INFO0948
Image Processing

Renaud Detry

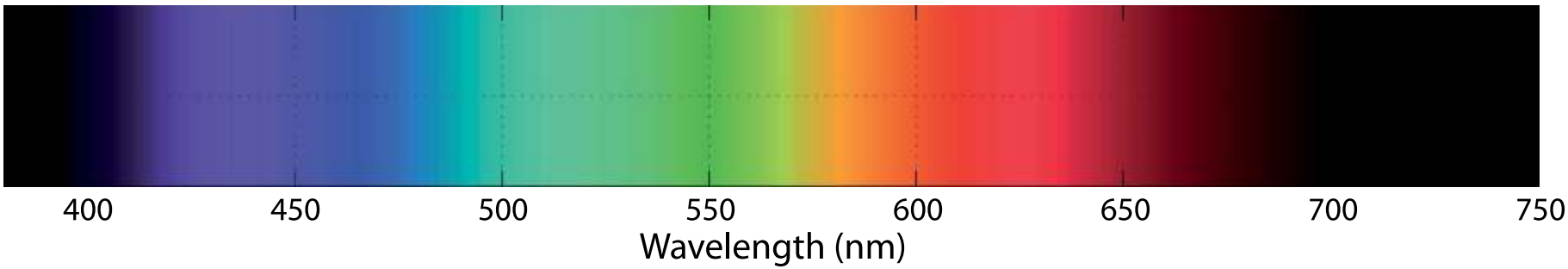
University of Liège, Belgium

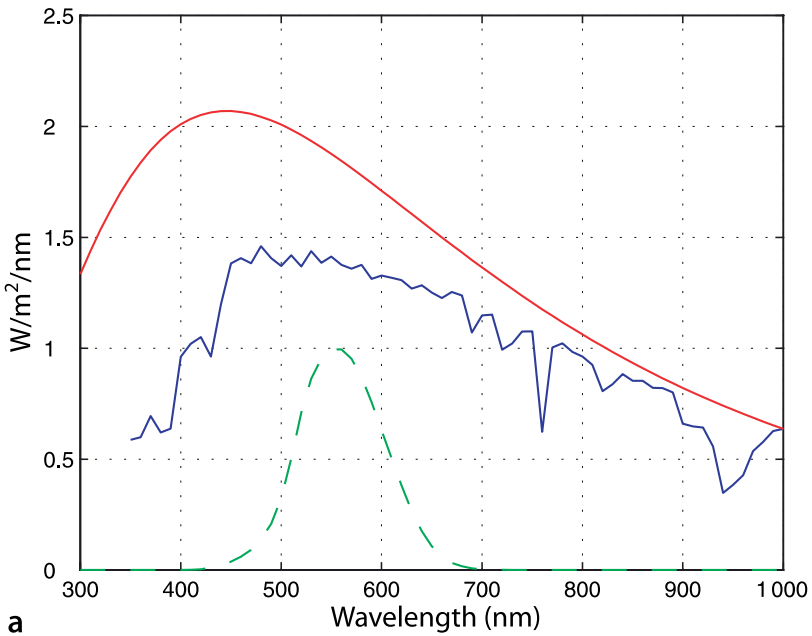
Updated November 20, 2013

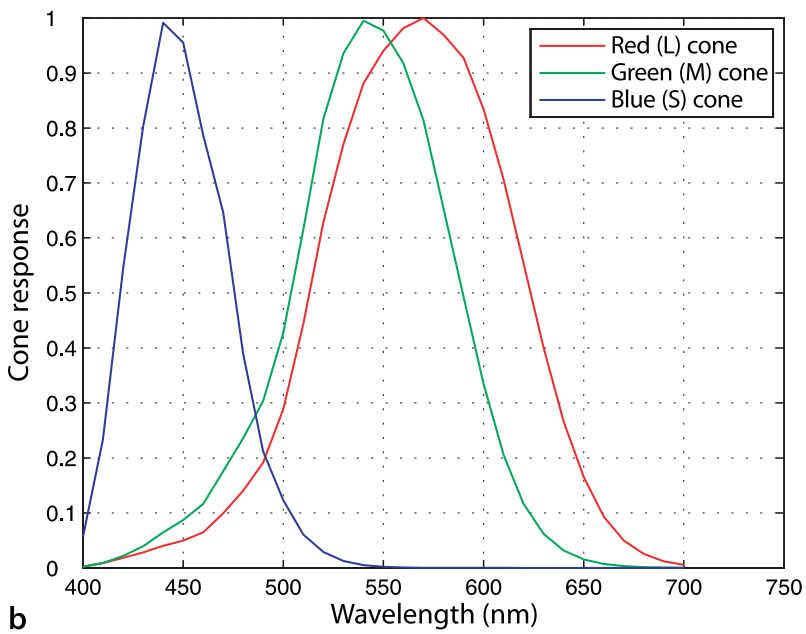
These slides are based on Chapter 12 of the book *Robotics, Vision and Control: Fundamental Algorithms in MATLAB* by Peter Corke, published by Springer in 2011.

Plan

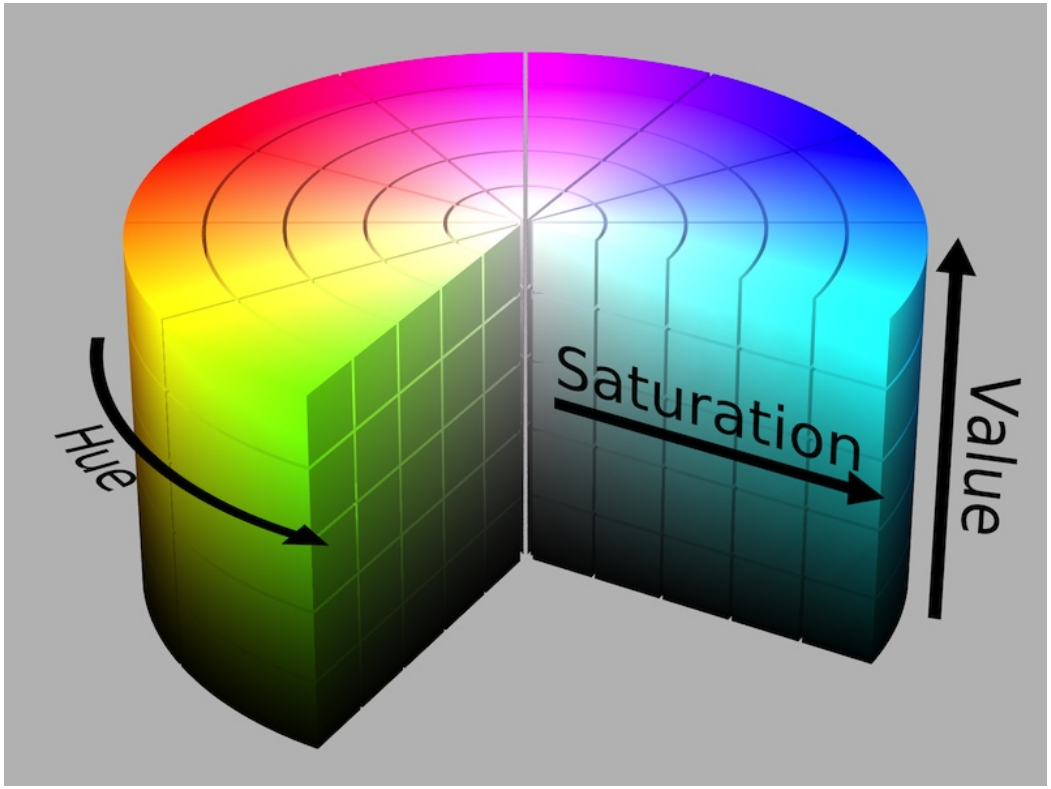
The Spectral Representation of Light, Color, and RGB







Hue-Saturation-Value: an Intuitive Represenation of RGB



Hue is more robust to common changes of lighting conditions than saturation or value.

Images: Computer Representation

A color image is a 2D array of pixels.

Each pixel encodes a color, typically with a triplet (R, G, B).

A color pixel is usually encoded into 24 bits (8 bits per color).

As a result, the R, G, and B components take values between 0 and 255.

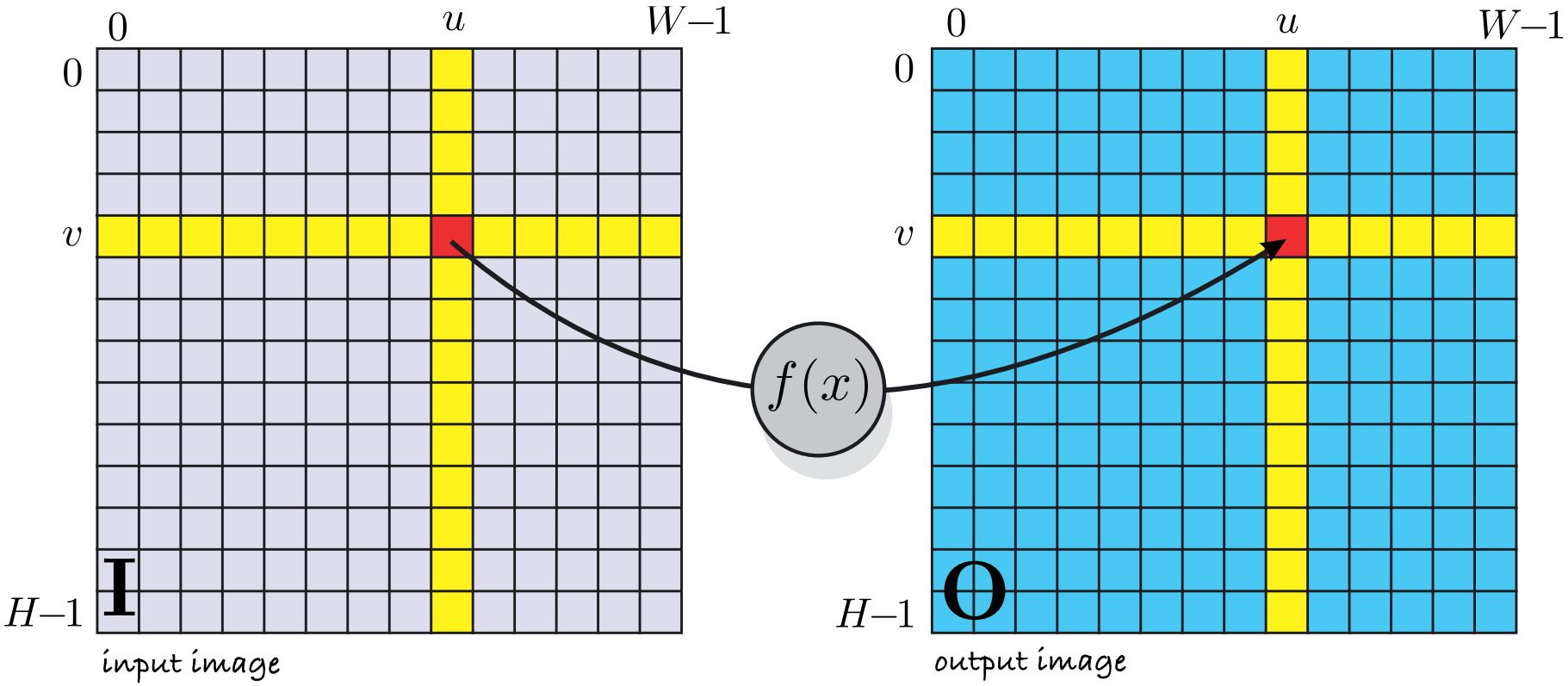
In image processing, it is not uncommon to represent colors with floating-point variables. In this case, R, G, and B usually scale between 0 and 1. Be careful that when converting back to 24-bit color, or writing to disk, the components must be scaled back to 0–255.

In Matlab, the commands `iread` and `idisp` read images from disk and display them.

Plan

Plan

Monadic Image Operations



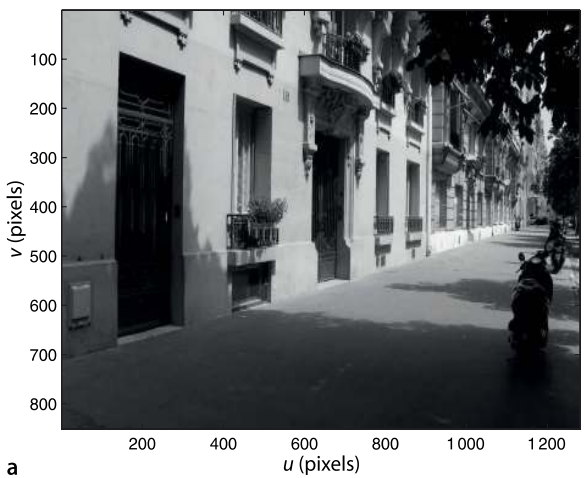
$$\boldsymbol{O}[u,v] = f\big(\boldsymbol{I}[u,v]\big), \quad \forall (u,v) \in \boldsymbol{I}$$

Simple Monadic Operations in Matlab

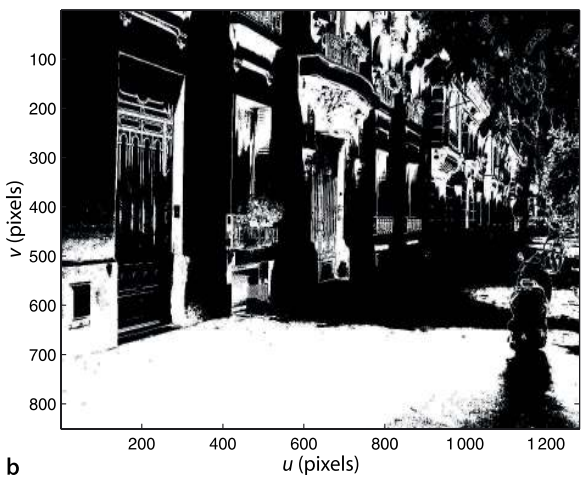
```
>> imd = idouble(im);
>> im = iint(imd);
>> grey = imono(im);
>> color = icolor(grey);
>> color = icolor(grey, [1 0 0]);
>> color = flipdim(im,3);
```

Intensity Histograms

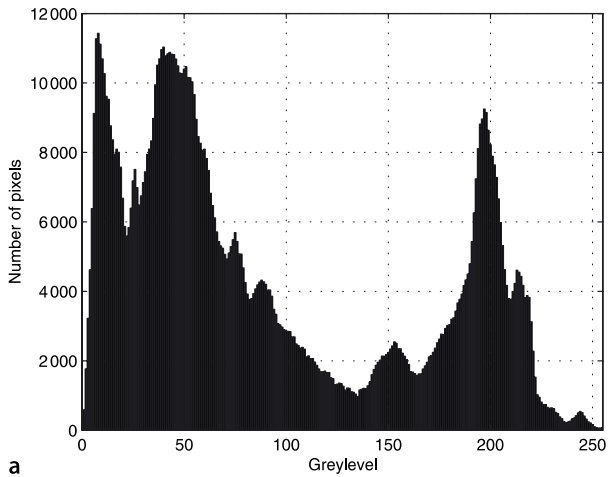
```
>> street = iread('street.png');
>> ihist(street);
>> shadows = (street >= 30) & (street<= 80);
```



a



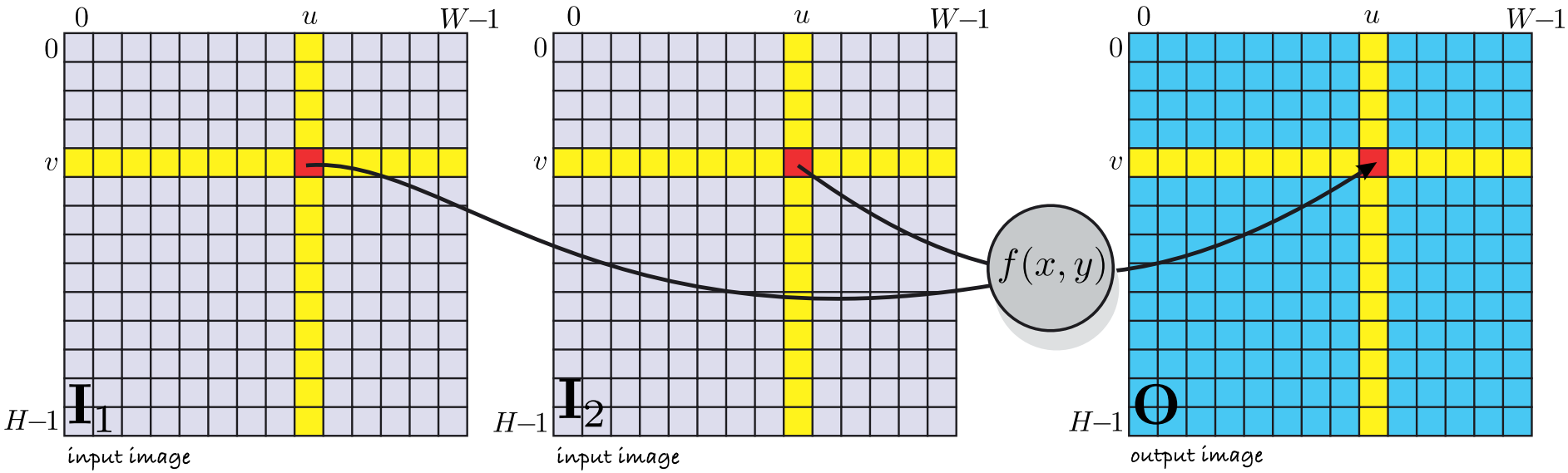
b



a

Plan

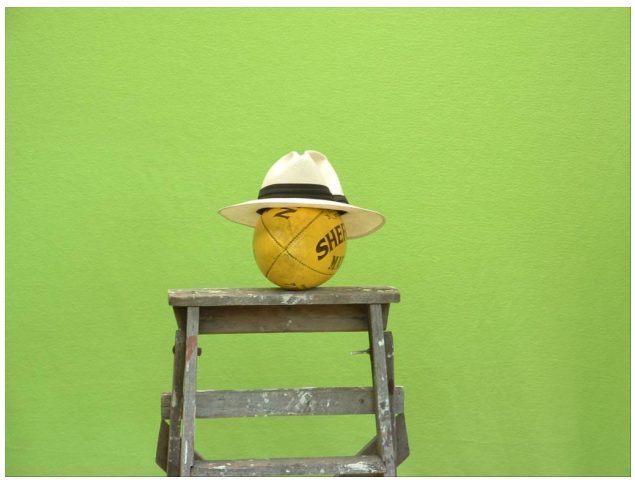
Diadic Operations

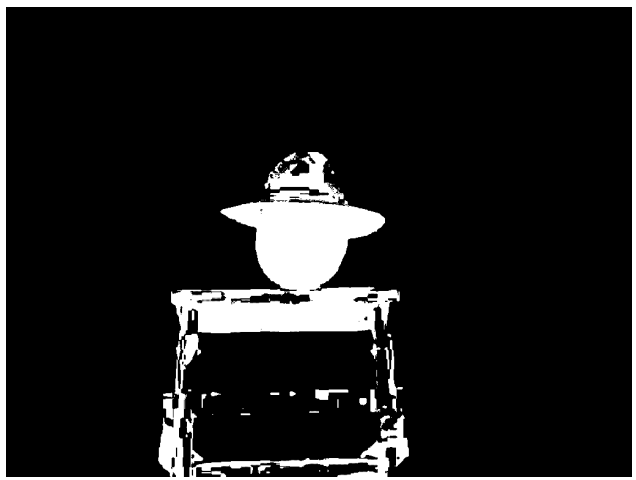


$$\boldsymbol{O}[u,v] = f\big(\boldsymbol{I}_1[u,v], \boldsymbol{I}_2[u,v]\big), \quad \forall (u,v) \in \boldsymbol{I}_1$$

Chroma-keying

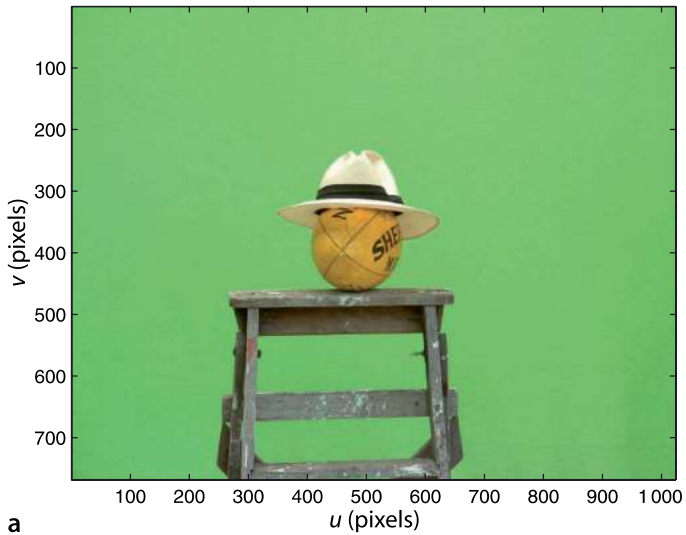
```
>> subject = iread('greenscreen.jpg', 'double');
>> r = subject(:,:,1);
>> g = subject(:,:,2);
>> b = subject(:,:,3);
>> mask = (g < r) | (g < b);
>> mask3 = icolor( idouble(mask) );
>> bg = isamesize(iread('road.png', 'double'), subject);
>> idisp( subject.*mask3 + bg.*(1-mask3) );
```



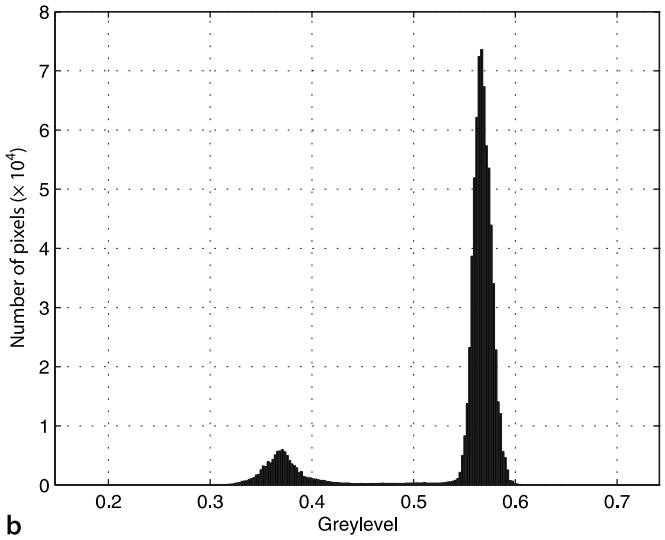


Chroma-keying

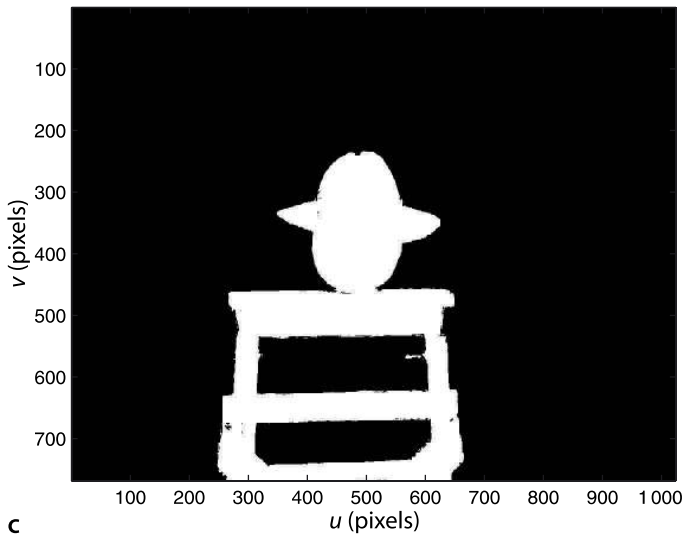
With a more sophisticated keying (see book)



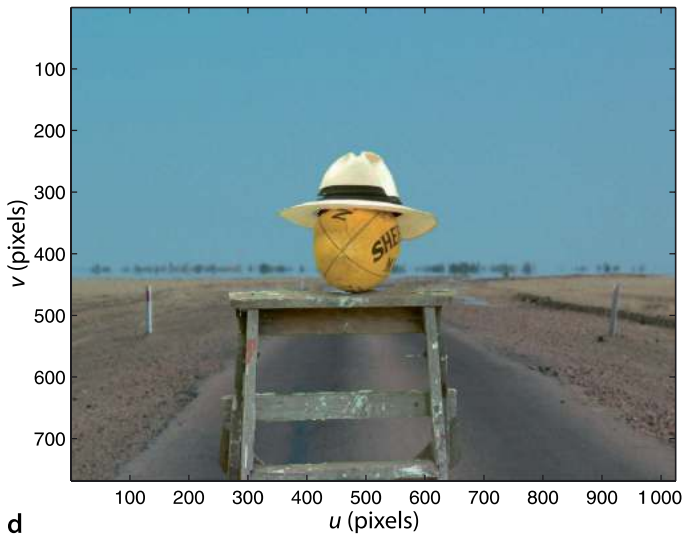
a



b



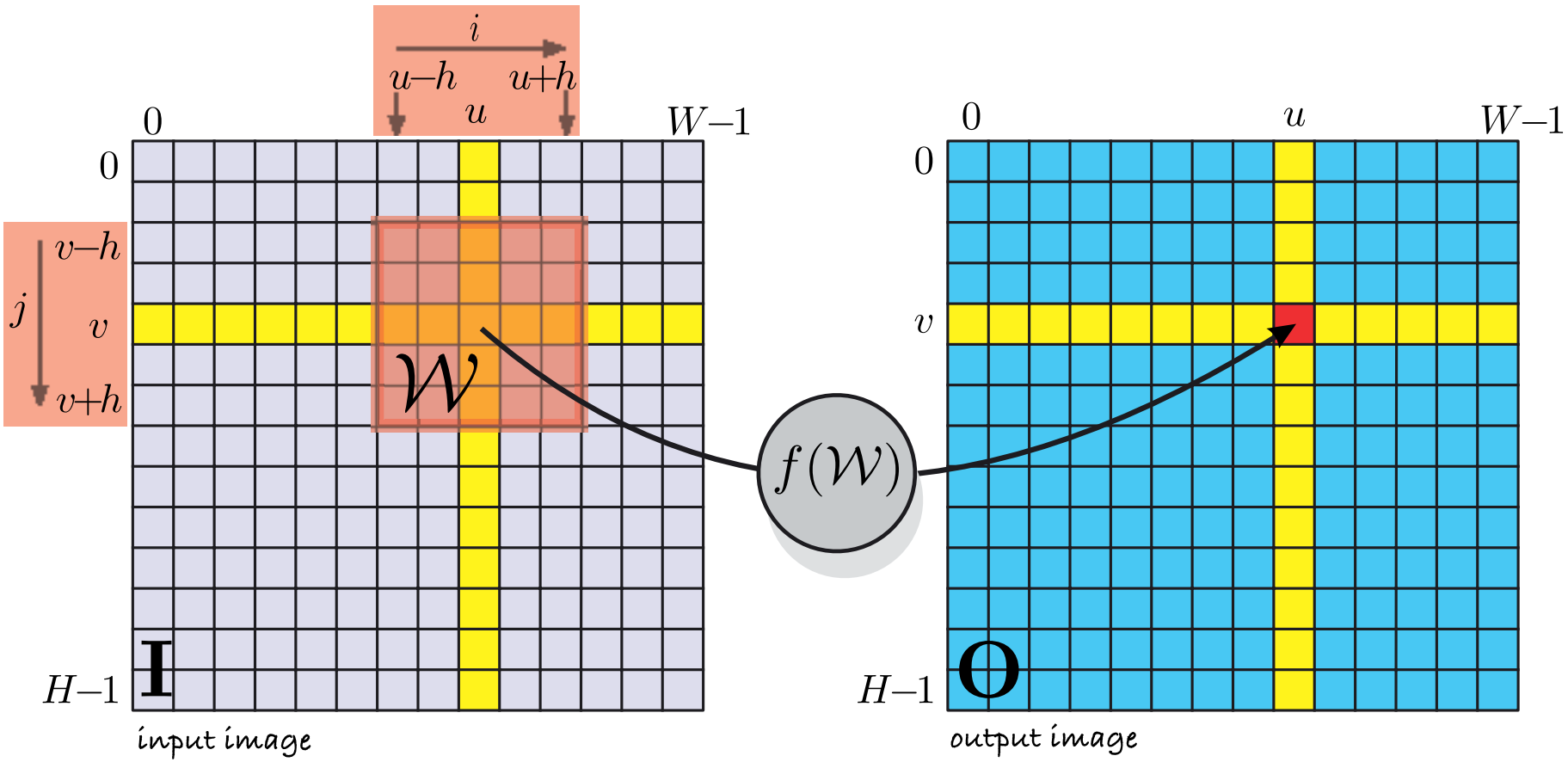
c



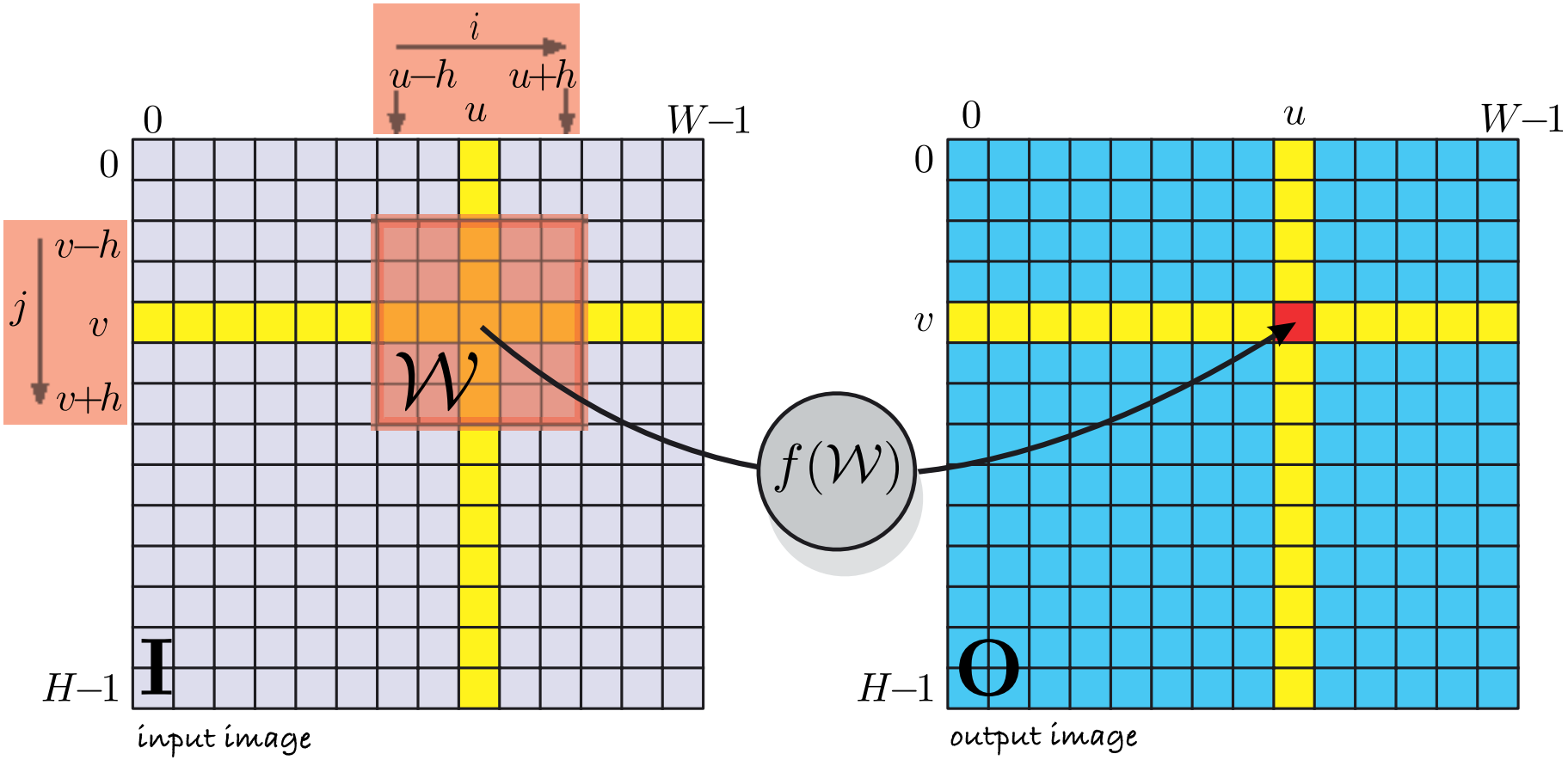
d

Plan

Spatial Operations



Convolution



$$\mathbf{O}[u,v] = \sum_{(i,j)\in\mathcal{W}} \boldsymbol{I}[u+i, v+j]\boldsymbol{K}[i,j], \quad \forall (u,v) \in \mathbf{I}$$

Computational cost: $O(h^2WH)$

Convolution

Properties of convolution. Convolution obeys the familiar rules of algebra, it is commutative

$$A \otimes B = B \otimes A$$

associative

$$A \otimes B \otimes C = (A \otimes B) \otimes C = A \otimes (B \otimes C)$$

distributive (superposition applies)

$$A \otimes (B + C) = A \otimes B + A \otimes C$$

linear

$$A \otimes (\alpha B) = \alpha(A \otimes B)$$

and shift invariant – if $S(\cdot)$ is a spatial shift then

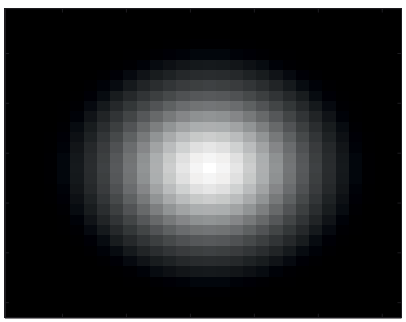
$$A \otimes S(B) = S(A \otimes B)$$

that is, convolution with a shifted image is the same as shifting the result of the convolution with the unshifted image.

Smoothing

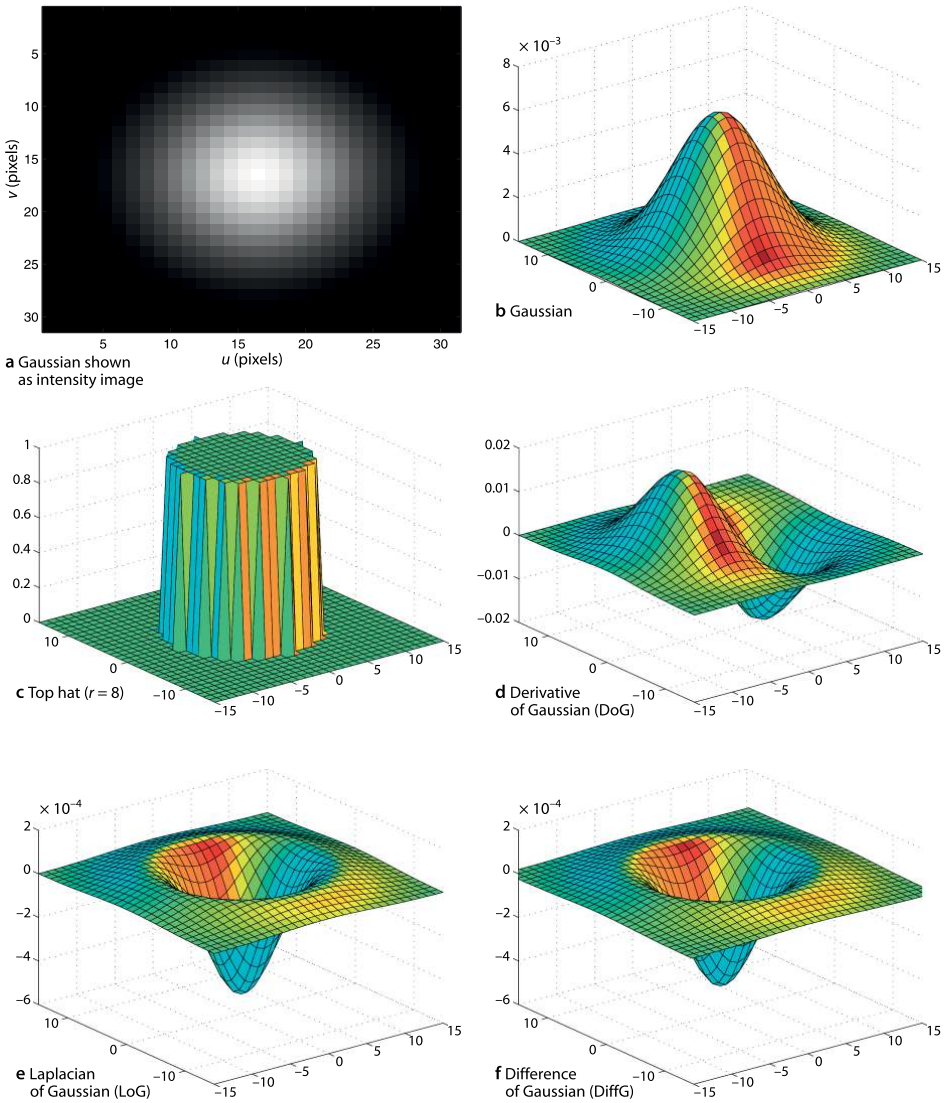


Gaussian kernel:



$$\mathbf{G}(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Popular Kernels



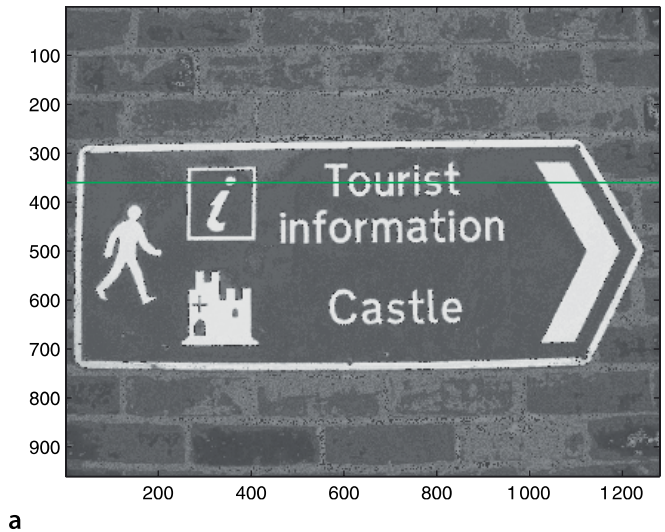
Edge Detection

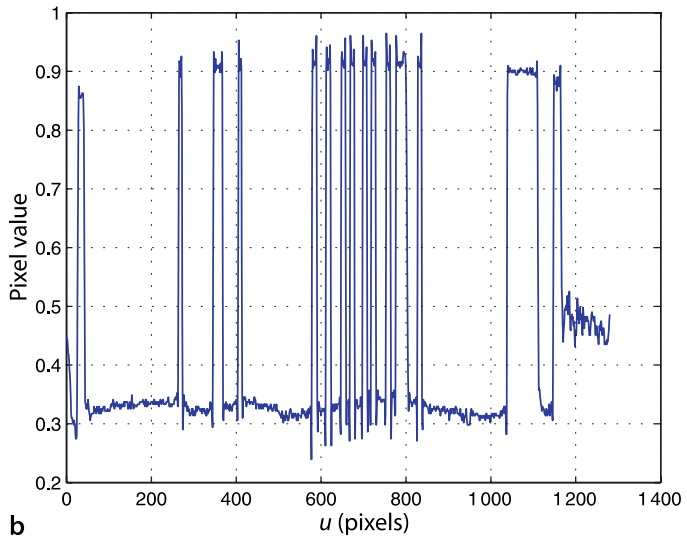
$$p'[v] = p[v] - p[v-1]$$

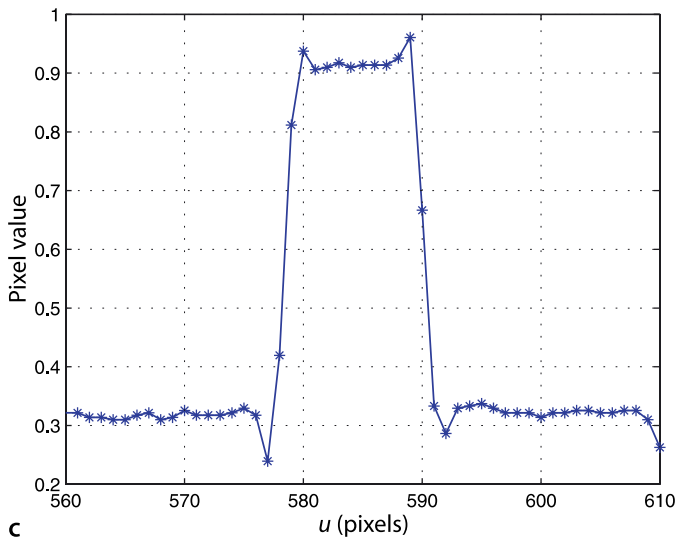
$$p'[v] = \frac{1}{2}(p[v+1] - p[v-1])$$

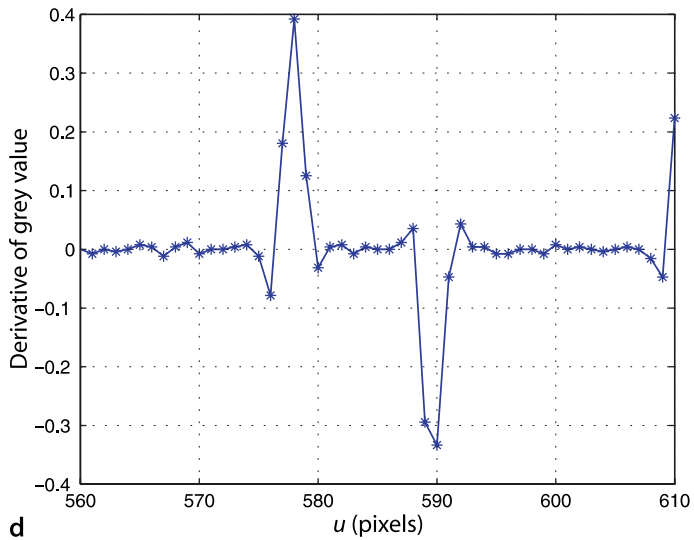
$$\boldsymbol{K} = \begin{pmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}$$

```
>> Du = ksobel
Du =
     -1     0     1
     -2     0     2
     -1     0     1
```



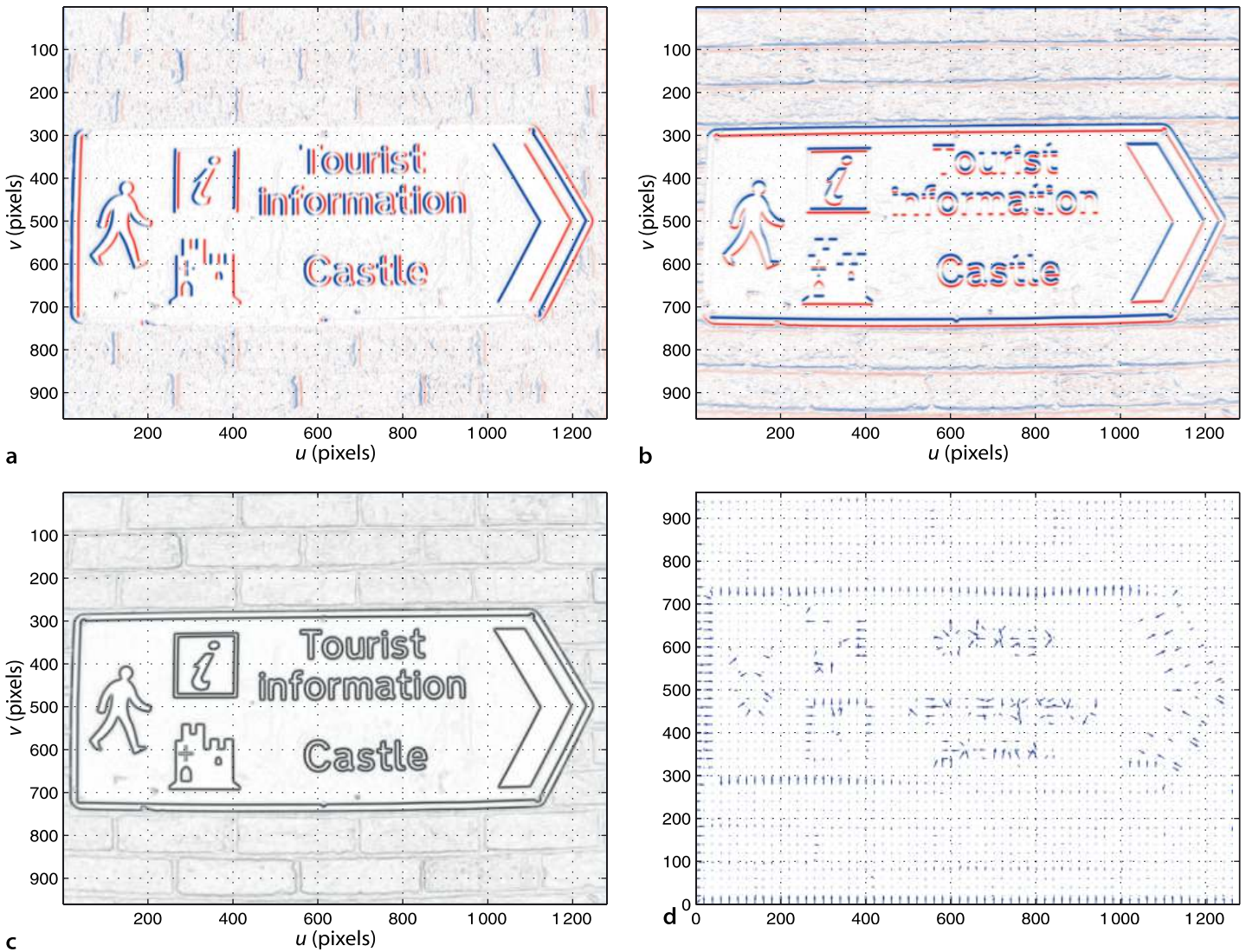






Edge Detection

```
>> Du = ksobel
Du =
     -1     0     1
     -2     0     2
     -1     0     1
```



Noise-robust Edge Detection with Smoothing

- Noise is a stationary random process
- Edge pixels are correlated over large regions

$\rightarrow$ we can reduce the effect of noise with spatial smoothing.

$$\mathbf{I}_u = \mathbf{D} \otimes \big(\mathbf{G}(\sigma) \otimes \mathbf{I}\big)$$

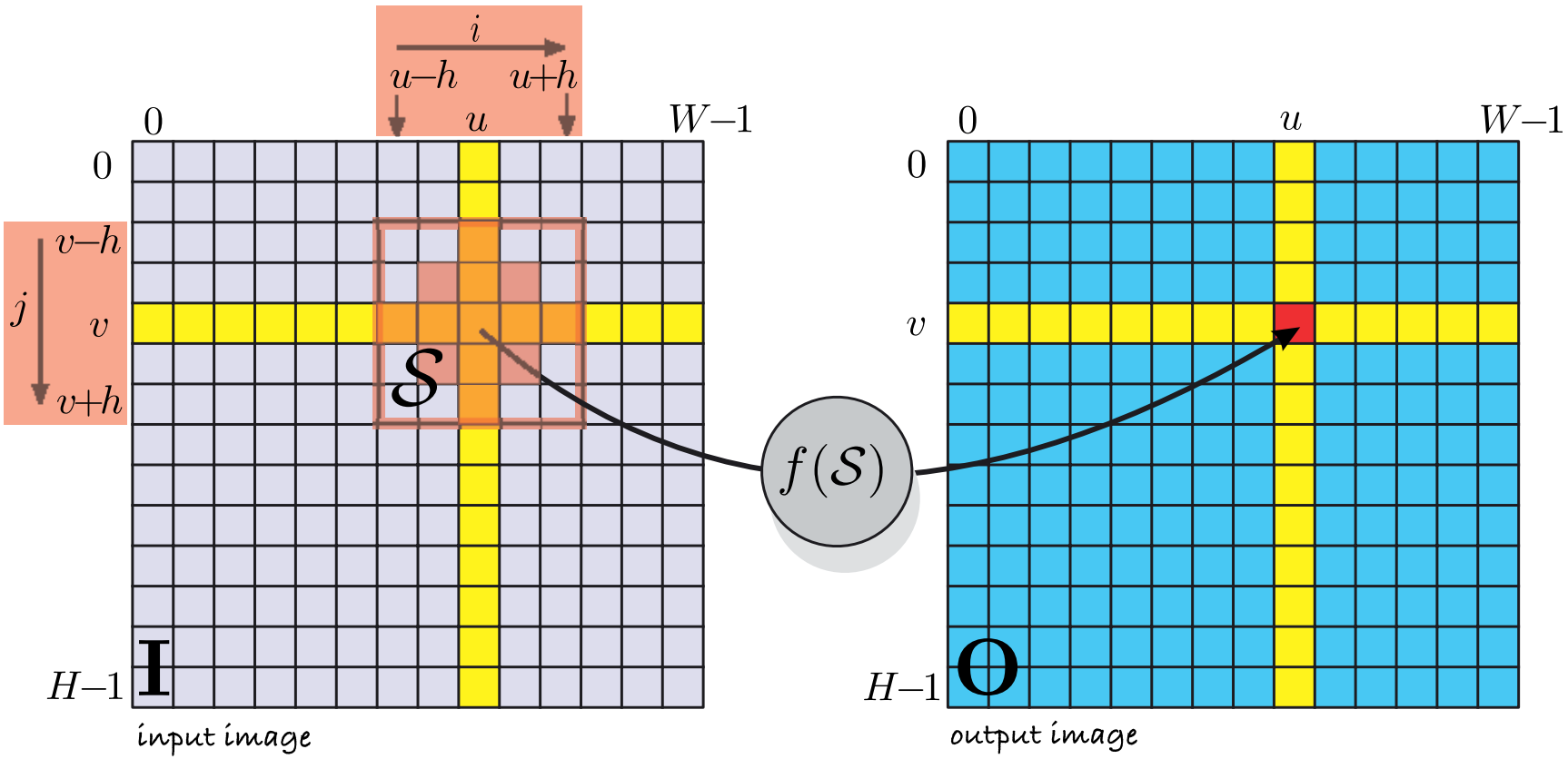
$$\nabla\mathbf{I} = \mathbf{D} \otimes \big(\mathbf{G}(\sigma) \otimes I\big) = \underbrace{\big(\mathbf{D} \otimes \mathbf{G}(\sigma)\big)}_{\text{DoG}} \otimes \mathbf{I}$$

$$\mathbf{G}_u(u, v) = -\frac{u}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

σ can be chosen to

- remove noise
- extract edges of different scales

Mathematical Morphology



$$\mathbf{O}[u,v] = f\big(\mathbf{I}[u+i, v+j]\big), \quad \forall (i,j) \in \mathcal{S}, \quad \forall (u,v) \in \mathbf{I}$$

Opening: Erosion then Dilatation

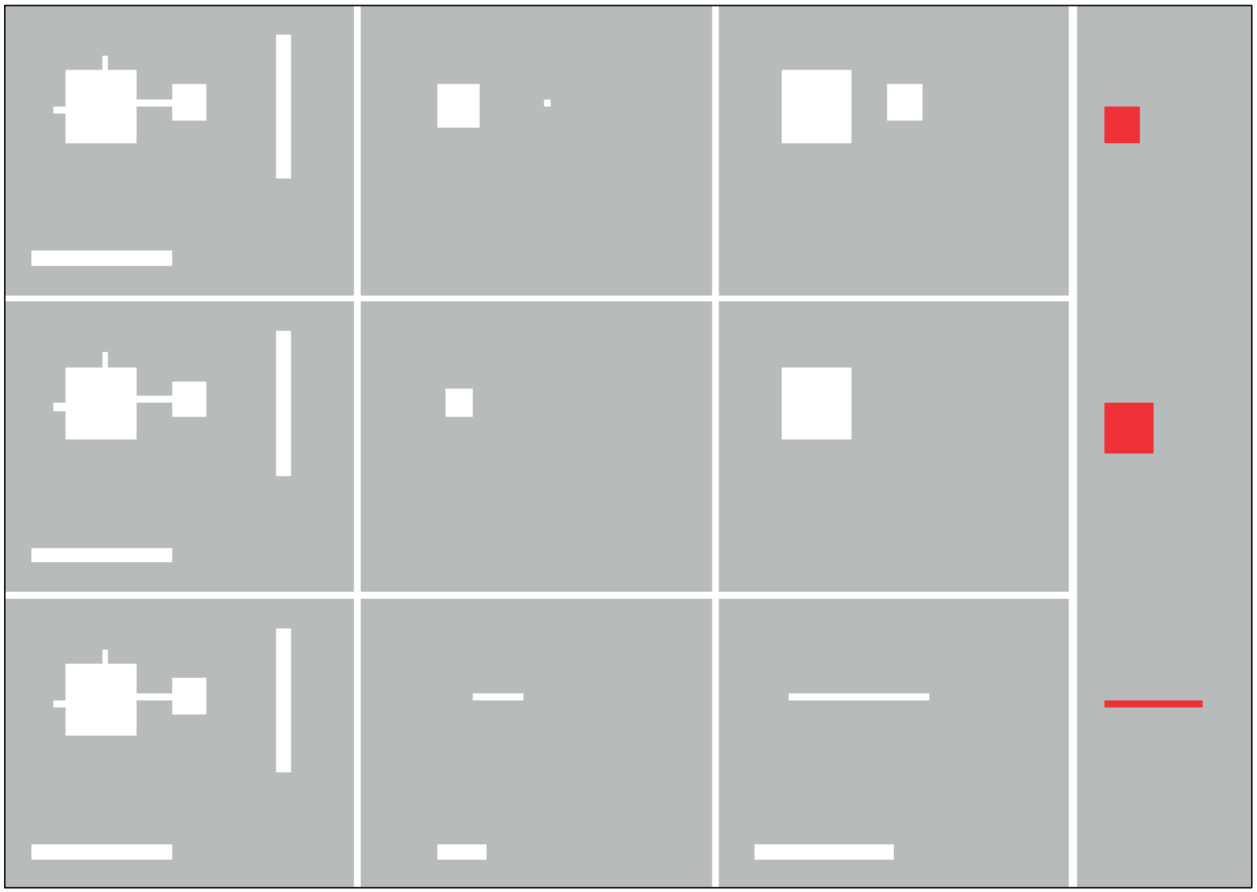
Erosion:
$\boldsymbol{O} = \boldsymbol{I} \ominus \mathcal{S}$

Dilatation:
$\boldsymbol{O} = \boldsymbol{I} \oplus \mathcal{S}$

Relation between erosion and dilatation:
$A \oplus B = \overline{\overline{A} \ominus B}$

$(A \oplus B) \oplus C = A \oplus (B \oplus C)$

$(A \ominus B) \ominus C = A \ominus (B \oplus C)$



$\boldsymbol{I} \circ \boldsymbol{S} = (\boldsymbol{I} \ominus \boldsymbol{S}) \oplus \boldsymbol{S}$

Closing: Dilatation then Erosion

Erosion:
$\boldsymbol{O} = \boldsymbol{I} \ominus \mathcal{S}$

Dilatation:
$\boldsymbol{O} = \boldsymbol{I} \oplus \mathcal{S}$

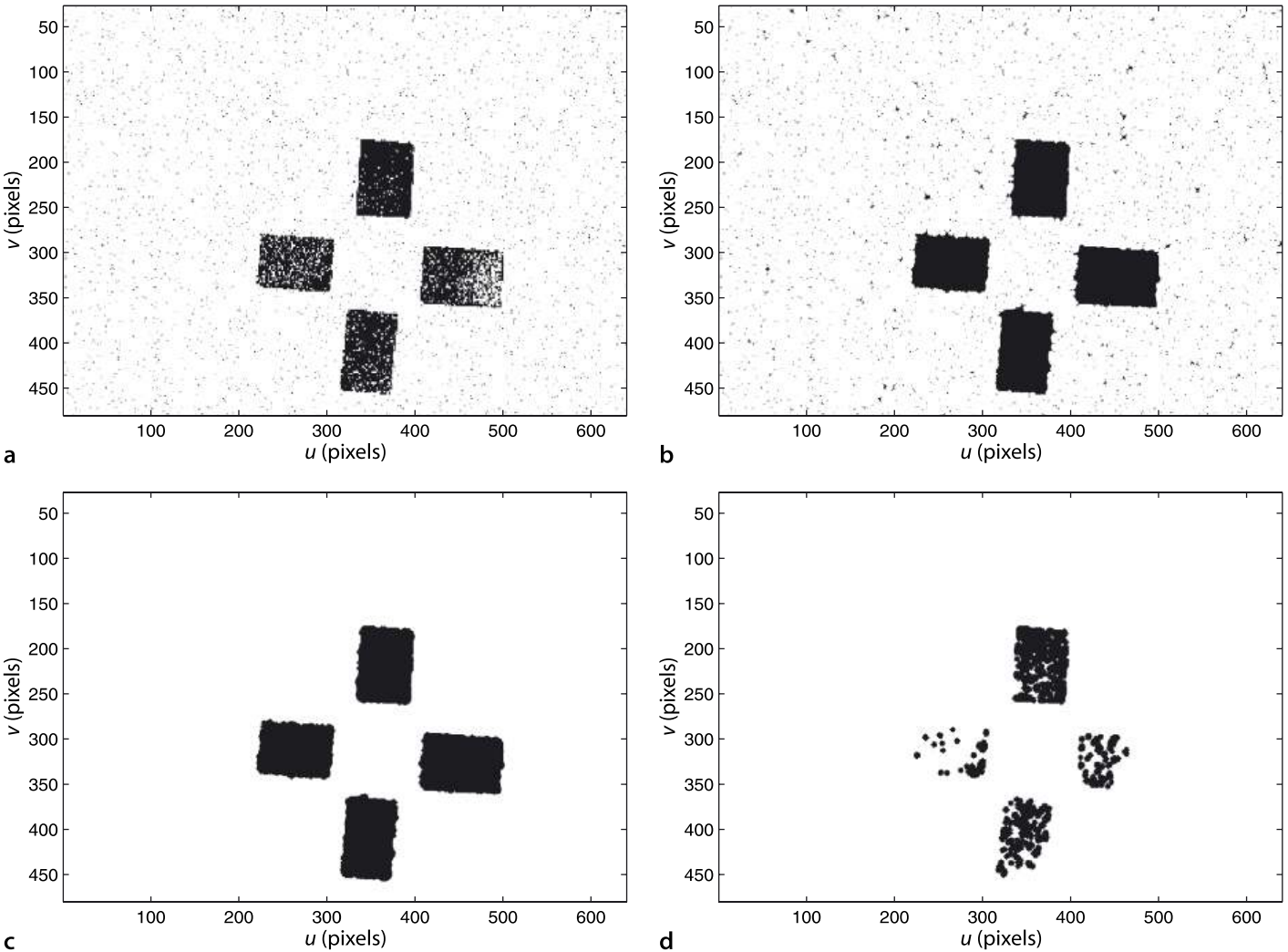
Relation between erosion and dilatation:
$A \oplus B = \overline{\overline{A} \ominus B}$

$(A \oplus B) \oplus C = A \oplus (B \oplus C)$

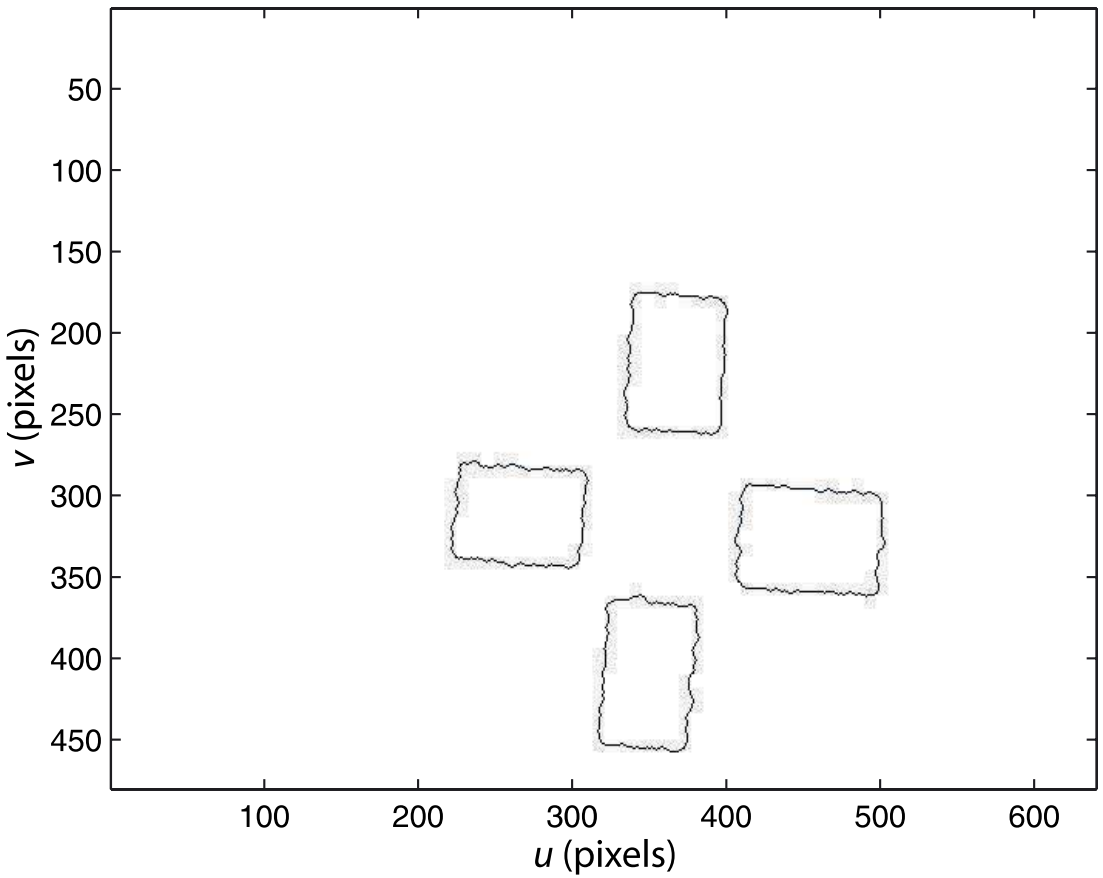
$(A \ominus B) \ominus C = A \ominus (B \oplus C)$

$\boldsymbol{I} \bullet \boldsymbol{S} = (\boldsymbol{I} \oplus \boldsymbol{S}) \ominus \boldsymbol{S}$

Noise Removal: Closing then Opening



Boundary Detection: Erosion then Substraction

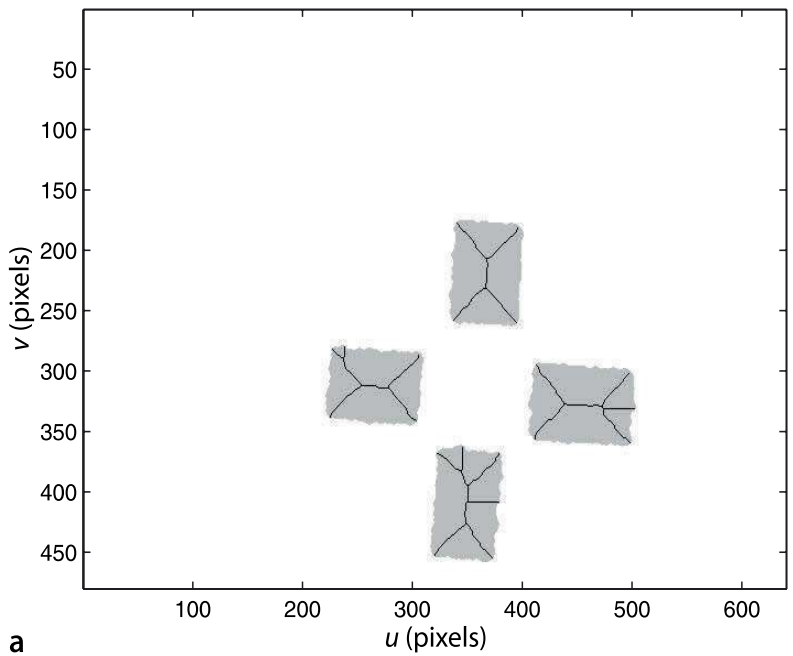


Hit and Miss Transform, Skeletonization

	1	
0	1	1
0	0	

1	1	0
0	1	1
0	0	1

1	1	0
0	1	1
1	0	1



a

Plan

Resizing

Resizing