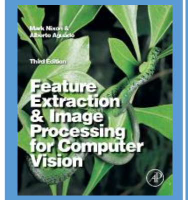
Lecture 5 Group Operators

COMP3204 & COMP6223 Computer Vision

How do we combine points to make a new point in a new image?

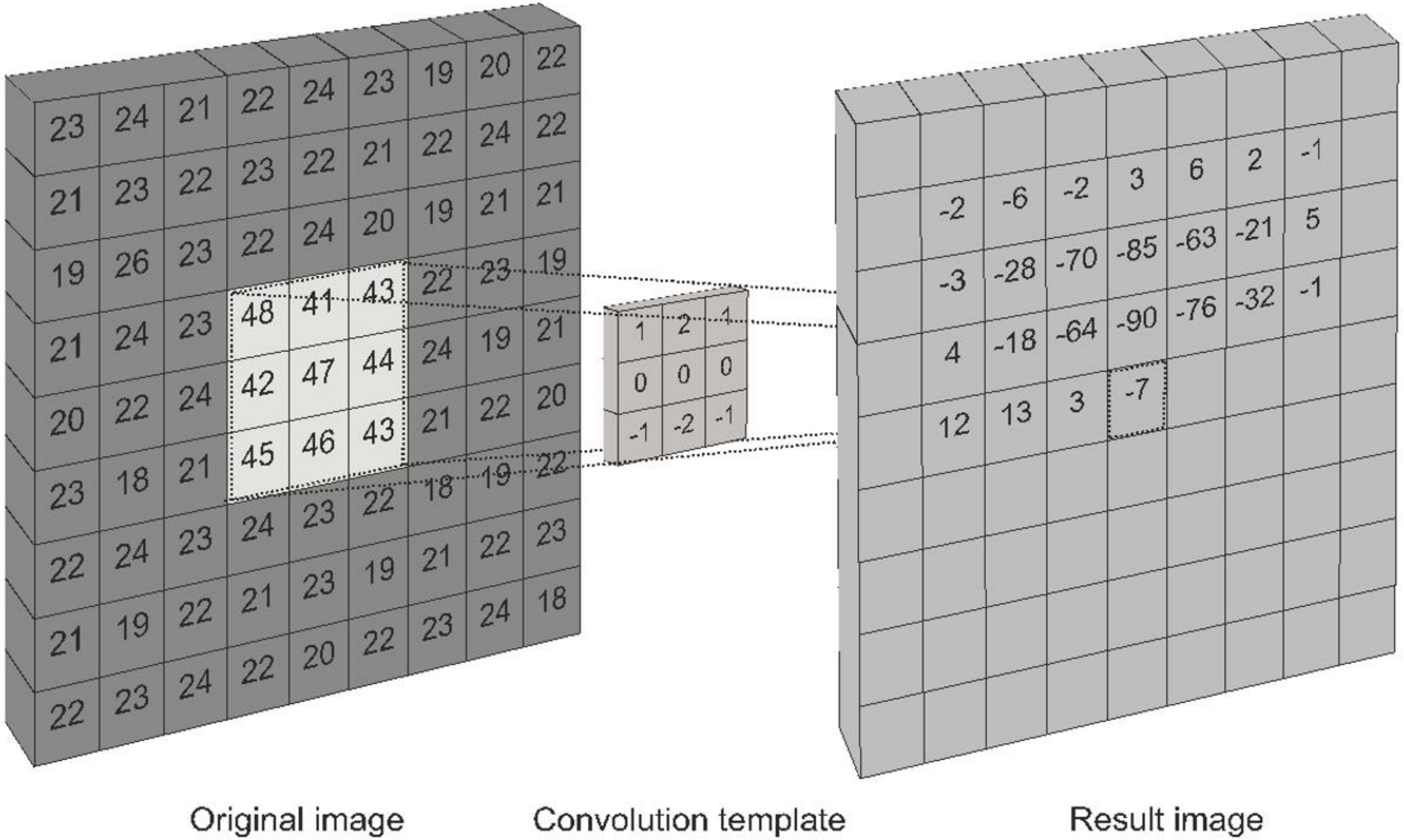


Book pp 98 - 112

Department of Electronics and Computer Science

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Template convolution



Template convolution

Image

100	100	200	200	200
100	100	200	200	200
100	100	200	200	200
200	200	400	400	400
300	300	400	400	400

G_y

0	0	0	0	0
0	400	400	0	0
0	400	400	0	0
0	400	400	0	0
0	0	0	0	0

Result

0	0	0	0	0
0	400	400	0	0
0	640	806	800	0
0	894	894	800	0
0	0	0	0	0

G_x

0	0	0	0	0
0	0	0	0	0
0	500	700	800	0
0	800	800	800	0
0	0	0	0	0

3×3 template and weighting coefficients

w_0	w_1	w_2
w_3	w_4	w_5
w_6	w_7	w_8

$$\mathbf{N}_{x,y} = \sum_{i \in \text{template}} \sum_{j \in \text{template}} w_{i,j} \times \mathbf{O}_{x(i),y(j)}$$

where $w_{i,j}$ are the weights and $x(i), y(j)$ denote the position of the point that matches the weighting coefficient position

3×3 averaging operator

$$\mathbf{N}_{x,y} = \frac{1}{9}\sum_{i\in 3}\sum_{j\in 3}\mathbf{O}_{x(i),y(j)}$$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Illustrating the effect of window size

3×3

5×5

7×7

Template convolution via the Fourier transform

Allows for fast computation for template size ≥ 7×7

$$\mathbf{P} * \mathbf{T} = \mathfrak{I}^{-1}\left(\mathfrak{I}(\mathbf{P}) .\times \mathfrak{I}(\mathbf{T})\right)$$

Template convolution $*$

Fourier transform of the picture, $\mathfrak{I}(\mathbf{P})$

Fourier transform of the template, $\mathfrak{I}(\mathbf{T})$

Point by point multiplication $(.\times)$

Beware of clowns … Oxford

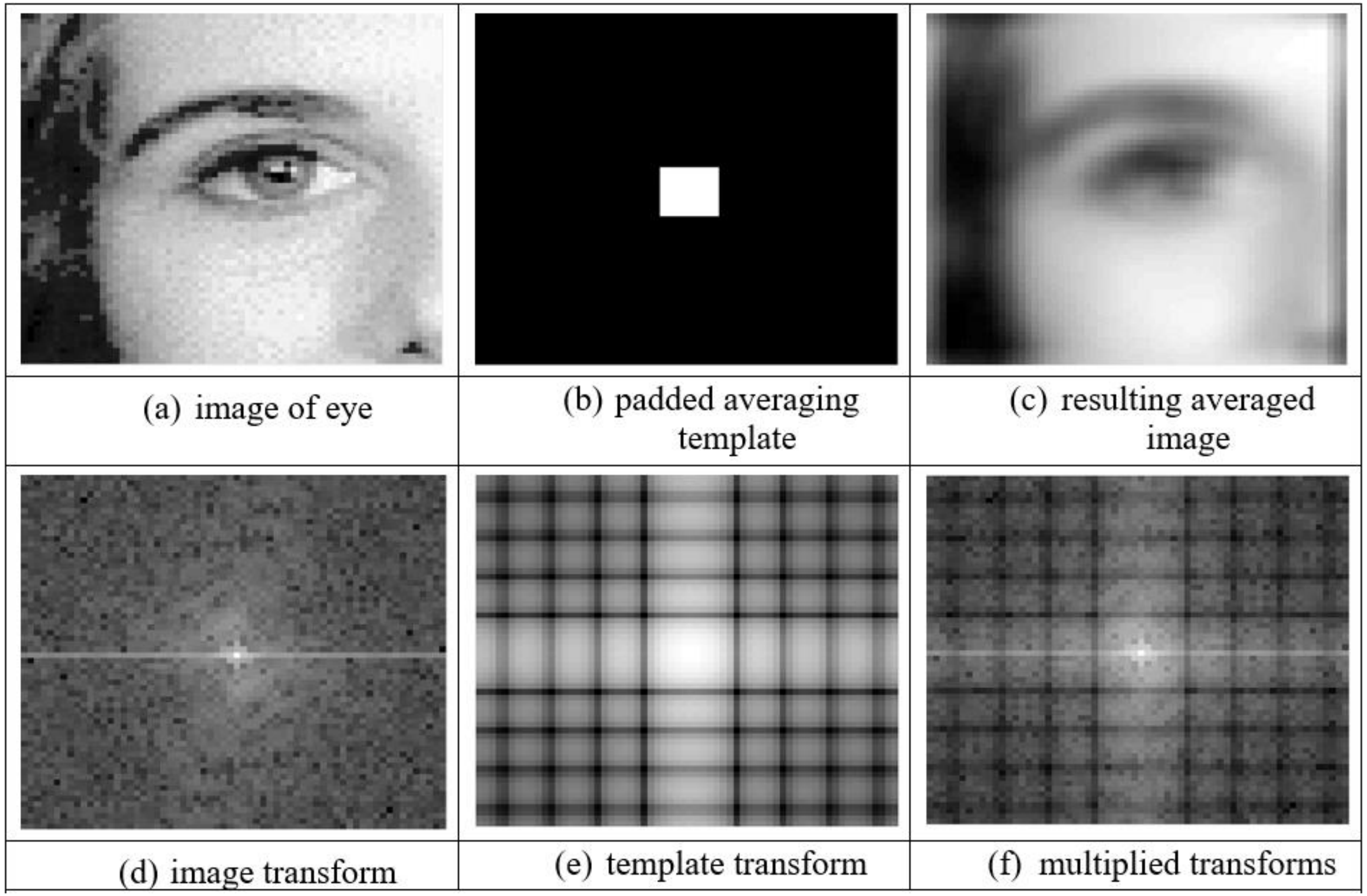
Imperial

$$f(x,y) * h(x,y) \quad \Leftrightarrow \quad F(u,v)H(u,v)$$

$$w(t) = u(t) * v(t) \quad \Leftrightarrow \quad W(f) = U(f)V(f)$$

it's point by point!!

Template Convolution via the Fourier Transform



(a) image of eye

(b) padded averaging template

(c) resulting averaged image

(d) image transform

(e) template transform

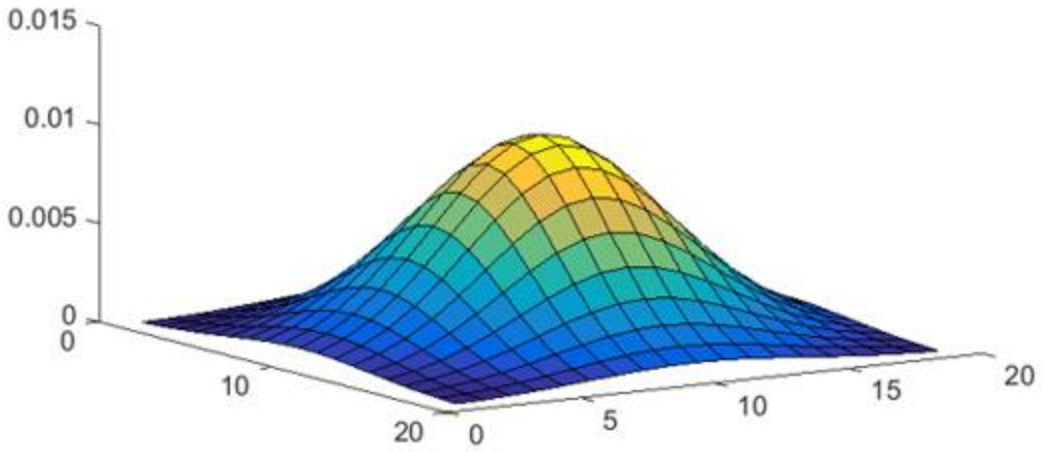
(f) multiplied transforms

2D Gaussian function

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

- Used to calculate template values
- Note compromise between variance σ^2 and window size
- Common choices 5×5, 1.0; 7×7, 1.2; 9×9, 1.4

2D Gaussian function and template



0.002	0.013	0.022	0.013	0. 002
0.013	0.060	0. 098	0. 060	0.013
0.022	0. 098	0.162	0. 098	0.022
0.013	0. 060	0.098	0. 060	0.013
0. 002	0.013	0.022	0.013	0. 002

Template for the 5×5 Gaussian Averaging Operator ($\sigma = 1.0$)

Applying Gaussian averaging

(a) 3×3 (b) 5×5 (c) 7×7

Finding the median from a 3×3 template

2	8	7
4	0	6
3	5	7

(a) 3×3 template

2	4	3	8	0	5	7	6	7

(b) unsorted vector

0	2	3	4	5	6	7	7	8

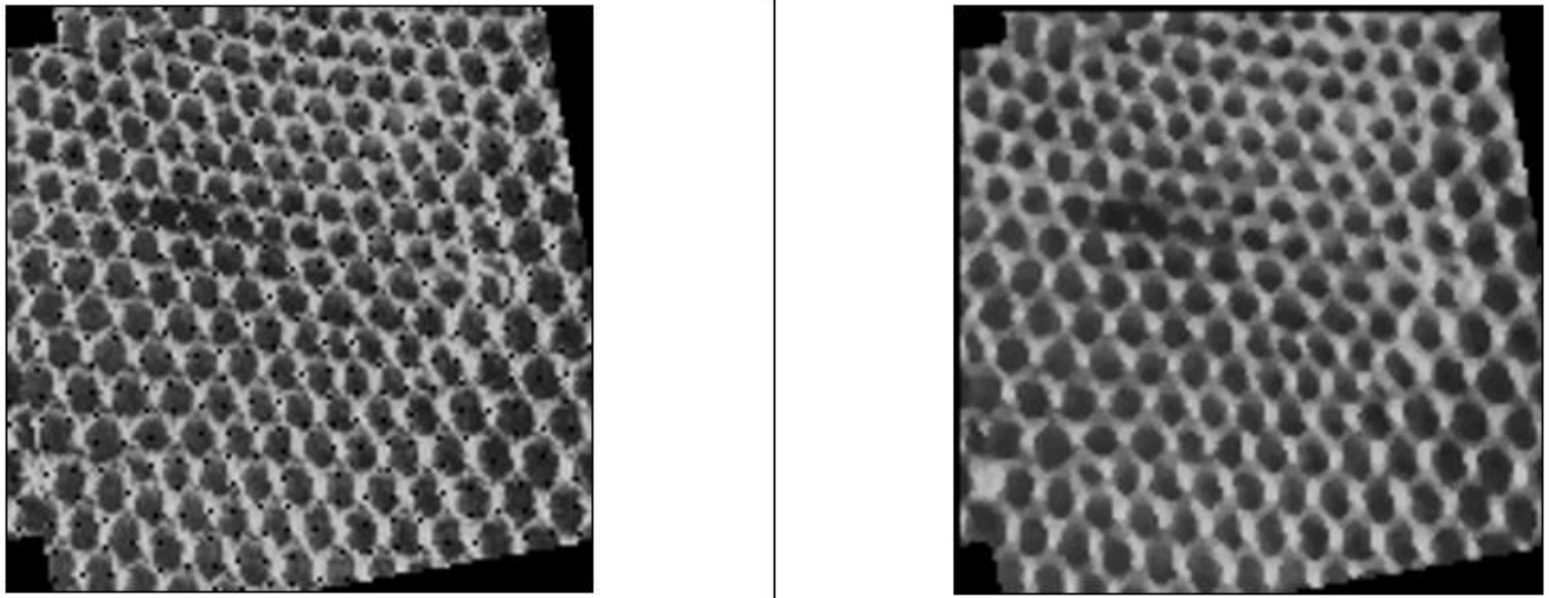
↑ median

(c) sorted vector, giving median

Finding the median from a 3×3 template

Preserves edges

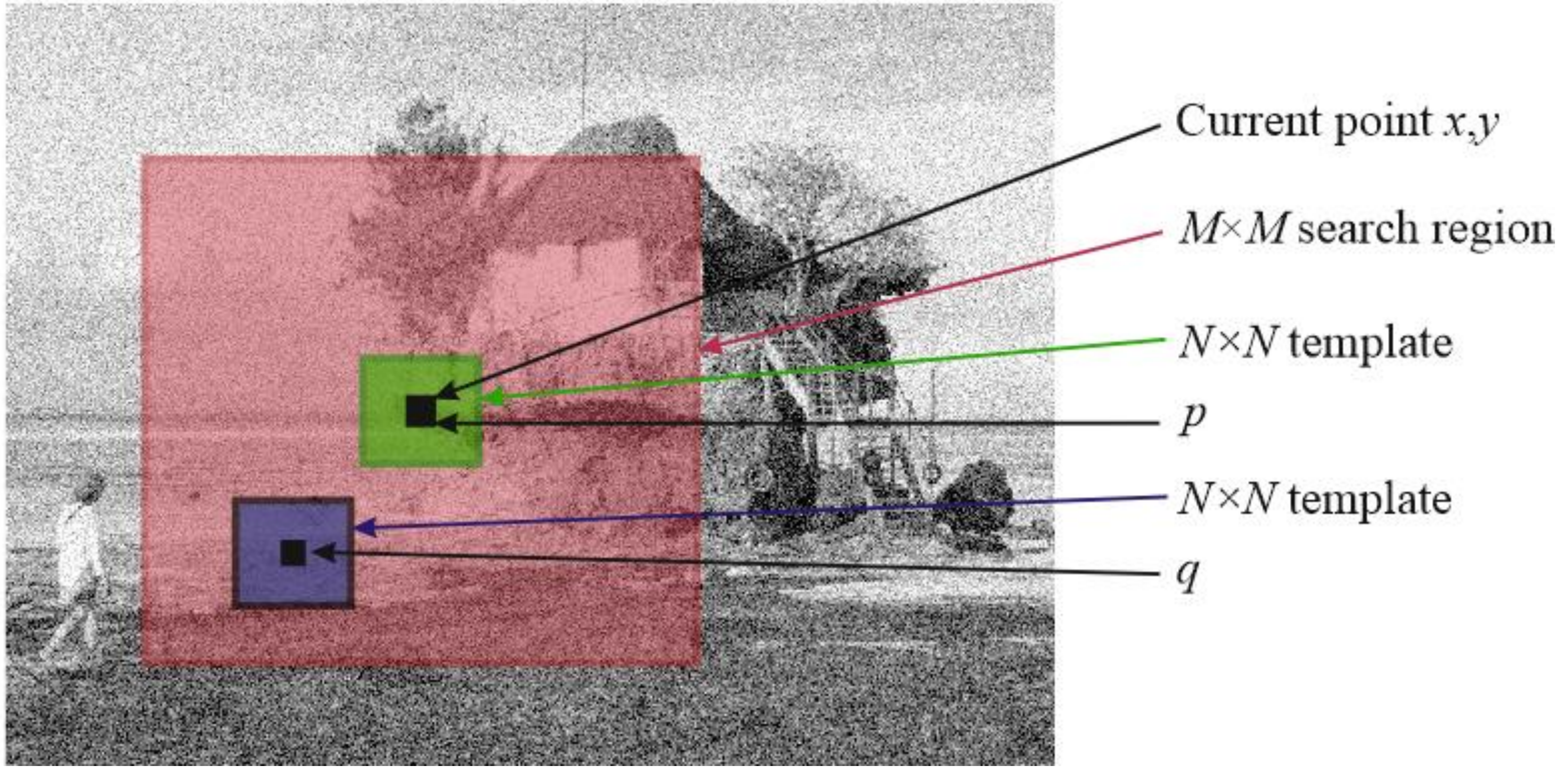
Removes salt and pepper noise



(a) rotated fence (b) median filtered

Newer stuff: non local means

Averaging which preserves regions



Applying non local means

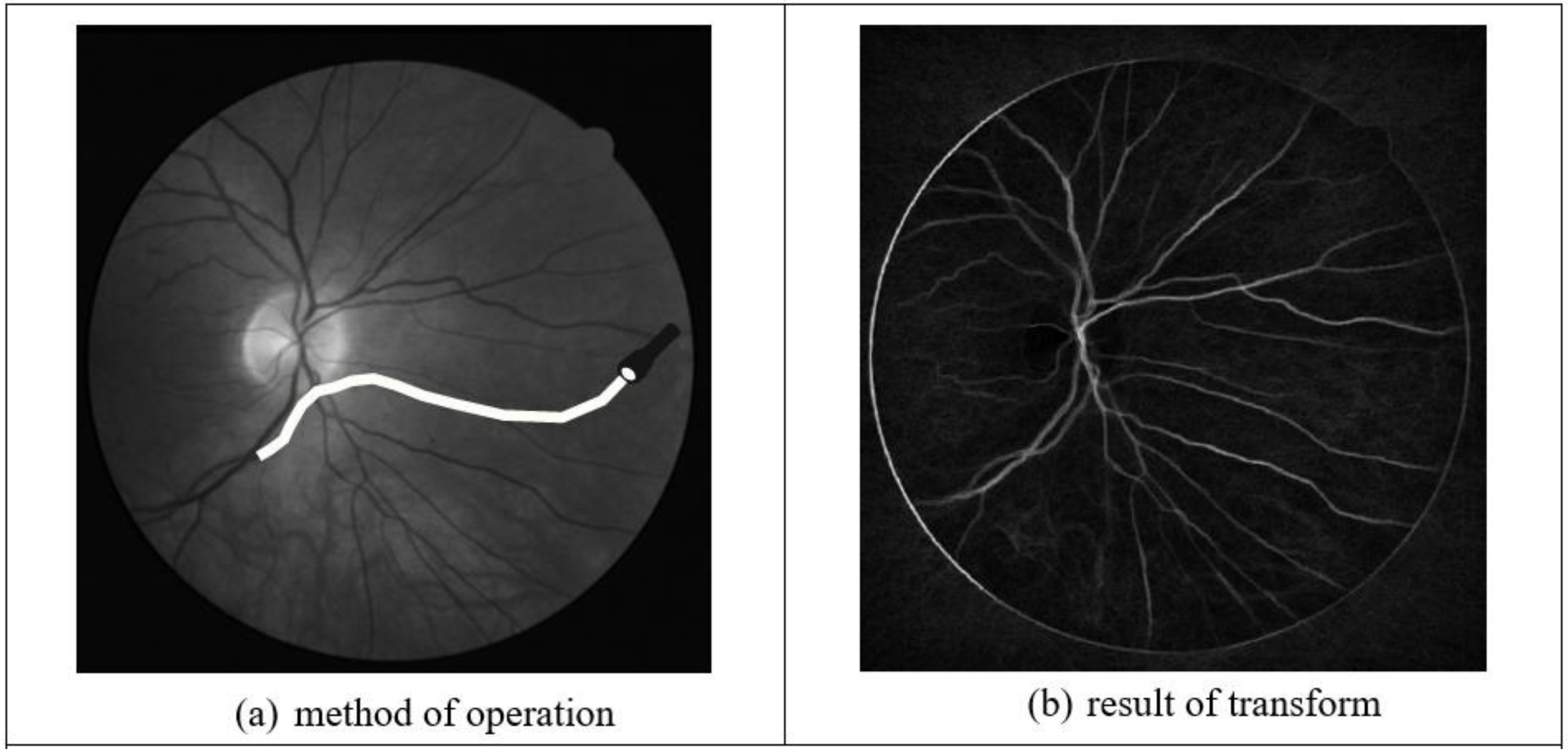
(a) original image

(b) Gaussian averaging

(c) nonlocal means

Even newer stuff: Image Ray Transform

Use analogy to light to find shapes, removing remainder



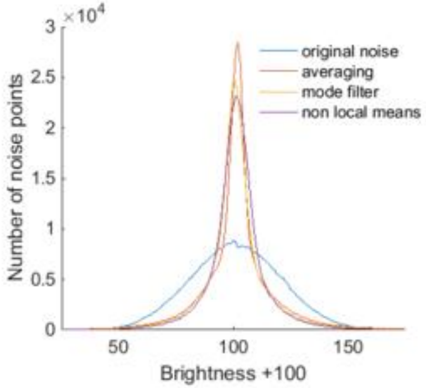
(a) method of operation (b) result of transform

Cummings, Nixon and Prugel-Bennett, *PRL* 2012

Applying Image Ray Transform

(a) tubular structure (b) circular structure (c) car

Good results Poor result

		
(a) Original	(b) (a) with added Gaussian noise	(c) Averaged
		
(d) Gaussian smoothed	(e) Median	(f) Truncated Median
		
(g) Anisotropic diffusion	(h) Non-local-means	(i) Effect of filtering on noise

Comparison of Filtering Operators