Lecture 6 Edge Detection

COMP3204 Computer Vision

What are edges and how do we find them?

Book
pp
118 -
130

Department of
Electronics and
Computer Science

UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

Content

1. Differentiation/ differencing can be used to find edges of features
2. How can we improve the differencing process?

Edge detection

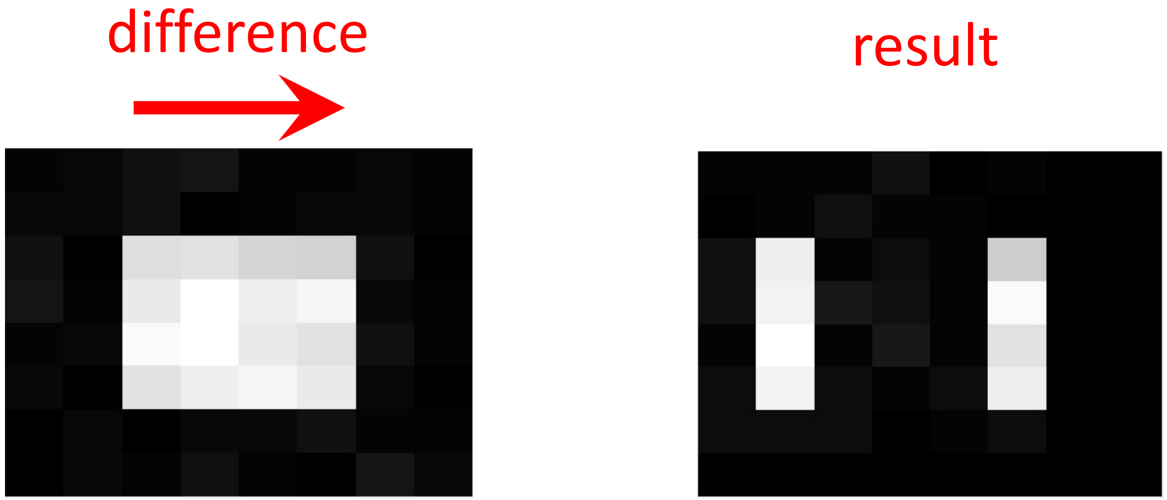
What is an edge? It’s contrast

(a) original image (b) Sobel edge magnitude (c) thresholded magnitude

U2’s Edge can’t detect edges

http://metro.co.uk/2015/05/15/the-edge-falls-off-the-edge-of-the-stage-in-spectacular-style-during-u2s-world-tour-5199503/

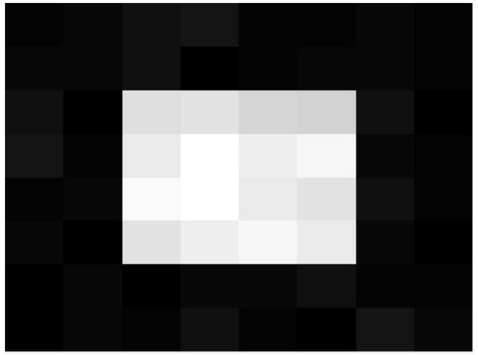
Horizontal differencing



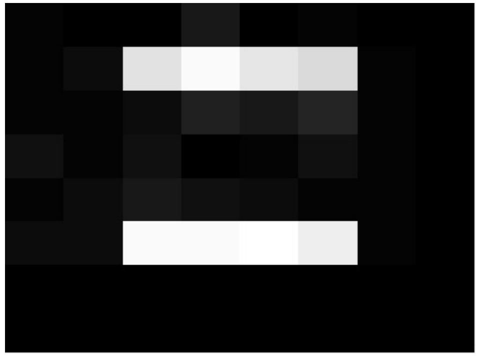
Horizontal differencing detects vertical edges

Vertical differencing

difference

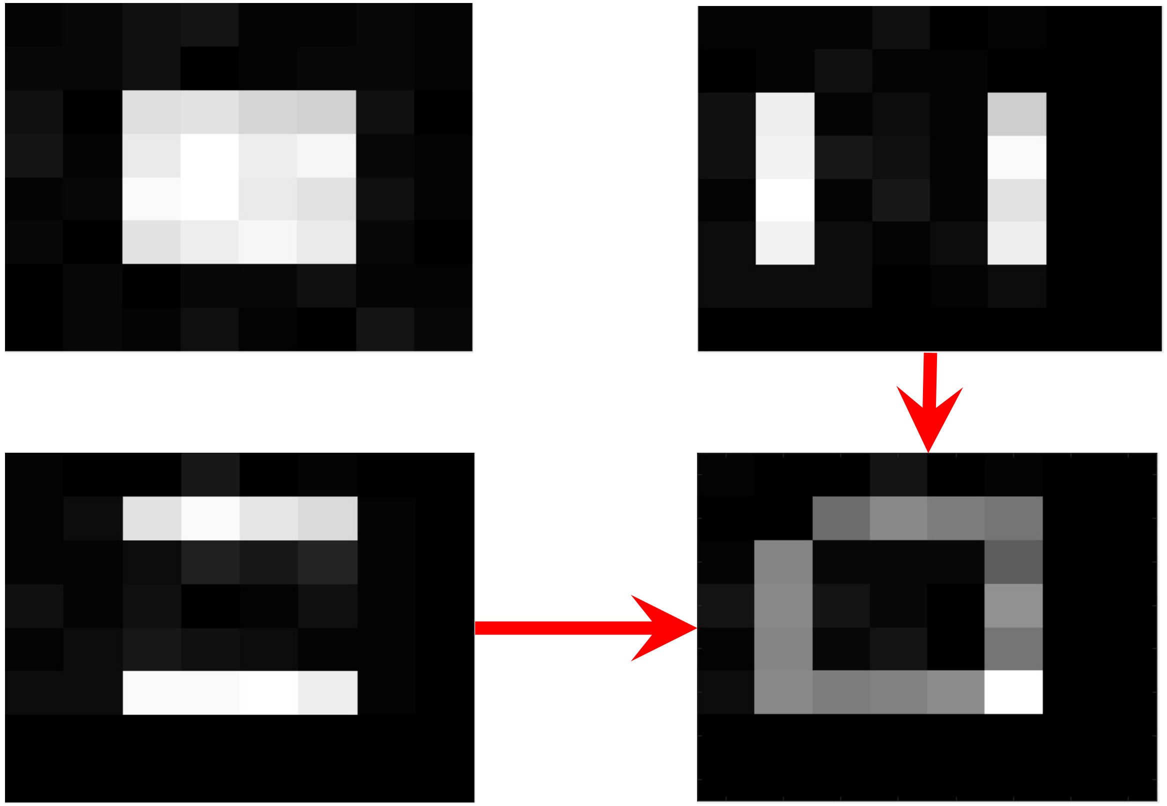


result



Vertical differencing detects horizontal edges

First order edge detection



Addition of horizontal and vertical

First order edge detection

- vertical edges, **Ex** $$\mathbf{Ex}_{x,y} = \left| \mathbf{P}_{x,y} - \mathbf{P}_{x+1,y} \right|$$
- horizontal edges, **Ey** $$\mathbf{Ey}_{x,y} = \left| \mathbf{P}_{x,y} - \mathbf{P}_{x,y+1} \right|$$
- vertical and horizontal edges $$\mathbf{E}_{x,y} = \left| 2 \times \mathbf{P}_{x,y} - \mathbf{P}_{x+1,y} - \mathbf{P}_{x,y+1} \right|$$

First order edge detection

Template

2	-1
-1	0

Code

```
function edge = basic_difference(image)

for x = 1:cols-2 %address all columns except border
  for y = 1:rows-2 %address all rows except border
    edge(y,x)=abs(2*image(y,x)-image(y+1,x)-image(y,x+1)); % Eq. 4.4
  end
end
```

How can we improve it?

Taylor series – evaluate $f(t + \Delta t)$

First approximation, original value

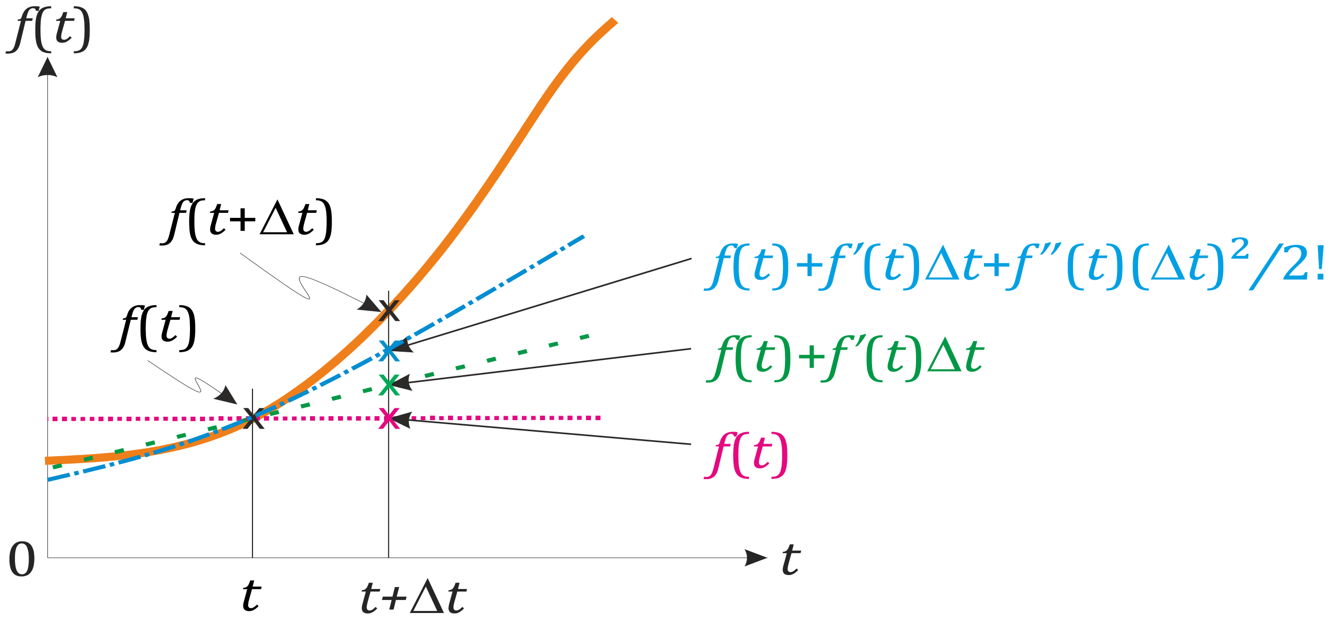
$$f(t + \Delta t) = f(t)$$

Second approximation, add gradient

$$f(t + \Delta t) = f(t) + f'(t)\Delta t$$

Third approximation, add f''

$$f(t + \Delta t) = f(t) + f'(t)\Delta t + \frac{f''(t)}{2!}(\Delta t)^2$$



Taylor series

$$f(t + \Delta t) = f(t) + f'(t)\Delta t + \frac{f''(t)}{2!}(\Delta t)^2 + \frac{f'''(t)}{3!}(\Delta t)^3 + \cdots + \frac{f^n(t)}{n!}(\Delta t)^n$$

Edge detection maths

Taylor expansion for $f(x+\Delta x)$ $$f(x+\Delta x) = f(x) + \Delta x \times f'(x) + \frac{\Delta x^2}{2!} \times f''(x) + O(\Delta x^3) \quad \textbf{A}$$

By rearrangement $$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} - O(\Delta x)$$

This is equivalent to $$\mathbf{Exx}_{x,y} = \left|\mathbf{P}_{x,y} - \mathbf{P}_{x-1,y}\right|$$

1	-1

Expand $f(x-\Delta x)$ $$f(x-\Delta x) = f(x) - \Delta x \times f'(x) + \frac{\Delta x^2}{2!} \times f''(x) - O(\Delta x^3) \quad \textbf{B}$$

A – B $$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} - O(\Delta x^2)$$ $$\mathbf{Exx}_{x,y} = \left|\mathbf{P}_{x+1,y} - \mathbf{P}_{x-1,y}\right|$$

If $\Delta x < 1$, this error is clearly smaller

1	0	-1

Templates for improved first order difference

1	0	-1

(a) Mx

1
0
-1

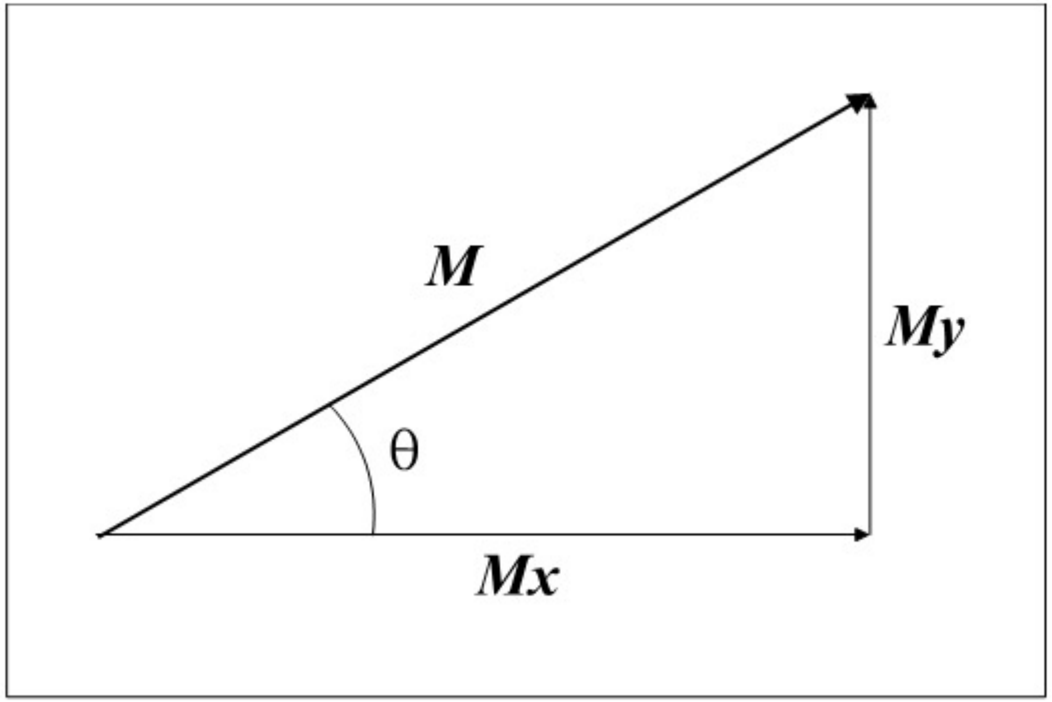
(b) My

Edge Detection in Vector Format

Vectors have magnitude (strength) and direction

$$M = \text{magnitude} = \sqrt{M_x^2 + M_y^2}$$

$$\theta = \text{direction} = tan^{-1}\left(\frac{M_y}{M_x}\right)$$



Templates for 3×3 Prewitt operator

Average improved horizontal and vertical operators over 3 rows/ columns to give Prewitt templates

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

(a) Mx

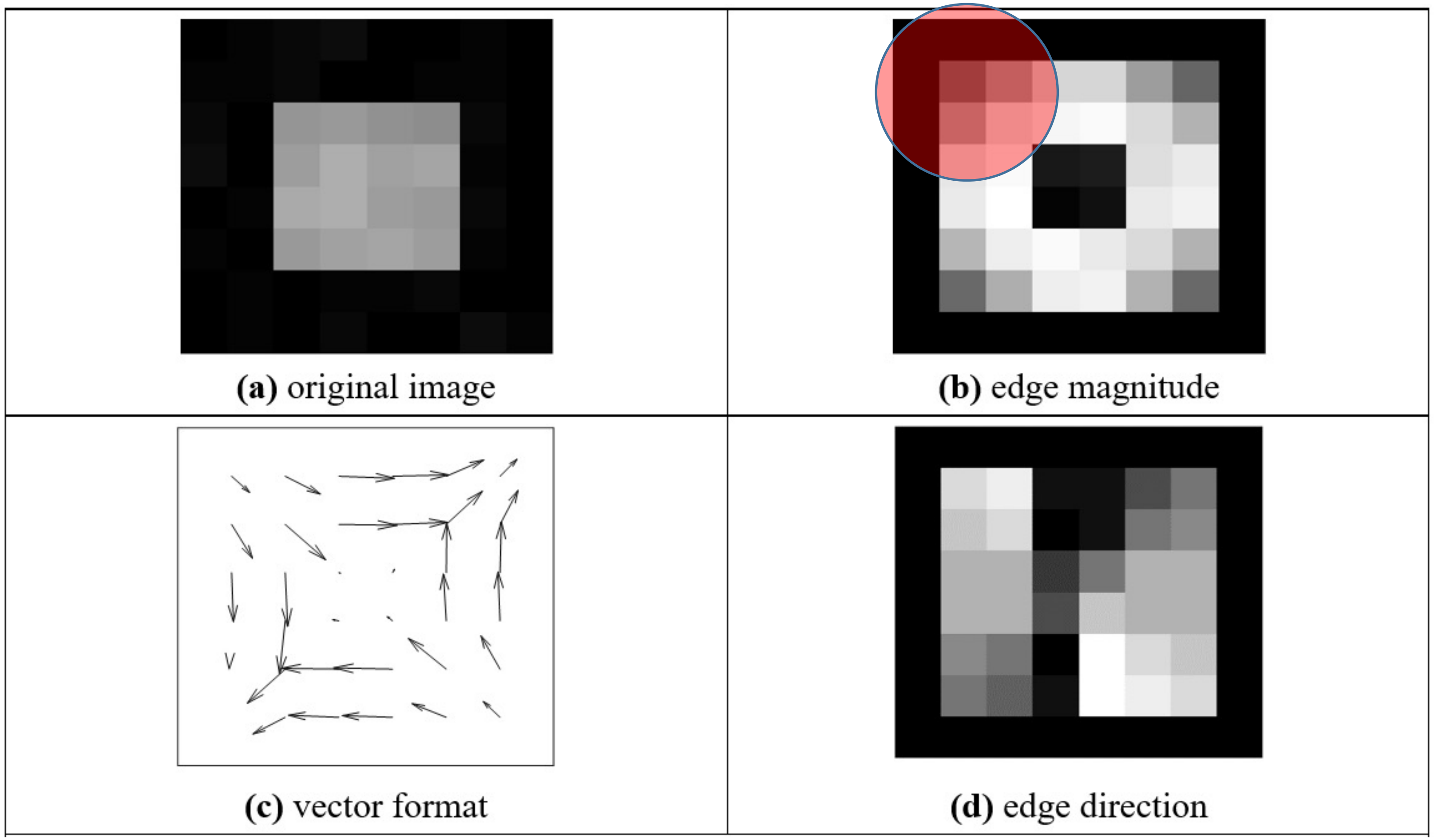
1	1	1
0	0	0
-1	-1	-1

(b) My

Edge magnitude and direction calculated for centre point

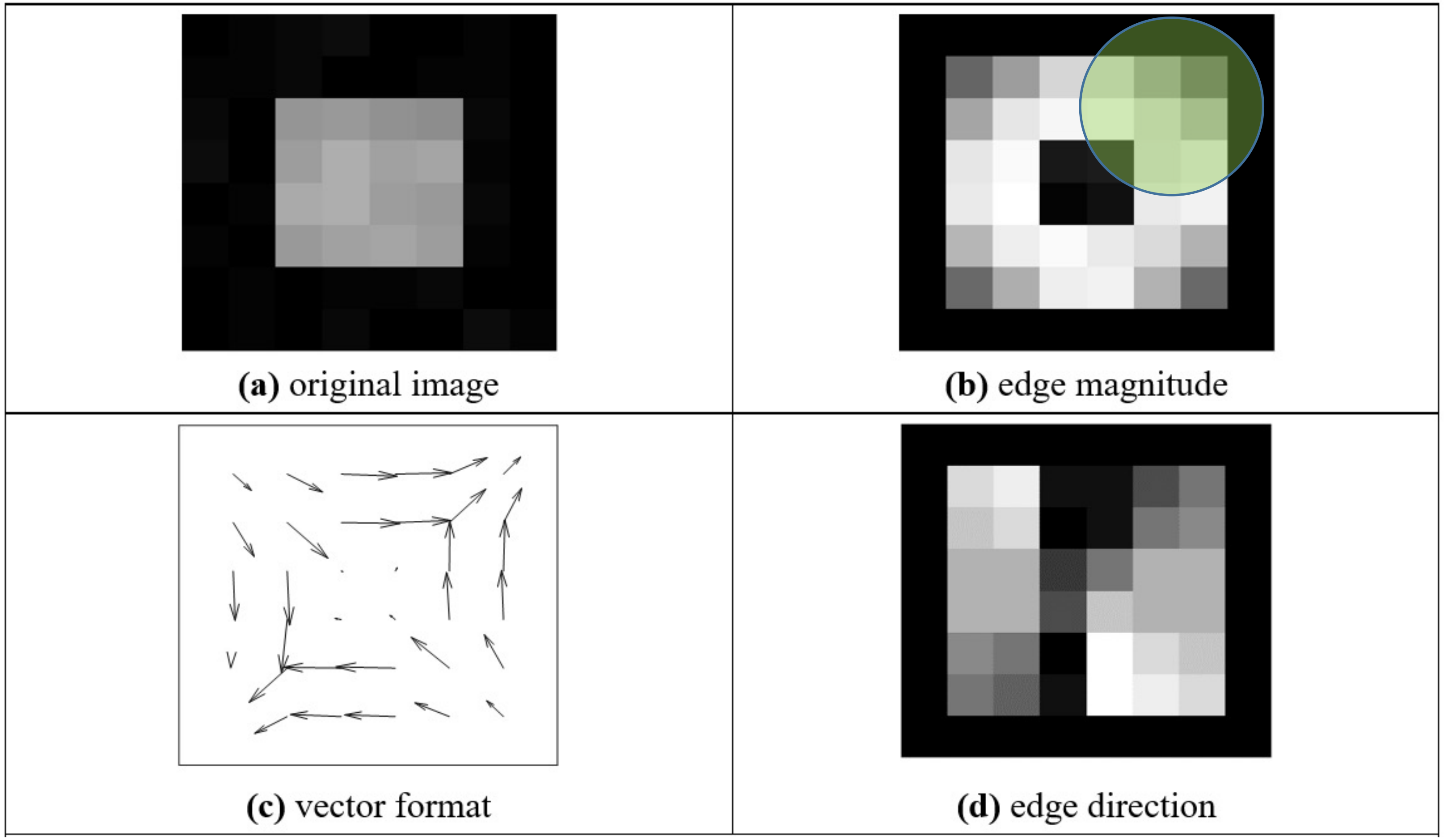
Applying the Prewitt Operator

No missing points



Applying the Prewitt Operator

Blurred edges



(a) original image

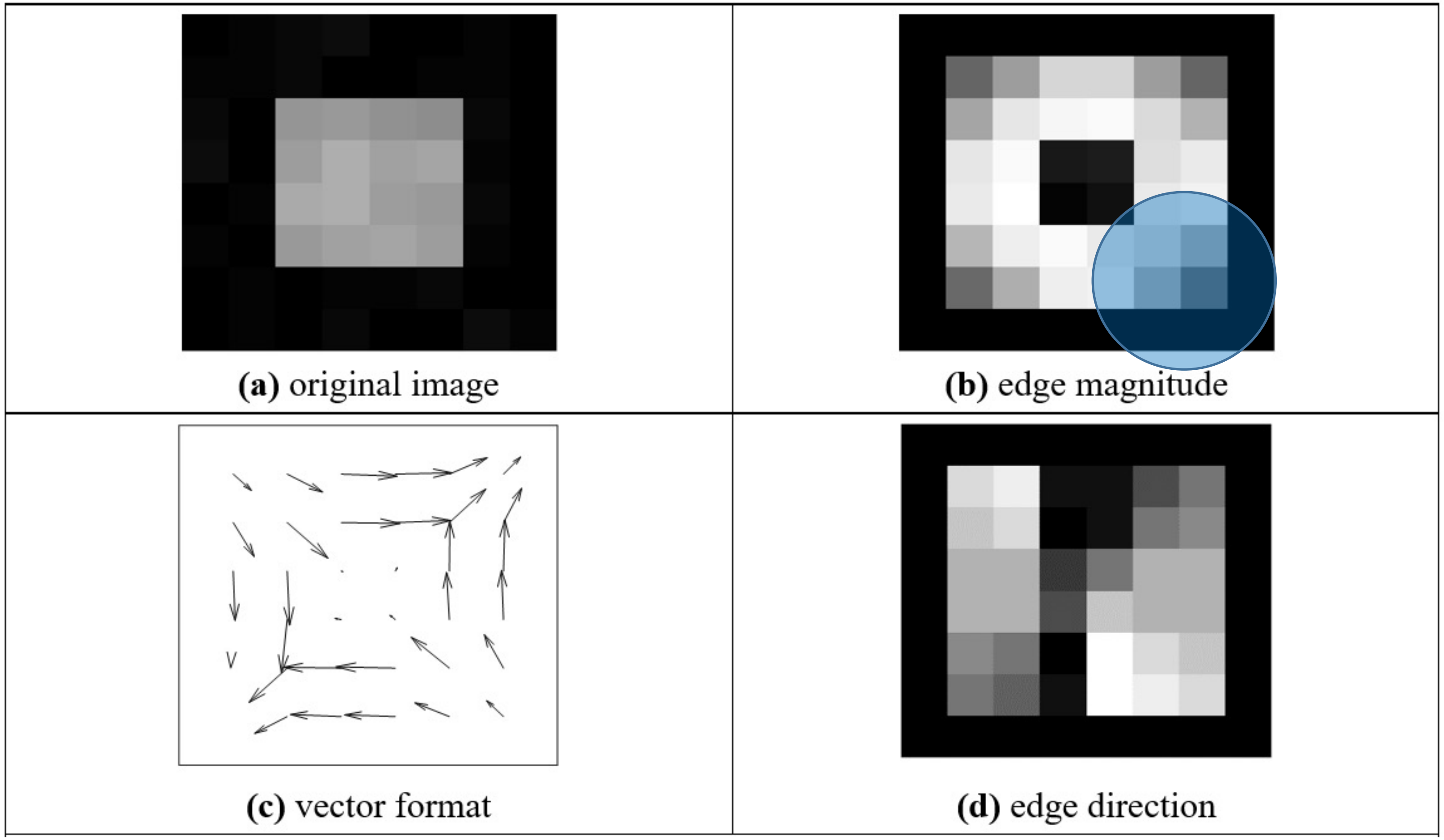
(b) edge magnitude

(c) vector format

(d) edge direction

Applying the Prewitt Operator

No double points



(a) original image

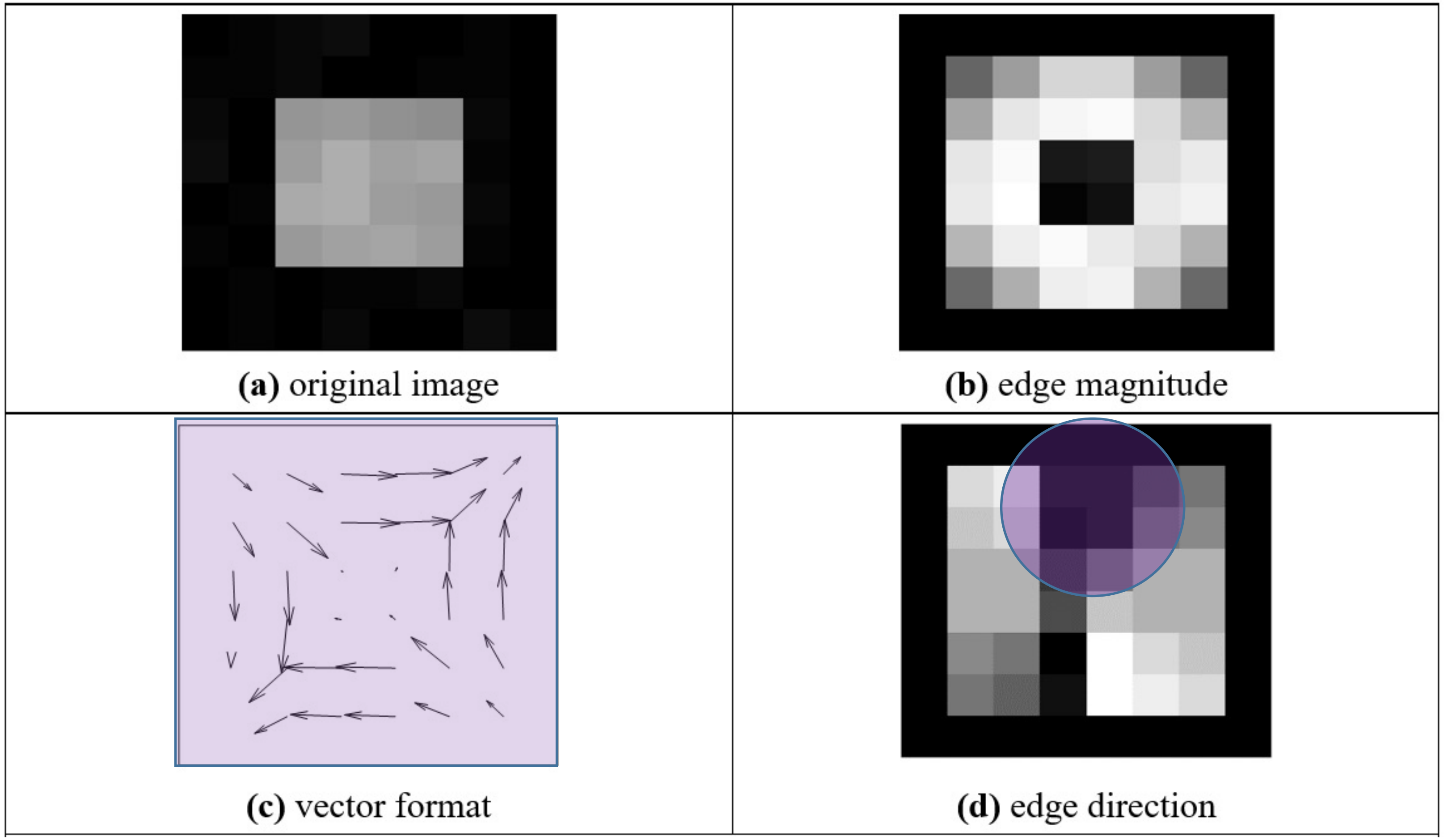
(b) edge magnitude

(c) vector format

(d) edge direction

Applying the Prewitt Operator

Displaying gradients as an image communicates nothing



So use vectors

Templates for Sobel operator

Sobel is most popular basic operator
Double the centre coefficients of Prewitt

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

(a) Mx

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

(b) My

WHY?

Applying Sobel operator

(a) original image (b) Sobel edge magnitude (c) thresholded magnitude

Generalising Sobel - use Pascal’s triangle

1. Averaging

Window size						
2	1	1				
3	1	2	1			Sobel 3×3
4	1	3	3	1		
5	1	4	6	4	1	Sobel 5×5

2. Differencing

Window size						
2	1	-1				
3	1	0	-1			Sobel 3×3
4	1	1	-1	-1		
5	1	2	0	-2	-1	Sobel 5×5

Generalised Sobel

Generated by: `averaging`^{T} `*(differencing)`

```
>> s=Sobel_templates(5)

s(:,:,1) =

     1     2     0    -2    -1

     4     8     0    -8    -4

     6    12     0   -12    -6

     4     8     0    -8    -4

     1     2     0    -2    -1
```

Main points so far

1 – differencing detects contrast and thus edges

2 - can improve the differencing process (by maths!!)

3 – Sobel is a good general purpose operator

We shall go to more sophisticated methods, coming up next.

Filters for edge detection