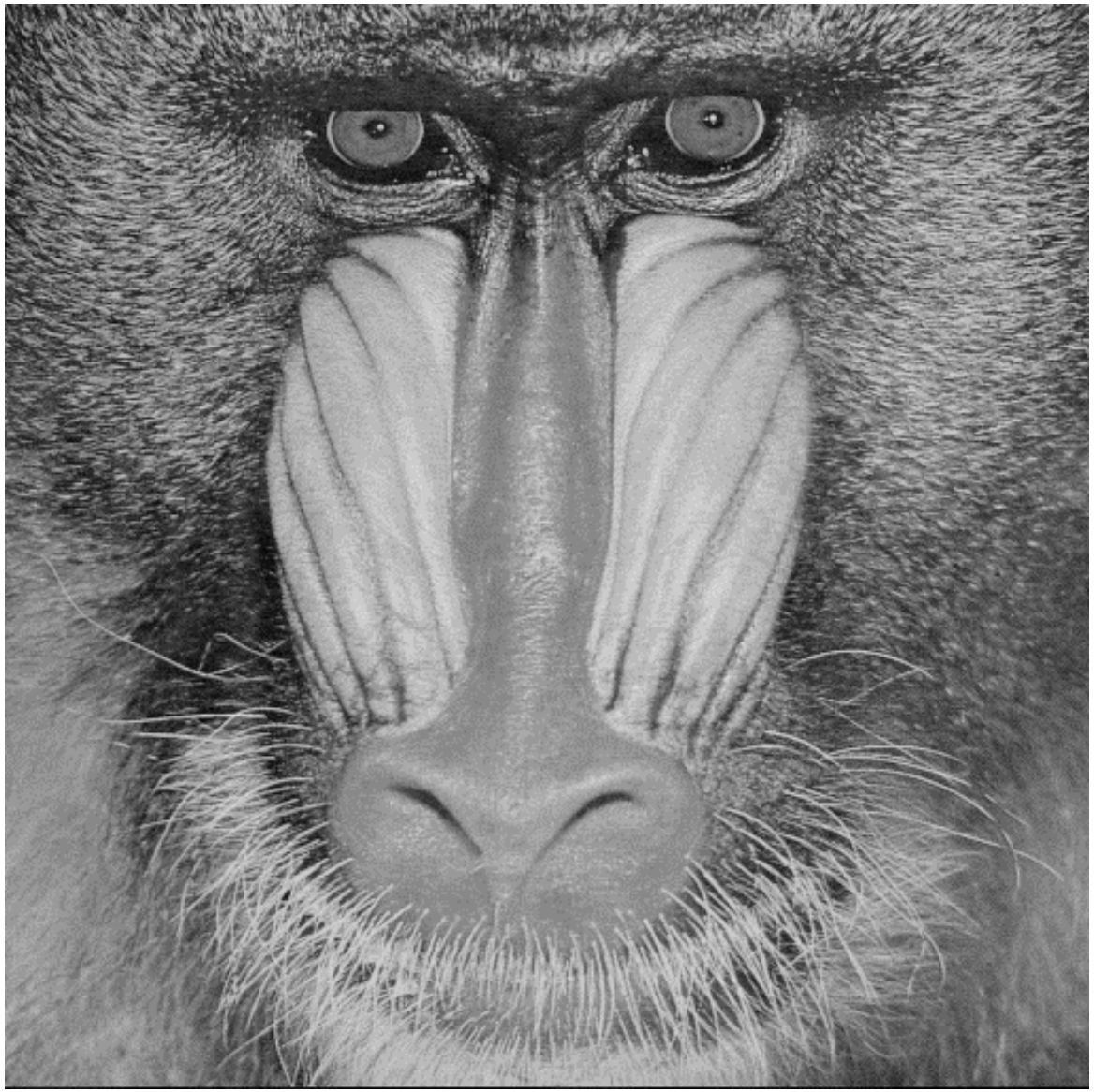
$$\frac{\partial I}{\partial x} = \frac{I(x + \Delta x, y) - I(x, y)}{\Delta x}$$

$$\frac{\partial I}{\partial y} = \frac{I(x, y + \Delta y) - I(x, y)}{\Delta y}$$

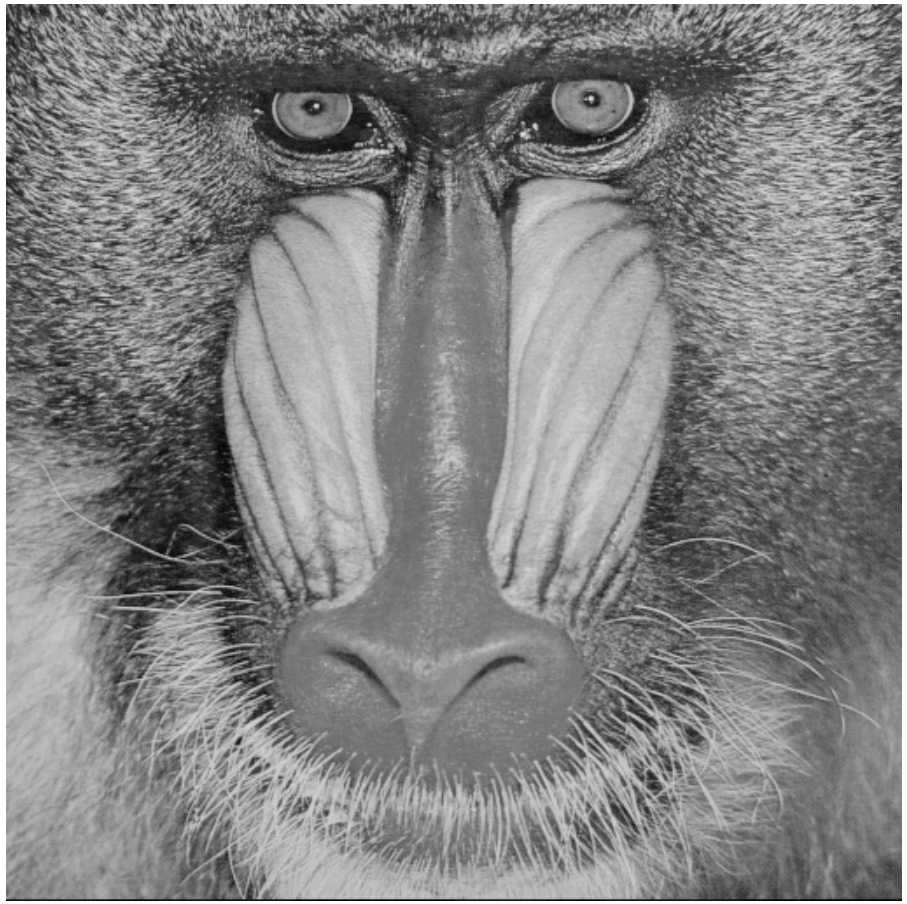
- Δx and Δy may be one pixel (one unit) or something more complex
- These are the x- and y- components of the rate of change of intensity

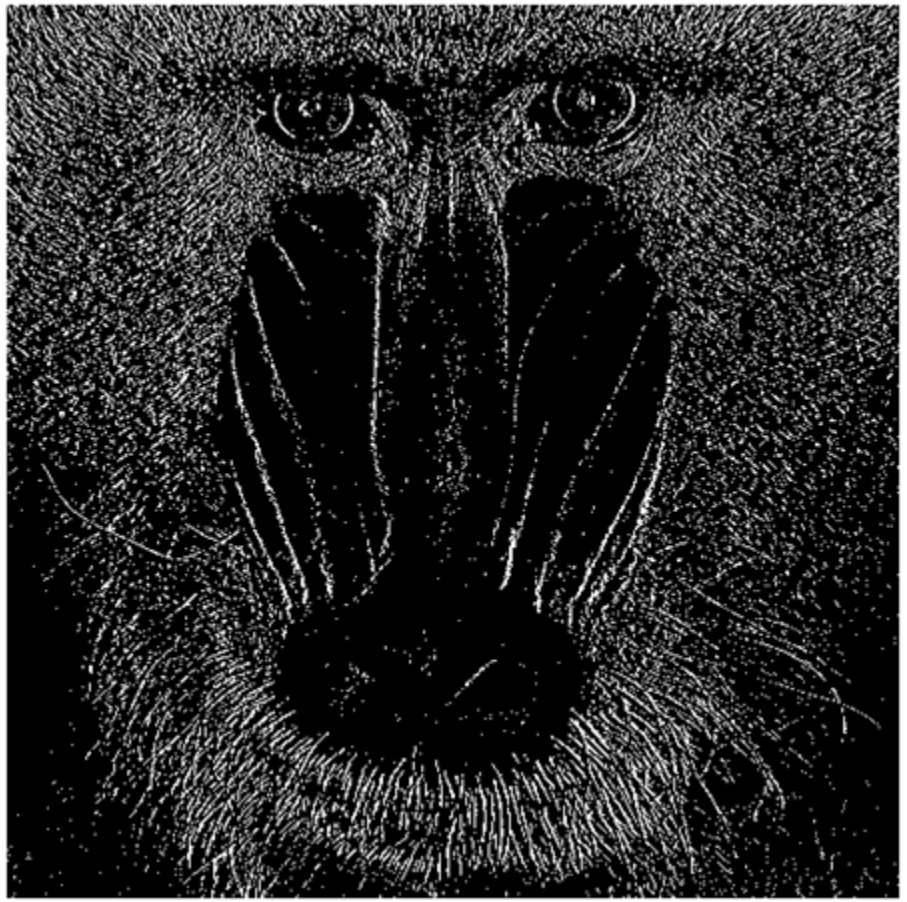
Mandrill: "a large fierce gregarious baboon of western Africa"

Mandrill: "a large fierce gregarious baboon of western Africa"



Compute $I_x(r,c) = I(r,c+1) - I(r,c)$ then take the absolute value, and then threshold (T = 20)





The derivative approximation $I_x(r,c) = I(r,c+1) - I(r,c)$ can be represented by a kernel operation

- Applying this 3x3 kernel will produce the same result

0	0	0
0	-1	1
0	0	0

- This isn't much of a kernel – but it does indicate that we can represent derivative operations as kernels…

Some of the earliest kernels for edge detection include the Roberts cross and the Prewitt kernel – they're OK, but there are much better methods

- The Roberts cross kernels are 2x2; they approximate the gradient in 2 directions (45 and 135 degrees)

-1	0
0	1

0	1
-1	0

- The Prewitt kernels are the ones we have seen on earlier slides – they approximate the gradient in the X and Y directions

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

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

A very common set of kernels for edge detection compose the Sobel operator; it produces two components – magnitude and angle of the edge

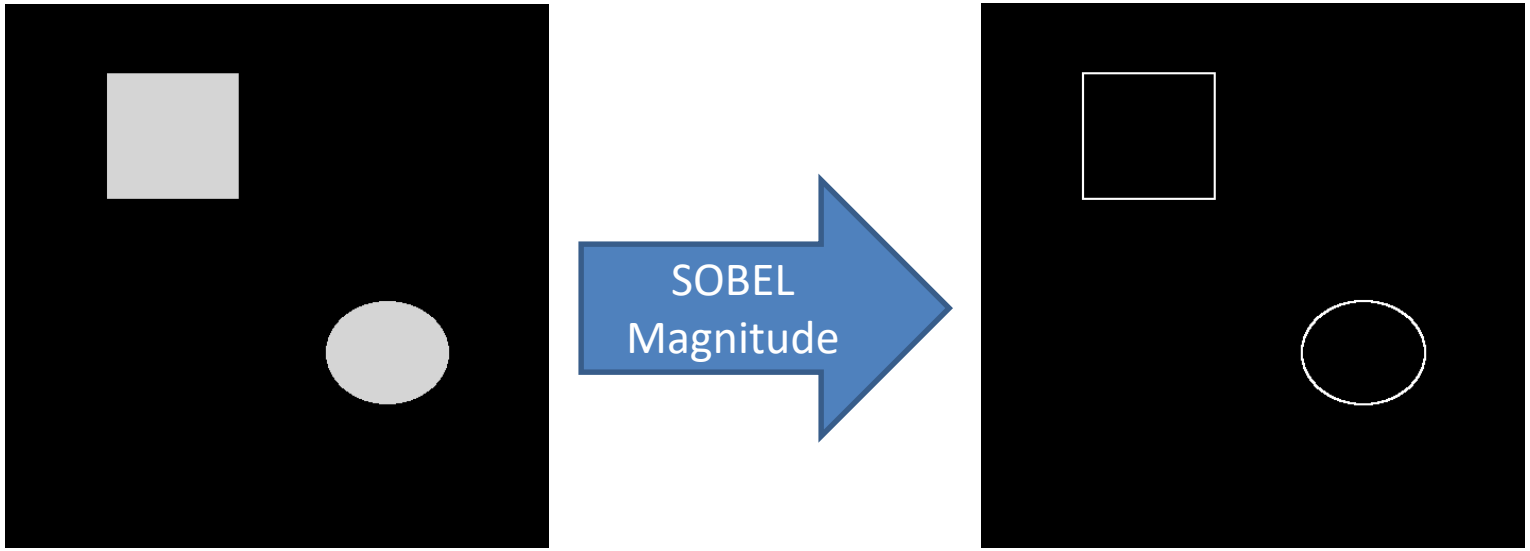
-1	0	+1
-2	0	+2
-1	0	+1

x filter

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

y filter

- Apply both the X and the Y filter to the image to produce two output images – one contains horizontal edge information and the other, vertical
- The Sobel magnitude image, at each point, is the square root of the sum of the squares of the X and Y edge images (the magnitude of the gradient vector)
- The angle is $\theta = \tan^{-1}(\frac{y_{edge}}{x_{edge}})$


SOBEL
Magnitude

The Sobel kernels also exist in larger sizes (though the difference in output is slight); here is one version of the 5 by 5 kernels

-5	-4	0	4	5
-8	-10	0	10	8
-10	-20	0	20	10
-8	-10	0	10	8
-5	-4	0	4	5

-5	-8	-10	-8	-5
-4	-10	-20	-10	-4
0	0	0	0	0
4	10	20	10	4
5	8	10	8	5

The Sobel kernels are popular because they are close to circular – but there are others that are closer

- The *Scharr* kernels were specifically designed to have good rotational symmetry
- 3x3 Scharr:

-3	0	3
-10	0	10
-3	0	3

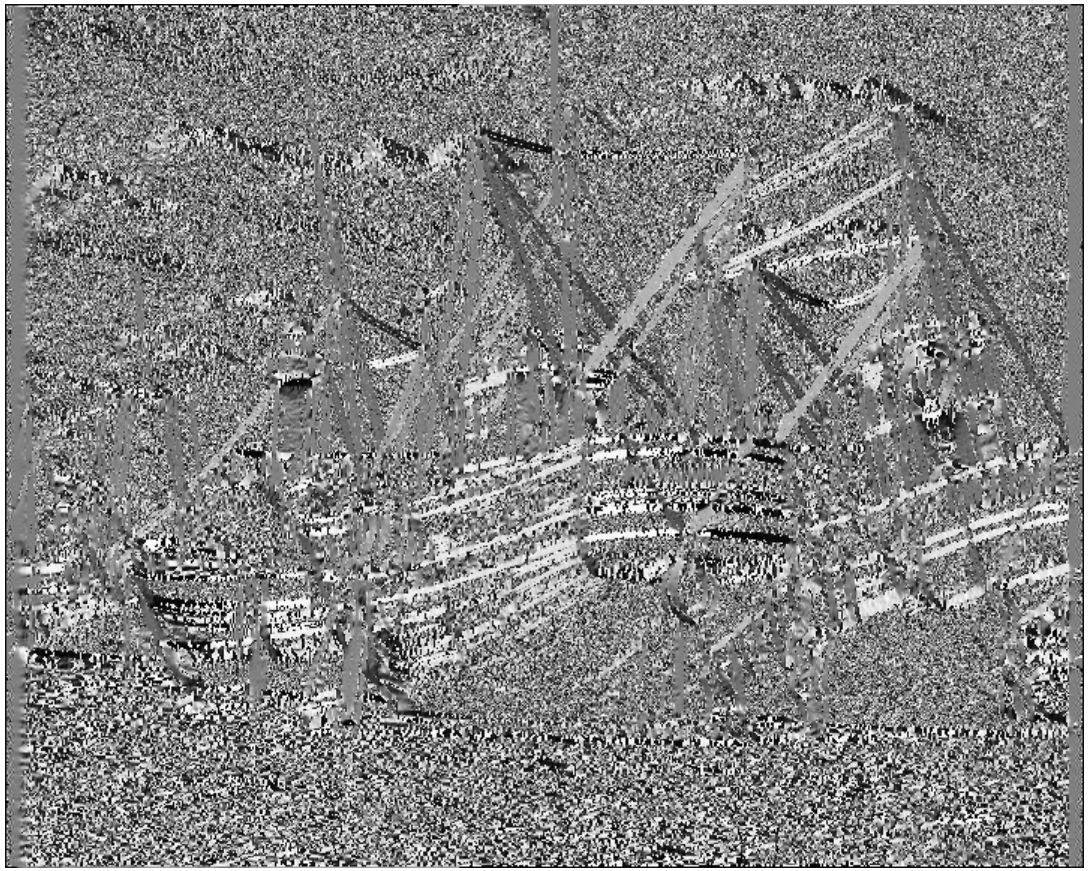
-3	-10	-3
0	0	0
3	10	3

- 5x5 Scharr:

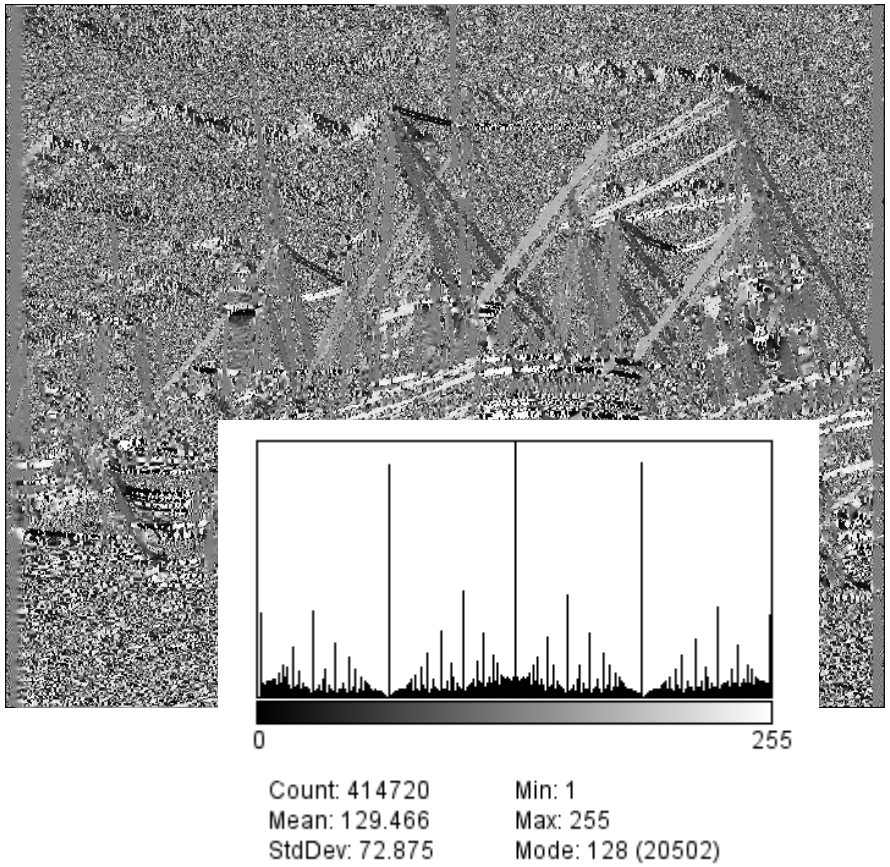
-1	-1	0	1	1
-2	-2	0	2	2
-3	-6	0	6	3
-2	-2	0	2	2
-1	-1	0	1	1

-1	-2	-3	-2	-1
-1	-2	-6	-2	-1
0	0	0	0	0
1	2	6	2	1
1	2	3	2	1

What does the *phase component* of the Sobel gradient look like? As an angle, it naturally ranges from $-\pi$ to π, so we scale it the range 0 to 255 for display; but remember, angle is a circular quantity!



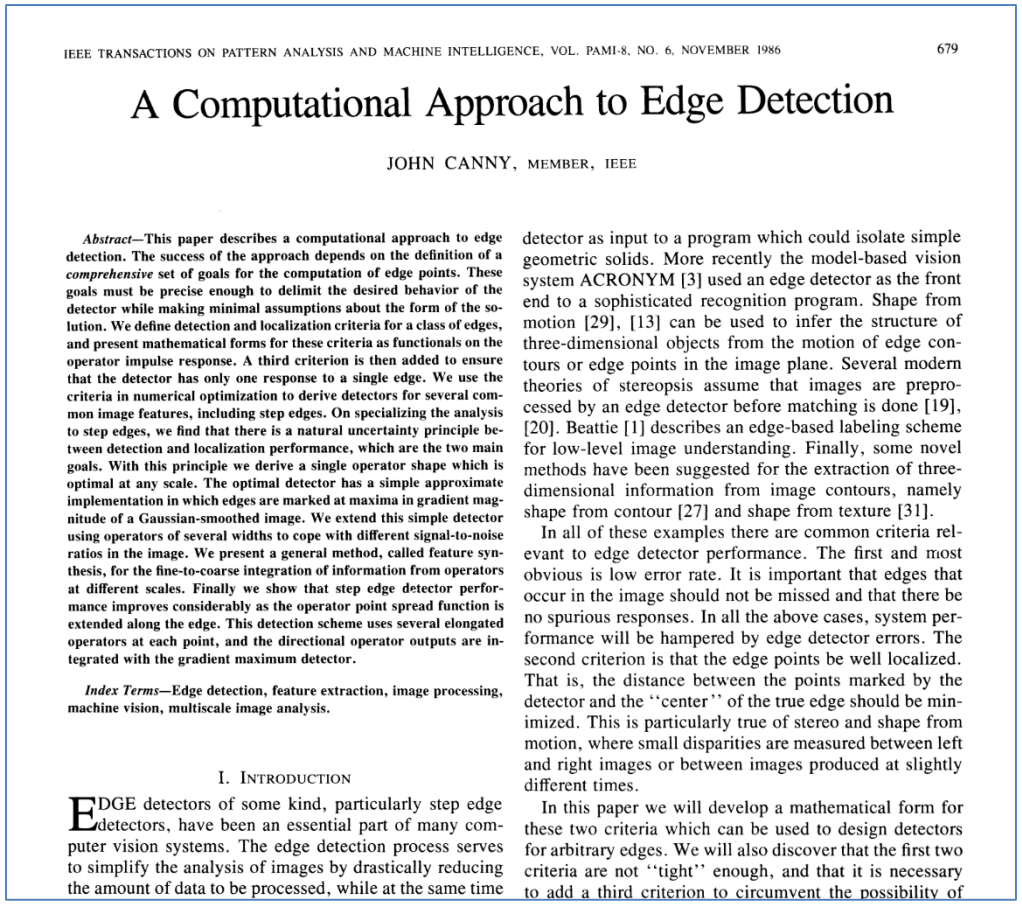
What does the *phase component* of the Sobel gradient look like? As an angle, it naturally ranges from $-\pi$ to π, so we scale it the range 0 to 255 for display; but remember, angle is a circular quantity!



So, when is the angle component ever used? If you want an image consisting of strong edges that are diagonal, then write out the Sobel magnitude times the difference of the angle from 45 degrees

CANNY EDGE DETECTION

A widely used and comprehensive method for computing an edge image was developed by Canny in the late 1980's



IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE

Abstract—This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a *comprehensive* set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the operator impulse response. A third criterion is then added to ensure that the detector has only one response to a single edge. We use the criteria in numerical optimization to derive detectors for several common image features, including step edges. On specializing the analysis to step edges, we find that there is a natural uncertainty principle between detection and localization performance, which are the two main goals. With this principle we derive a single operator shape which is optimal at any scale. The optimal detector has a simple approximate implementation in which edges are marked at maxima in gradient magnitude of a Gaussian-smoothed image. We extend this simple detector using operators of several widths to cope with different signal-to-noise ratios in the image. We present a general method, called feature synthesis, for the fine-to-coarse integration of information from operators at different scales. Finally we show that step edge detector performance improves considerably as the operator point spread function is extended along the edge. This detection scheme uses several elongated operators at each point, and the directional operator outputs are integrated with the gradient maximum detector.

Index Terms—Edge detection, feature extraction, image processing, machine vision, multiscale image analysis.

I. Introduction

EDGE detectors of some kind, particularly step edge detectors, have been an essential part of many computer vision systems. The edge detection process serves to simplify the analysis of images by drastically reducing the amount of data to be processed, while at the same time detector as input to a program which could isolate simple geometric solids. More recently the model-based vision system ACRONYM [3] used an edge detector as the front end to a sophisticated recognition program. Shape from motion [29], [13] can be used to infer the structure of three-dimensional objects from the motion of edge contours or edge points in the image plane. Several modern theories of stereopsis assume that images are preprocessed by an edge detector before matching is done [19], [20]. Beattie [1] describes an edge-based labeling scheme for low-level image understanding. Finally, some novel methods have been suggested for the extraction of three-dimensional information from image contours, namely shape from contour [27] and shape from texture [31].

In all of these examples there are common criteria relevant to edge detector performance. The first and most obvious is low error rate. It is important that edges that occur in the image should not be missed and that there be no spurious responses. In all the above cases, system performance will be hampered by edge detector errors. The second criterion is that the edge points be well localized. That is, the distance between the points marked by the detector and the "center" of the true edge should be minimized. This is particularly true of stereo and shape from motion, where small disparities are measured between left and right images or between images produced at slightly different times.

In this paper we will develop a mathematical form for these two criteria which can be used to design detectors for arbitrary edges. We will also discover that the first two criteria are not "tight" enough, and that it is necessary to add a third criterion to circumvent the possibility of

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf

Canny's idea was to find the edge pixels that satisfied a few criteria on what a "good" representation of the edge would look like – assuming a noisy image

- Low error rate
  - Don't mark any non-edge pixels as edges
  - Don't miss marking any edge pixels
- Good localization
  - Edge pixels correspond well to the actual edges of objects in the image
- No multiple responses
  - We want to find a single-pixel wide outline of each edge – not a "thick" outline

Notice that John Canny was still a grad student when he developed and published this method

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE

Abstract—**This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a *comprehensive* set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the operator impulse response. A third criterion is then added to ensure that the detector has only one response to a single edge. We use the criteria in numerical optimization to derive detectors for several common image features, including step edges. On specializing the analysis to step edges, we find that there is a natural uncertainty principle between detection and localization performance, which are the two main goals. With this principle we derive a single operator shape which is optimal at any scale. The optimal detector has a simple approximate implementation in which edges are marked at maxima in gradient magnitude of a Gaussian-smoothed image. We extend this simple detector using operators of several widths to cope with different signal-to-noise ratios in the image. We present a general method, called feature synthesis, for the fine-to-coarse integration of information from operators at different scales. Finally we show that step edge detector performance improves considerably as the operator point spread function is extended along the edge. This detection scheme uses several elongated operators at each point, and the directional operator outputs are integrated with the gradient maximum detector.**

Index Terms—**Edge detection, feature extraction, image processing, machine vision, multiscale image analysis.**

I. Introduction

EDGE detectors of some kind, particularly step edge detectors, have been an essential part of many computer vision systems. The edge detection process serves to simplify the analysis of images by drastically reducing the amount of data to be processed, while at the same time detector as input to a program which could isolate simple geometric solids. More recently the model-based vision system ACRONYM [3] used an edge detector as the front end to a sophisticated recognition program. Shape from motion [29], [13] can be used to infer the structure of three-dimensional objects from the motion of edge contours or edge points in the image plane. Several modern theories of stereopsis assume that images are preprocessed by an edge detector before matching is done [19], [20]. Beattie [1] describes an edge-based labeling scheme for low-level image understanding. Finally, some novel methods have been suggested for the extraction of three-dimensional information from image contours, namely shape from contour [27] and shape from texture [31].

In all of these examples there are common criteria relevant to edge detector performance. The first and most obvious is low error rate. It is important that edges that occur in the image should not be missed and that there be no spurious responses. In all the above cases, system performance will be hampered by edge detector errors. The second criterion is that the edge points be well localized. That is, the distance between the points marked by the detector and the "center" of the true edge should be minimized. This is particularly true of stereo and shape from motion, where small disparities are measured between left and right images or between images produced at slightly different times.

In this paper we will develop a mathematical form for these two criteria which can be used to design detectors for arbitrary edges. We will also discover that the first two criteria are not "tight" enough, and that it is necessary to add a third criterion to circumvent the possibility of

John Canny (S'81–M'82) was born in Adelaide, Australia, in 1958. He received the B.Sc. degree in computer science and the B.E. degree from Adelaide University in 1980 and 1981, respectively, and the S.M. degree from the Massachusetts Institute of Technology, Cambridge, in 1983.

He is with the Artificial Intelligence Laboratory, M.I.T. His research interests include low-level vision, model-based vision, motion planning for robots, and computer algebra.

Mr. Canny is a student member of the Association for Computing Machinery.

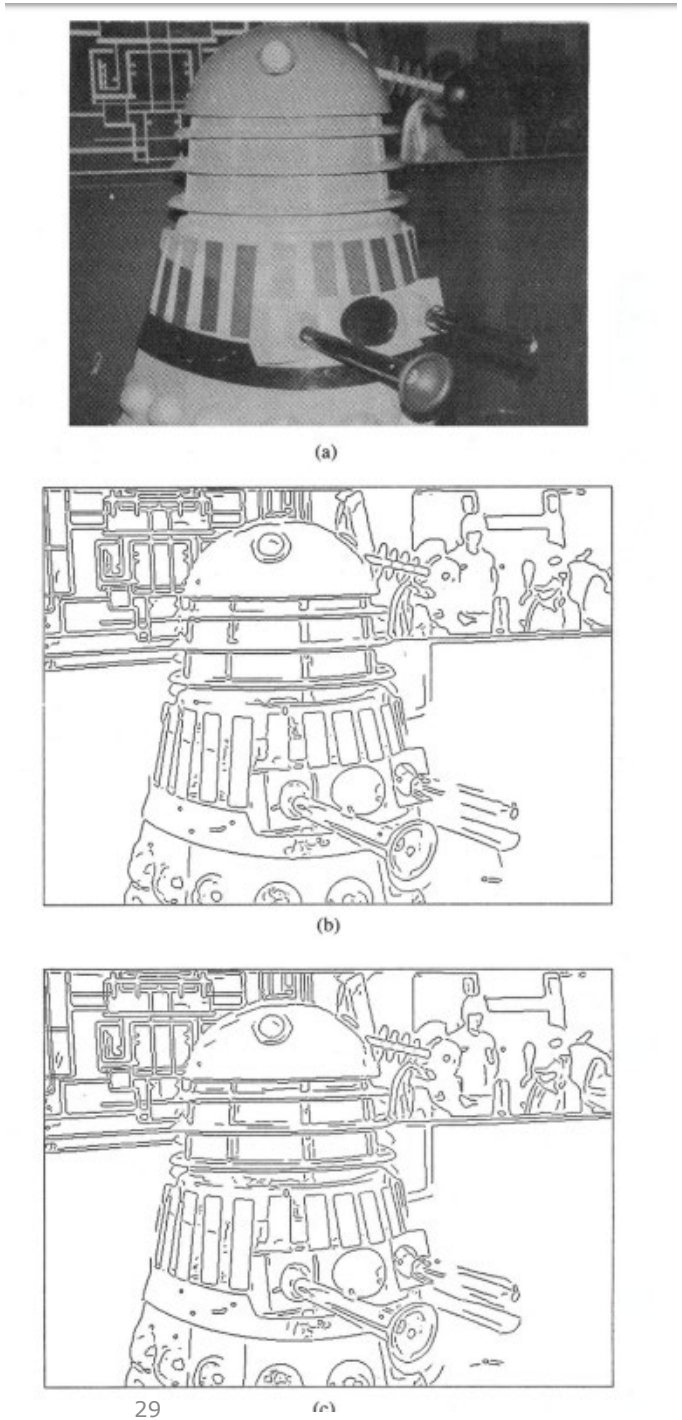
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf

The Canny edge detection process uses both an estimate of the gradient and the local content of the edge image to produce a *skeleton* of the edge

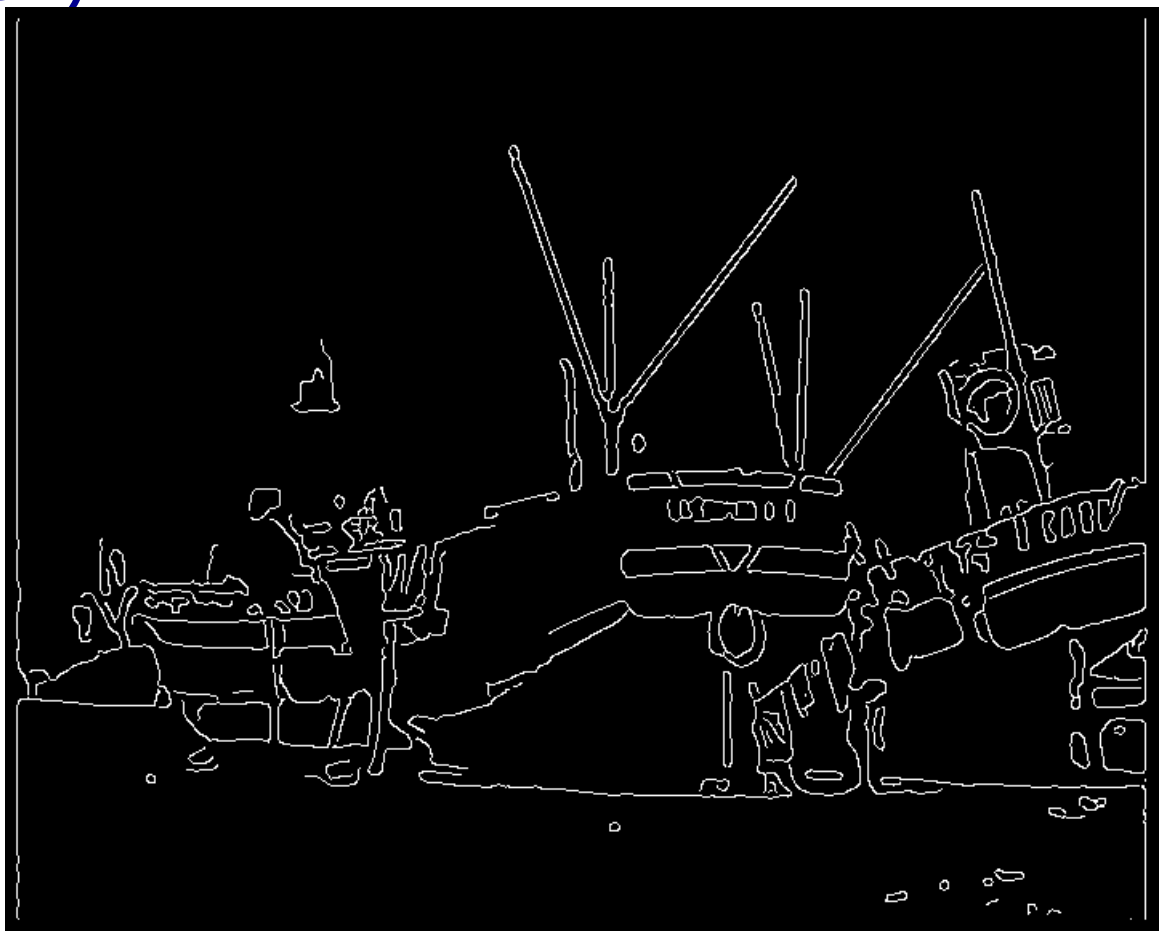
There are five steps to the Canny edge process:

- The image is smoothed using a Gaussian filter
- The gradient is approximated (usually using Sobel kernels)
- Non-maximum suppression thins the edge to its max value
- A dual threshold is applied to binarize
- Hysteresis-based edge tracking ensures connectivity

The results of Canny edge detection are often (but not always) shown as dark-on-light images, as seen here in Canny's paper



Canny edge detection results are binary, think edges along each object – parameters must be set (width of Gaussian, and upper and lower thresholds for the edge strength)

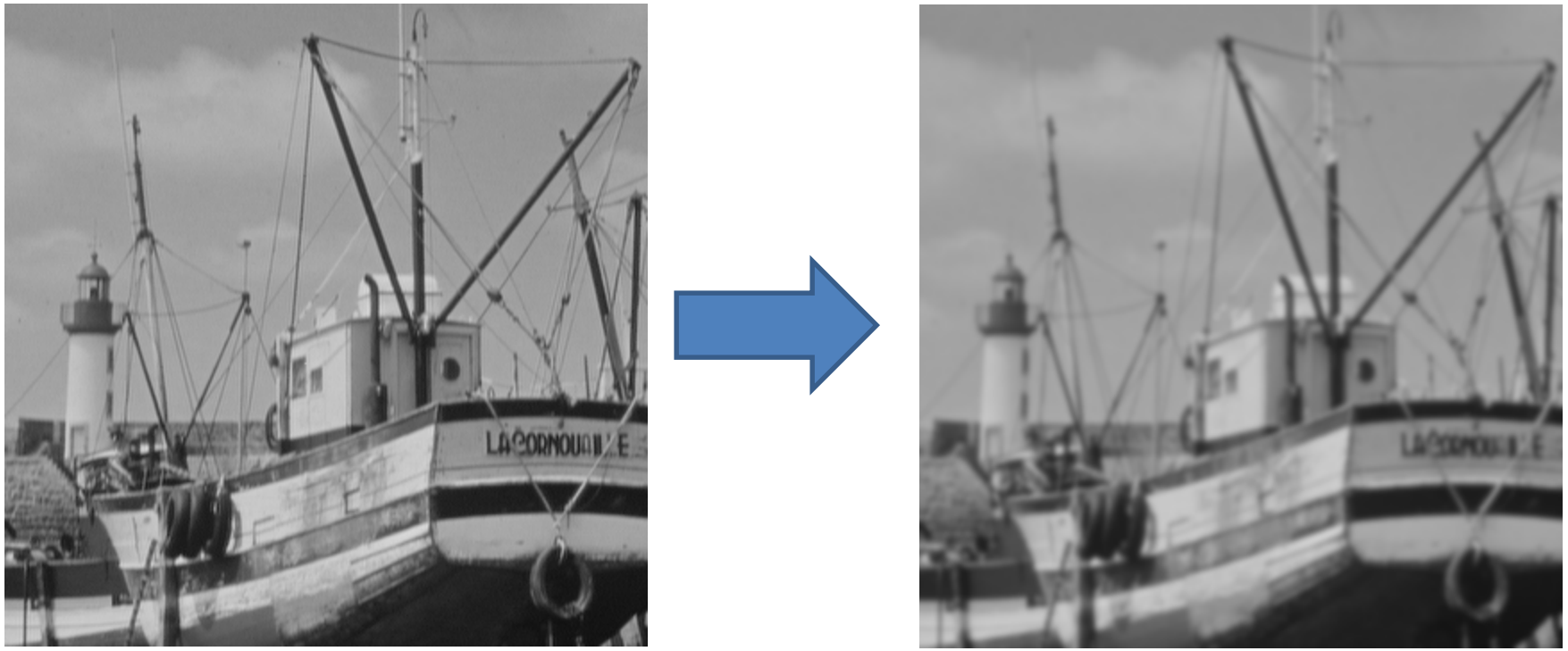


First step: the image is smoothed using a Gaussian filter – but what is the right σ to use?

- If sigma is small, then the final edges will be sensitive to noise
  - but edges will be located more accurate
- If sigma is large, then image noise will be more suppressed
  - but edge localization may be less accurate
- Canny recommended that $\frac{d}{\sigma} = 1.4$, where d is the sample (pixel) spacing
- A popular value is $\sigma = 1.4$ pixels;

$$\frac{1}{115} \times \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

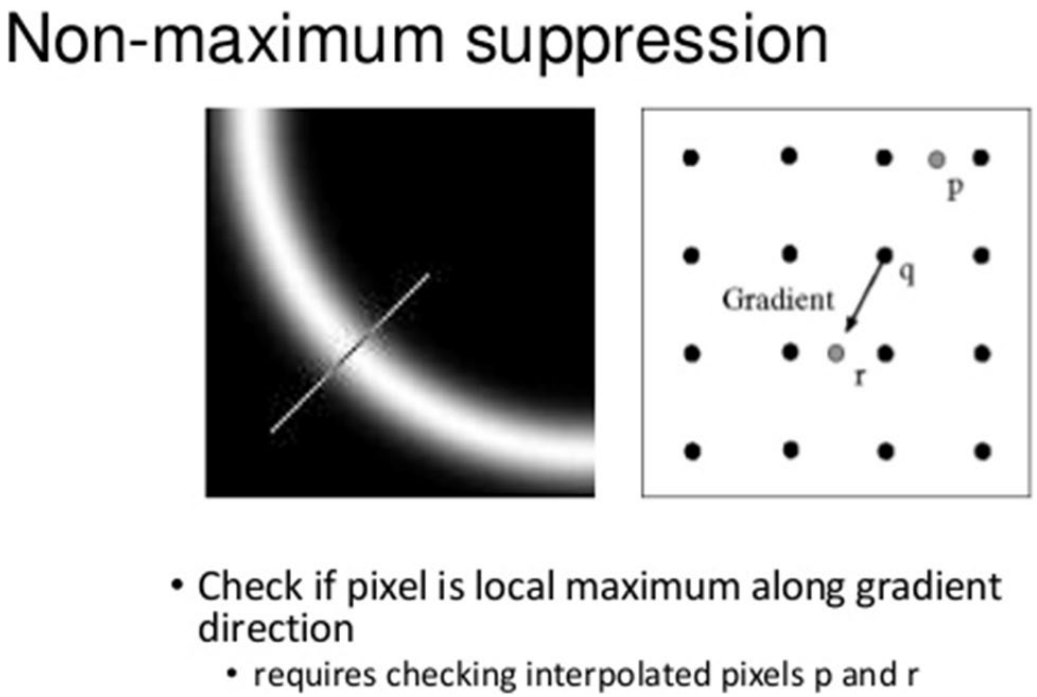
Canny step 1: the original image is smoothed using a Gaussian kernel – in this case, $\sigma = 1.4$



Canny step 2: the gradient is approximated (usually using Sobel kernels); both magnitude and phase are extracted

Canny step 3: compare the Sobel magnitude (edge strength) to each pixel in the neighborhood in the direction of max gradient and suppress (set to zero) all edge pixels that are not the maximum

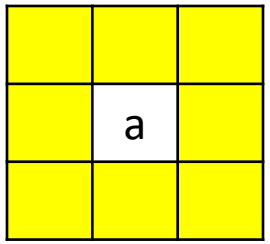
- This step has the result of thinning the edge to only the points of maximum gradient



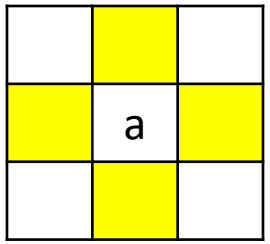
Canny step 4: a dual threshold is applied to binarize with *hysteresis*: a lower edge strength is required if a higher edge strength pixel is in the neighborhood

- First, threshold the image with an initial, higher threshold – any pixel with edge intensity above this threshold is marked as an edge pixel
- Then, re-examine the edge image – any pixel with edge intensity above a second, lower threshold that is connected to a pixel that is above the first threshold is also marked as an edge pixel
- For a 3x3, there are two kinds of connectivity (or neighbor)
  - usually, 8-connectivity is used

8 neighbors of pixel a
(8-connectivity)



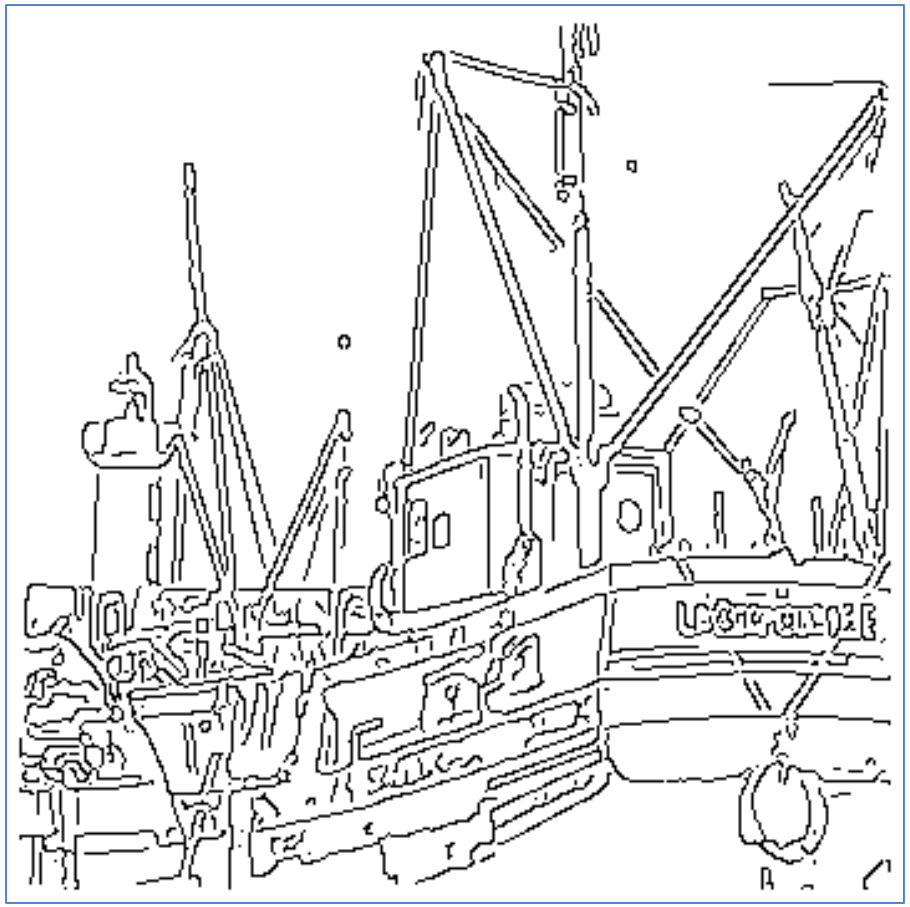
4 neighbors of pixel a
(4-connectivity)



Canny step 5: edge tracking ensures connectivity

- Sometimes, the edge is tracked (via a contour-following process) to either remove pixels below the high threshold, under certain circumstances, or to fill in gaps below a certain size
- This step is not always done, as it makes some assumptions about the content of the image

The result of the Canny edge detection has single-pixel wide "skeletons" of the intensity transitions in the image – useful for object definition



Original image

Smoothened image

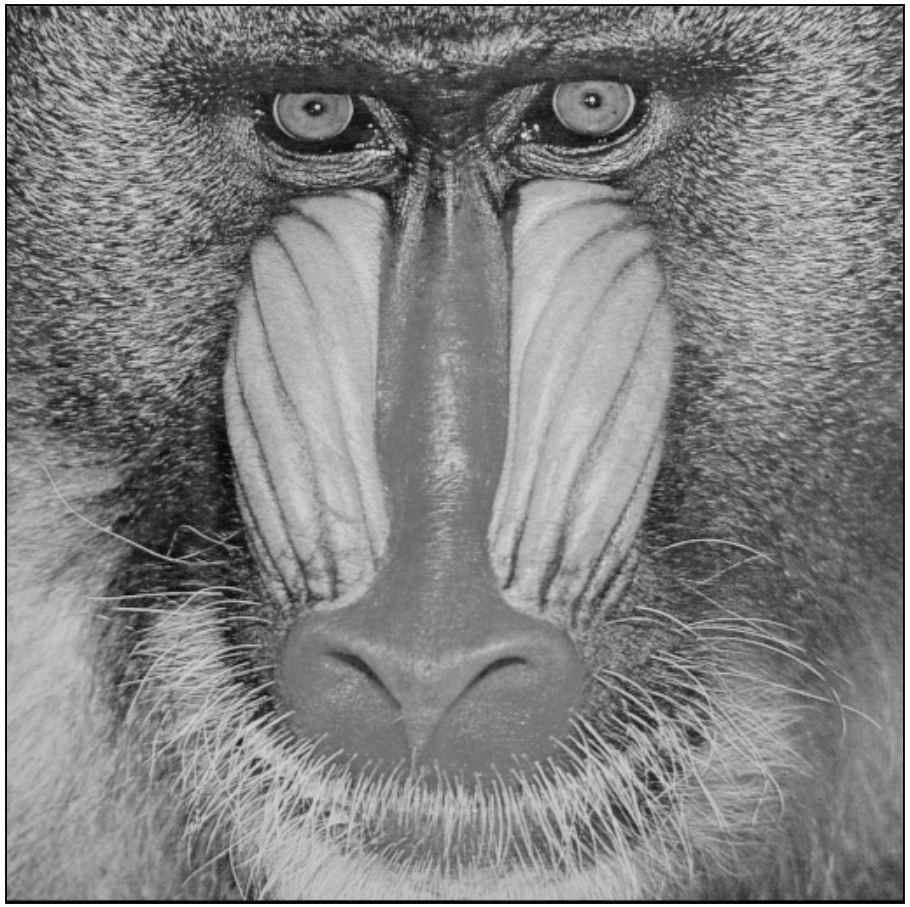
X Gradient

Y Gradient

NonMaximum Suppression

Final Image

Original image



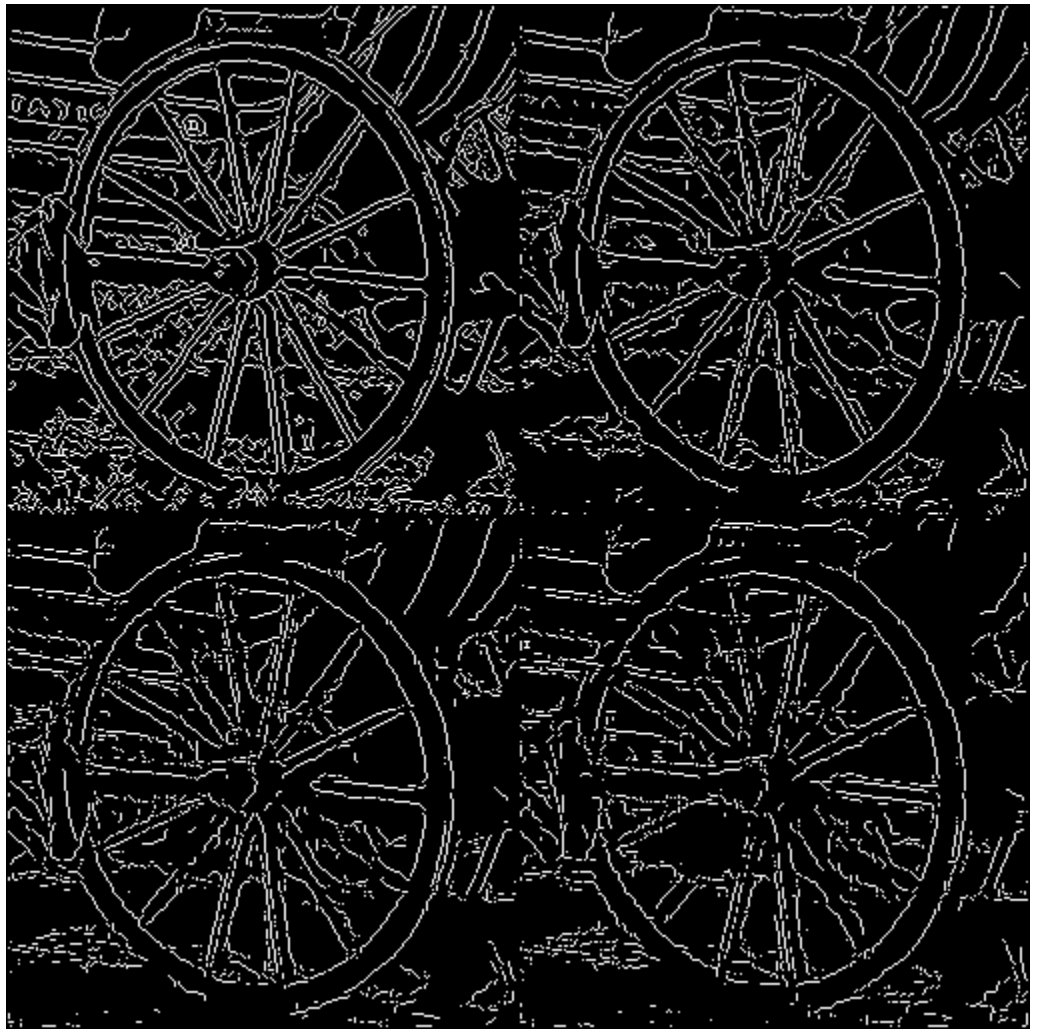
Canny example

```
>> im2=edge(im1,'canny');
```

Canny example

$\sigma = 1$

$\sigma = 3$



$\sigma = 2$

$\sigma = 4$

Things to note about the Canny edge detection process

- Note that the edges are outlines of objects
- The edge image is binary and edge are one-pixel wide
  - No measure of edge strength remains – but we can get this from the Sobel magnitude image
- Many methods make use of the Canny to identify edge locations and then look in other related images for correlated information
  - Sobel magnitude for edge strength
  - Sobel angle for edge direction
  - Original image for object intensity
- Edge polarity is lost – no distinction between light-to-dark and dark-to-light transitions
  - though the method can be modified to filter on polarity

People have proposed and implemented improvements to the Canny edge process

- Deriche (1987) replaced the Sobel with a different edge operator, designed to optimize the Canny criterion
  - He replaced a FIR (Finite Impulse Response) filter with an IIR (Infinite Impulse Response) filter
  - This has computational advantages but is less well-tempered on edges of all directions

- Others have proposed enhancements designed to better reject noise
  - Wang and He 2004 - http://en.cnki.com.cn/Article_en/CJFDTOTAL-ZGTB200408010.htm
  - Rong, Li and Zhang 2014 - http://ieeexplore.ieee.org/abstract/document/6885761/

COLOR EDGE DETECTION

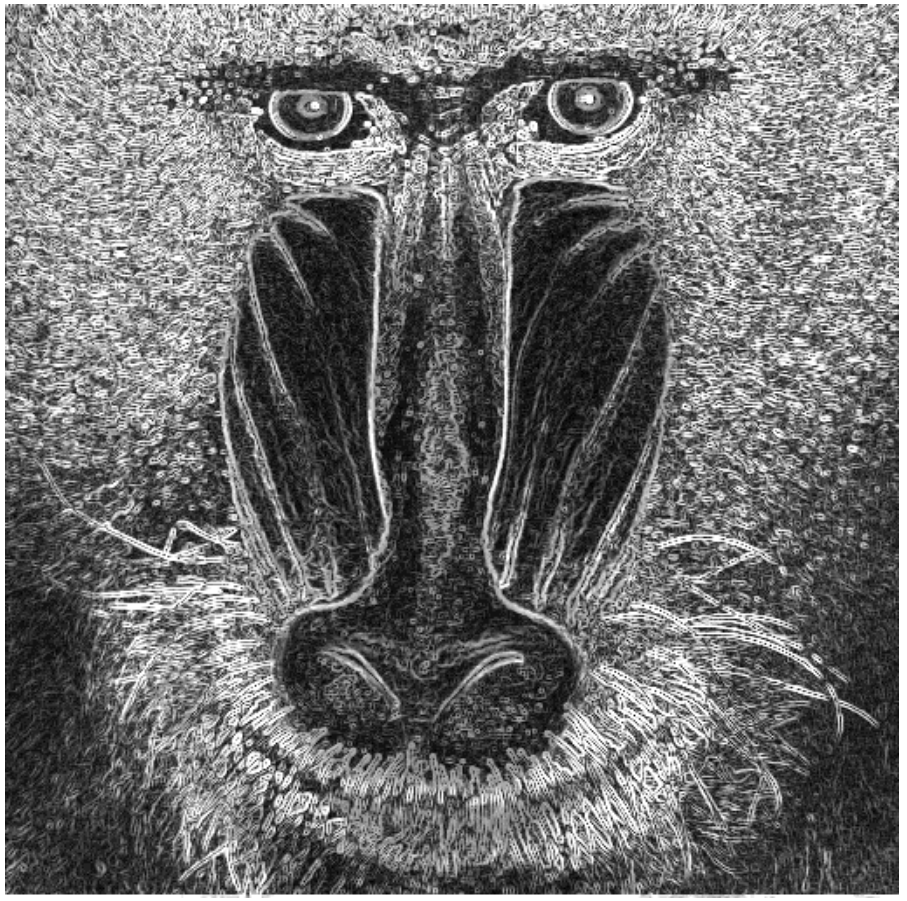
Recall our conclusions on filtering: smoothing RGB planes can be acceptable but not HSI planes

- Filtering methods that work with the vector quantities of the color values are also in existence
- For smoothing, it's generally not worth the effort

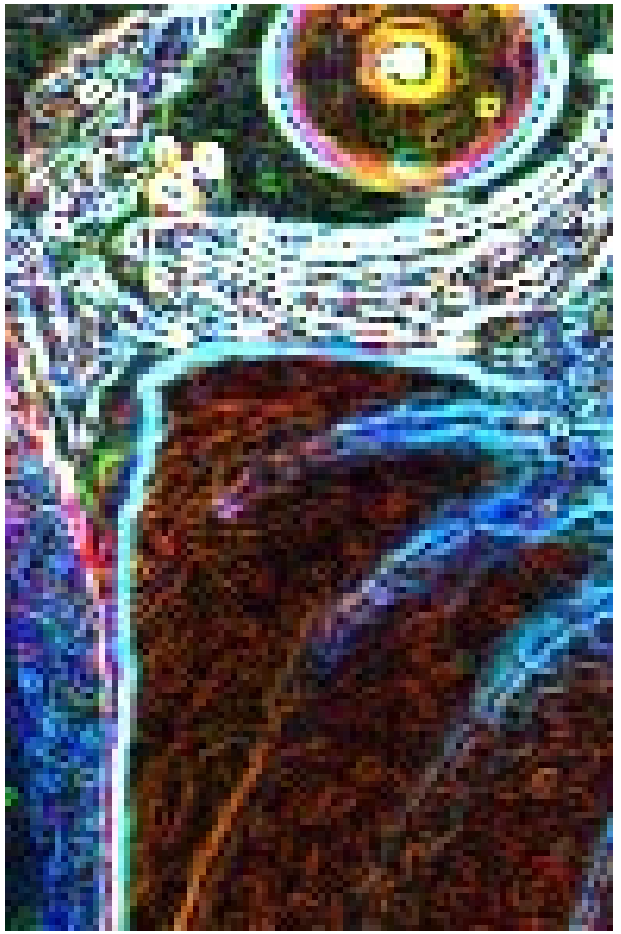
- What about edge detection?

Is there a way to naively apply kernel-based edge detection such as the Sobel kernel set to a color image? What if we just find the Sobel magnitude of each color plane?

Is there a way to naively apply kernel-based edge detection such as the Sobel kernel set to a color image? What if we just find the Sobel magnitude of each color plane?

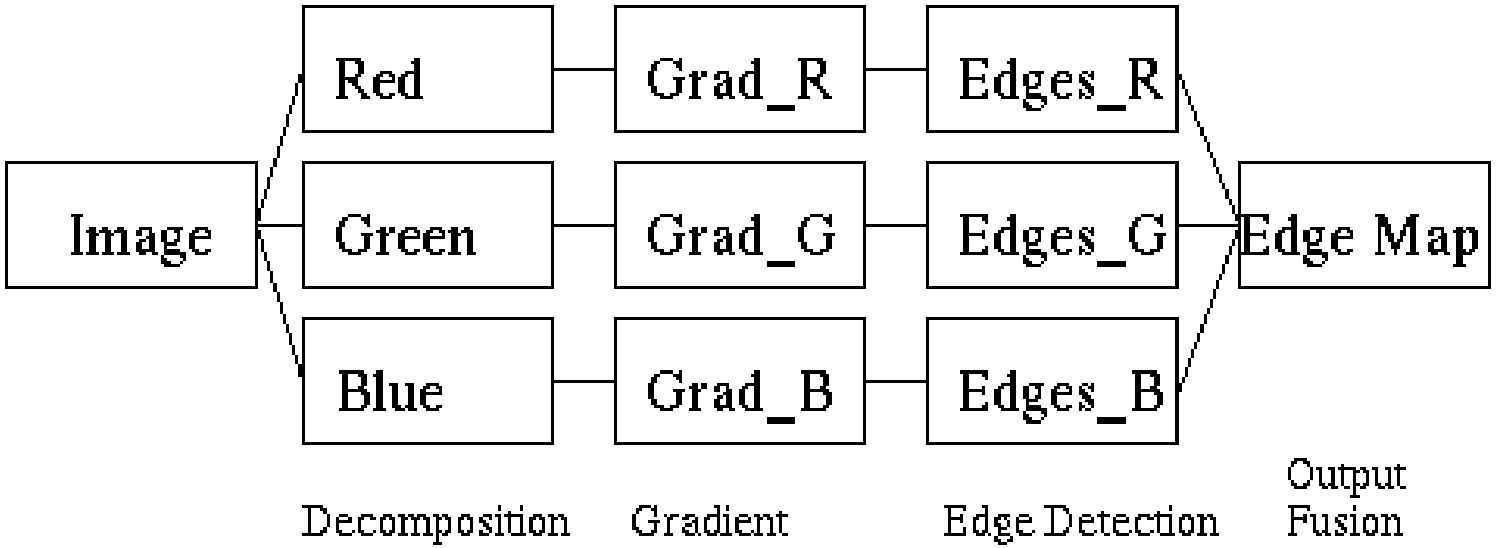


In this strategy for an RGB image, each color plane is edge detected individually – *is this a good solution?*



In this strategy for an RGB image, each color plane is edge detected individually – *is this a good solution?*

The independent application of Sobel kernels is an example of processing each color plane separately



Other color edge detection methods exist – with the goal of not only finding transitions in intensity but in hue and/or saturation

- The *compass operator* considers the collection of color points on each side of a supposed edge and measures the total difference (using a measure called the *Earth Mover distance*)
- A method using the *color opponent* mechanism has been developed recently - http://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Yang_Efficient_Color_Boundary_2013_CVPR_paper.pdf
- Since color points are vectors, we can measure the difference between vectors, which has both magnitude and angles (two of them)
  - The Canny operator has been extended to color images

The opponent color system uses differences in the typical color responses – it's somewhat biologically inspired

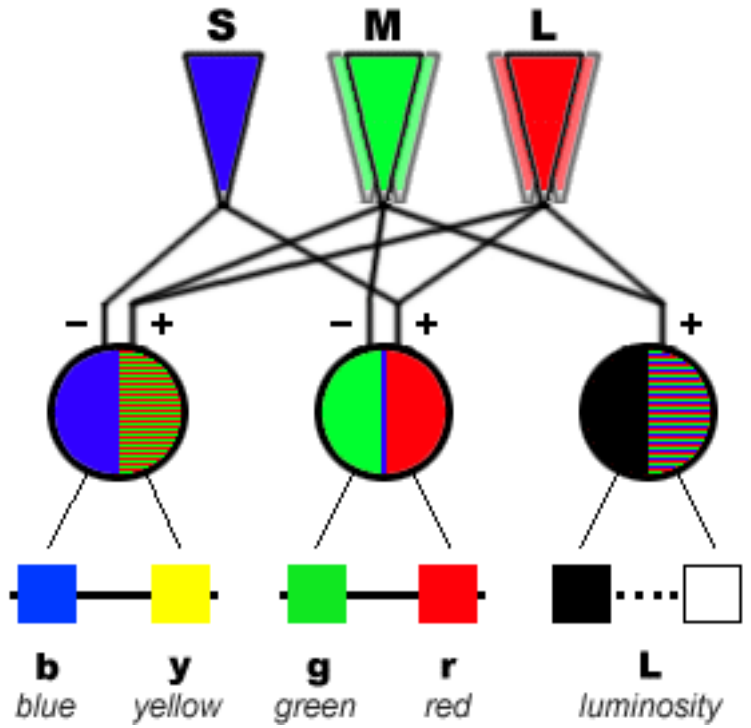
Computer Vision Opponent Color Systems

Convert RGB into opponent-like color system:

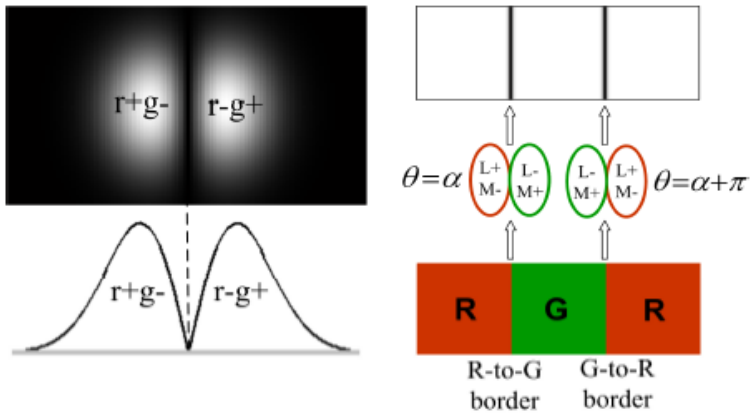
$$\begin{pmatrix} R+G+B \\ R-G \\ R+G-2B \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 1 & -2 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Used in Semantic Pathfinder:

$$\begin{bmatrix} E \\ E_\lambda \\ E_{\lambda\lambda} \end{bmatrix} = \begin{pmatrix} 0.06 & 0.63 & 0.27 \\ 0.3 & 0.04 & -0.35 \\ 0.34 & -0.6 & 0.17 \end{pmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$



- Yang's color opponent system for edge detection
- They call it "boundary detection"
- In the cone layer, the image is separated into Red, Green, Blue and Yellow (average of Red and Green).
- The four are filtered with Gaussian kernels
- The ganglion layer forms weighted differences of two color planes; for example, $S = 0.8G - 0.3B$
- In the cortex layer, these difference planes are processed with "double-opponent" filters that detect differences in color or intensity…



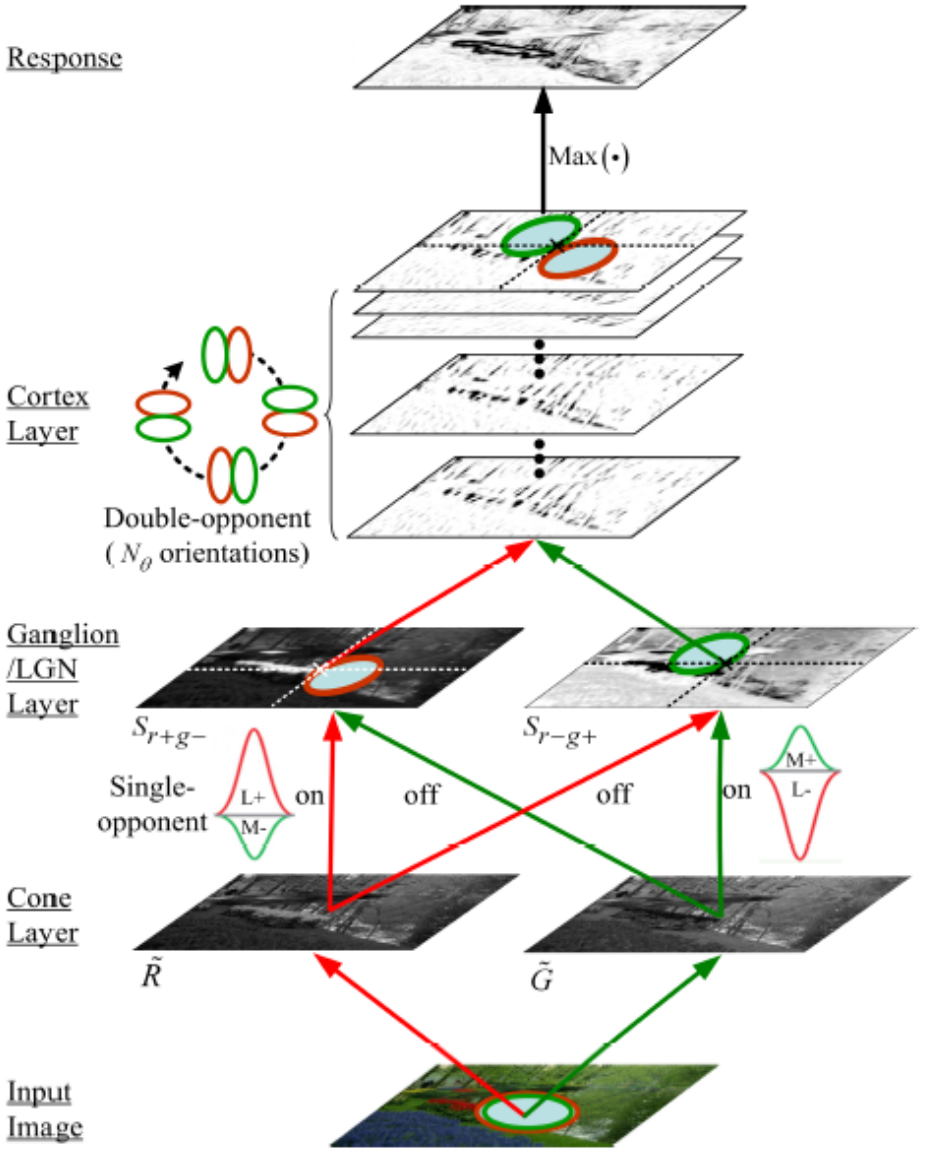
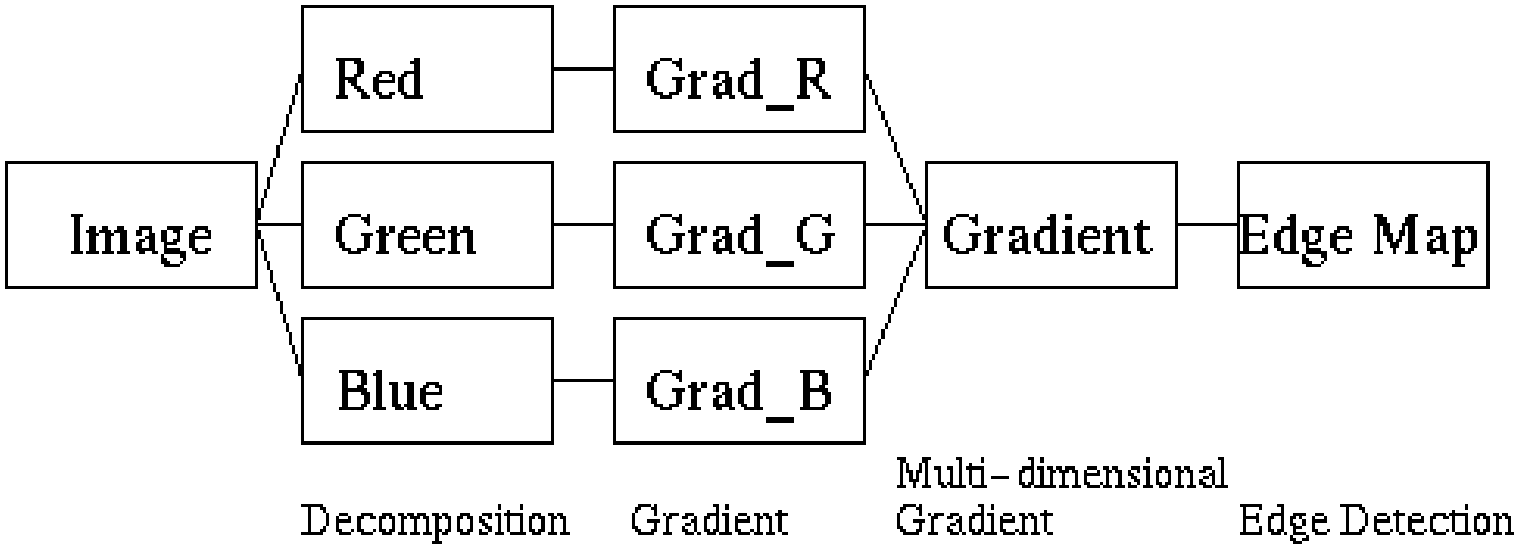


Figure 3. The flowchart of our framework for boundary detection in the R-G channel. The similar computational steps are used in the other channels.

Figure 6. The proposed model responds differently to color and achromatic boundaries with various cone-input weightings. **From left to right:** Original images and the responses of $CO(w)$ with different cone-input weightings (w): -1.0, -0.6, -0.4 and 0.0. Note that the four color blocks in the artificial image of the top row have pure colors, i.e., they are of equi-luminance.

From Yang 2013

Some gradient-based methods operate on the combination of the three color plane gradients



The Canny method has been extended to color images

- Each of the RGB color planes is smoothed independently, using Gaussian kernels
- The color gradient in the dominant direction is found as the largest eigenvalue of the local structure matrix M:

$$M(c,r) = \begin{bmatrix} \boldsymbol{I}_x(x,y) \cdot \boldsymbol{I}_x(x,y) & \boldsymbol{I}_x(x,y) \cdot \boldsymbol{I}_y(x,y) \\ \boldsymbol{I}_x(x,y) \cdot \boldsymbol{I}_y(x,y) & \boldsymbol{I}_y(x,y) \cdot \boldsymbol{I}_y(x,y) \end{bmatrix}$$

- where:

$$\boldsymbol{I}_x(c,r) = \frac{\partial \boldsymbol{I}}{\partial x}(c,r) = \begin{bmatrix} \frac{\partial I_R(c,r)}{\partial x} \\ \frac{\partial I_G(c,r)}{\partial x} \\ \frac{\partial I_B(c,r)}{\partial x} \end{bmatrix} \quad \text{and} \quad \boldsymbol{I}_y(c,r) = \frac{\partial \boldsymbol{I}}{\partial y}(c,r) = \begin{bmatrix} \frac{\partial I_R(c,r)}{\partial y} \\ \frac{\partial I_G(c,r)}{\partial y} \\ \frac{\partial I_B(c,r)}{\partial y} \end{bmatrix}$$

16.4 Other Color Edge Operators

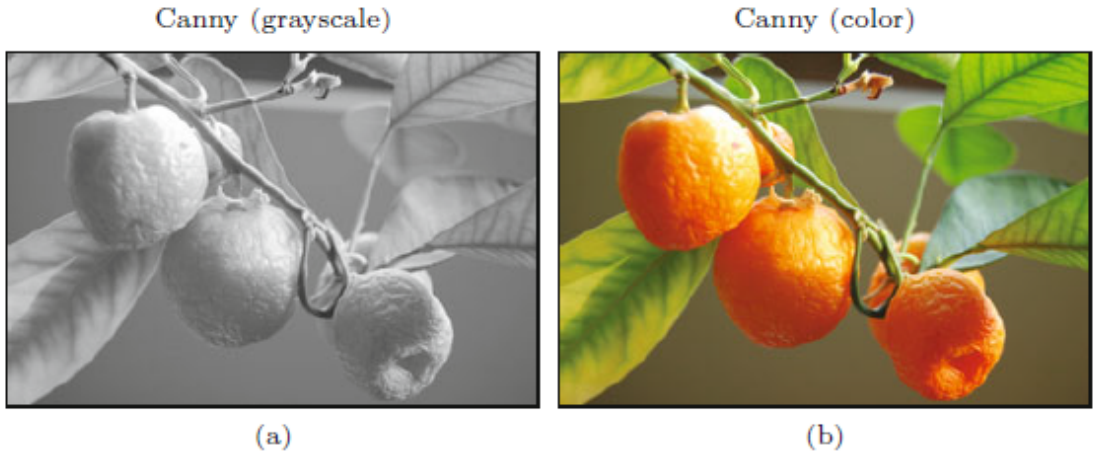


Fig. 16.6
Canny grayscale vs. color version. Results from the grayscale (left) and the color version (right) of the Canny operator for different values of σ ($t_{hl} = 20\%$, $t_{lo} = 5\%$ of max. edge magnitude).

Today's Objectives

Edge detection

- Difference kernels
- Edges and their properties
- Edge detection kernels
- Canny edge detection
- Edge detection in color images