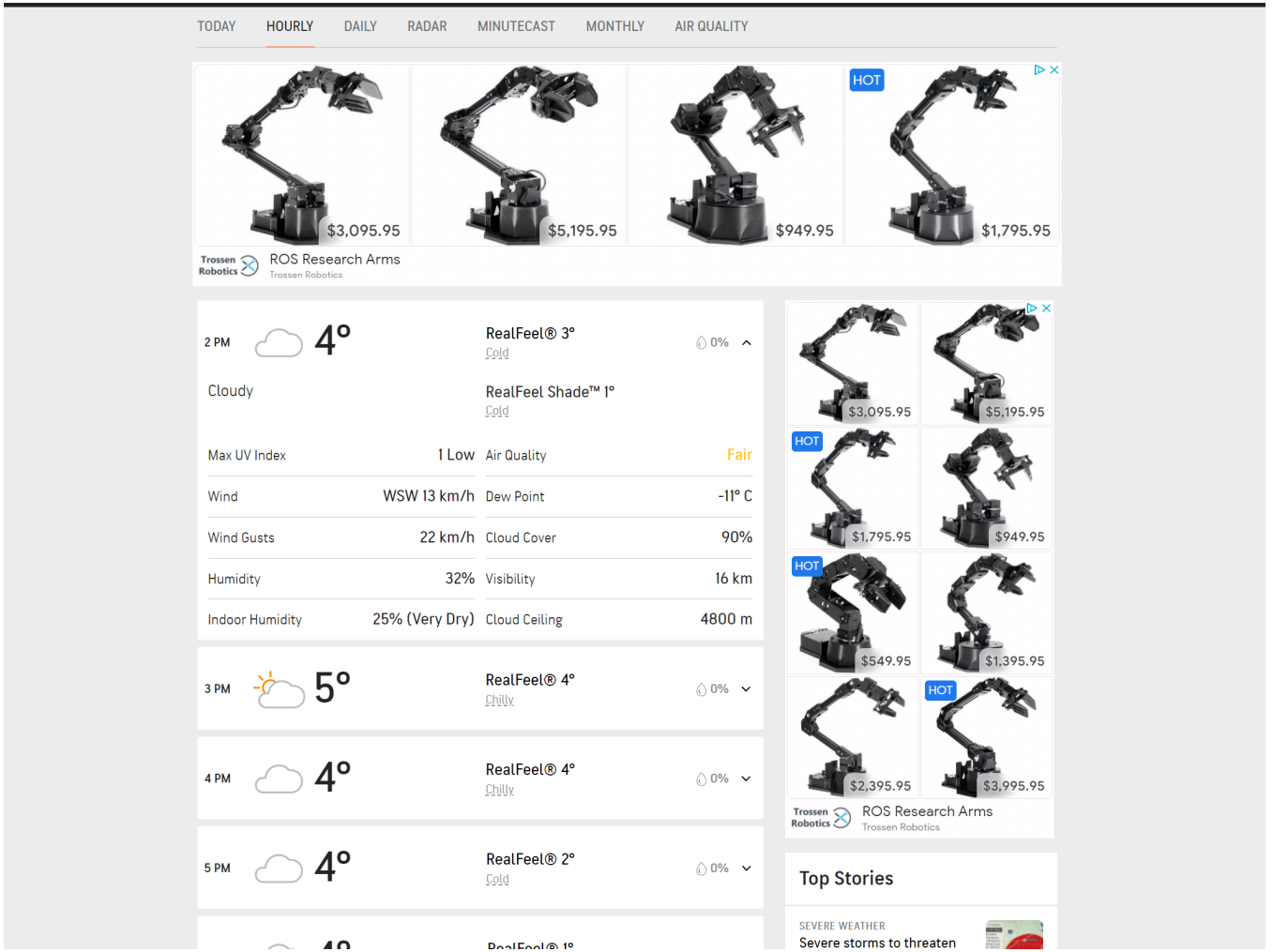
CS 4610/5335 – Lecture 16
Robot Vision (2-D)

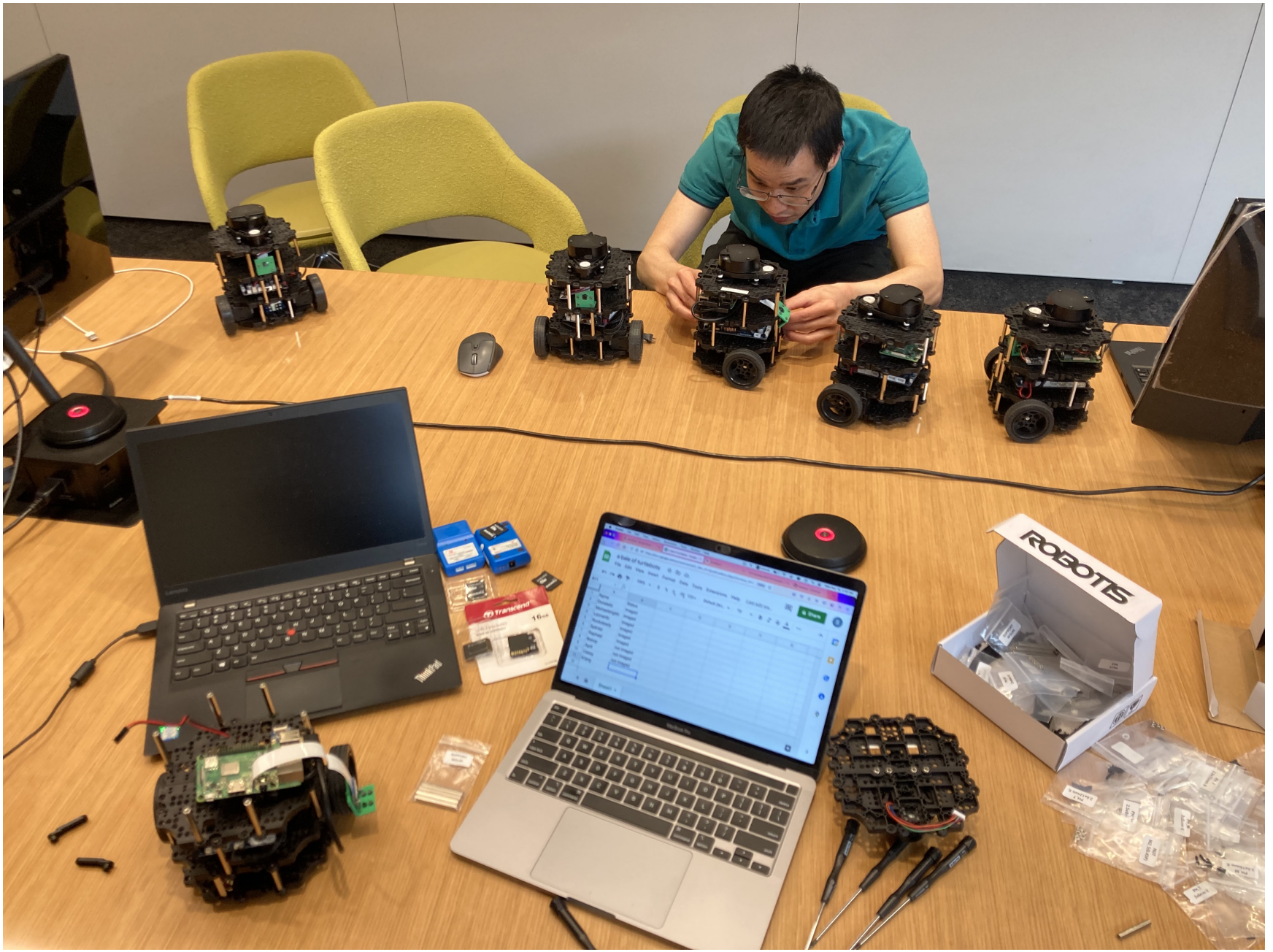
Lawson L.S. Wong
Northeastern University
3/23/22

Material adapted from:
1. Robert Platt, CS 4610/5335
2. Peter Corke, Robotics, Vision and Control

Over spring break ...



Over spring break ...



… just in today!

Announcements

Piazza @202

- Updated course schedule
- Ex4 E1-E4 reference outputs
- Check/update your Piazza project thread

Piazza @206

- Ex4 deadline / late policy

Announcements

Piazza @202
- Updated course schedule
- Ex4 E1-E4 reference outputs
- Check/update your Piazza project thread

Piazza @206
- Ex4 deadline / late policy

Project:
- Pick up hardware soon? (likely starting Friday)
- 3/28: Try to install ROS + simulator before session
- 4/11: Short check-ins with teams

Outline

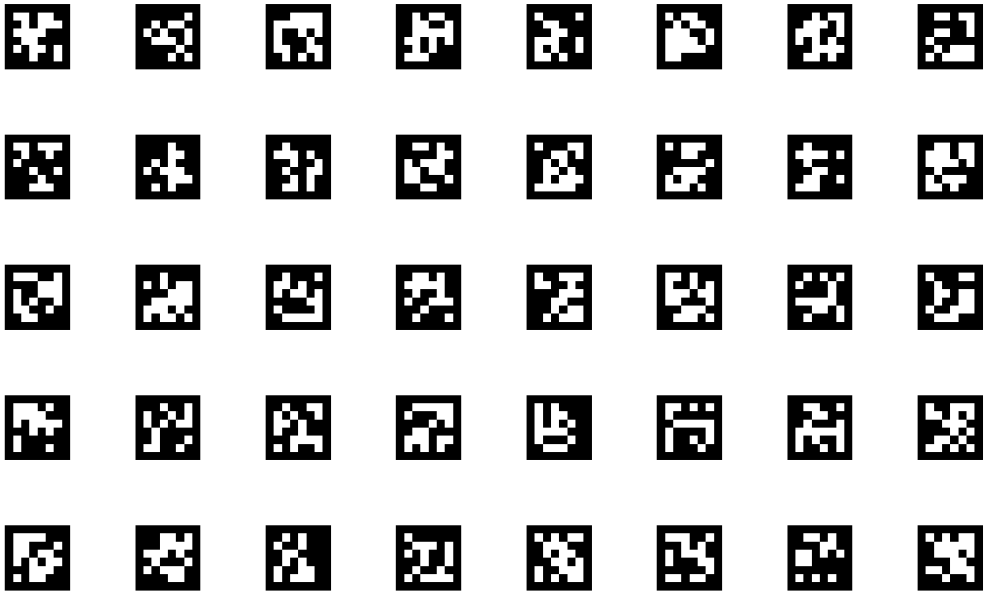
MOAR SLAM
- Rao-Blackwellized SLAM (e.g., FastSLAM)
  - with particle filtering / sequential Monte-Carlo
- Occupancy-grid mapping and SLAM
- RGB-D SLAM
- Pose-graph SLAM (e.g., GraphSLAM)

Trajectory optimization
- Direct transcription
- Direct shooting
- Model-predictive control (MPC)

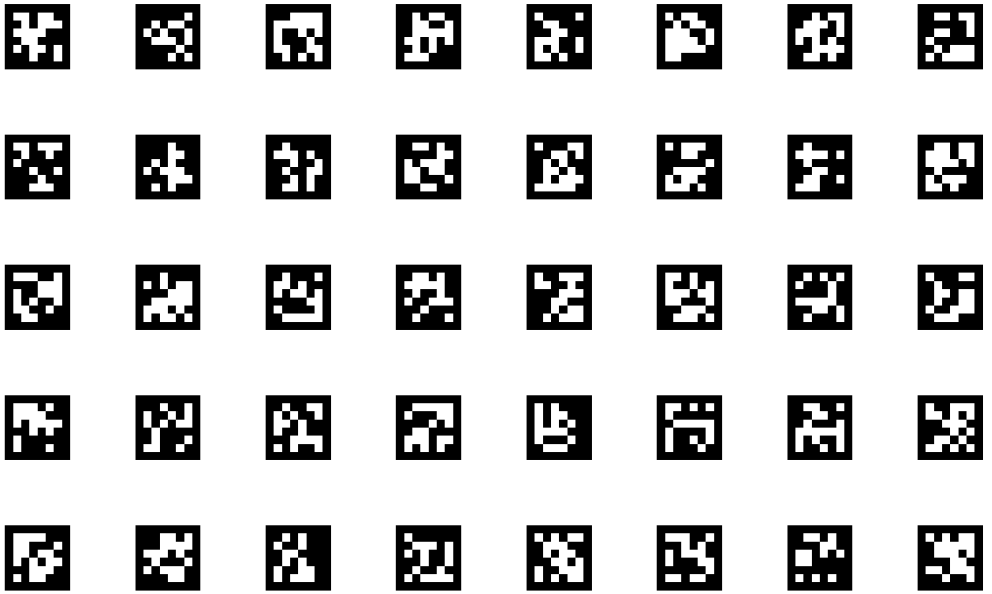
Linear control
- Linear-quadratic regulator (LQR)
- Time-varying LQR
- Linear-quadratic-Gaussian (LQG)

The first rule of perception

The first rule of perception



The first rule of perception



2-D images

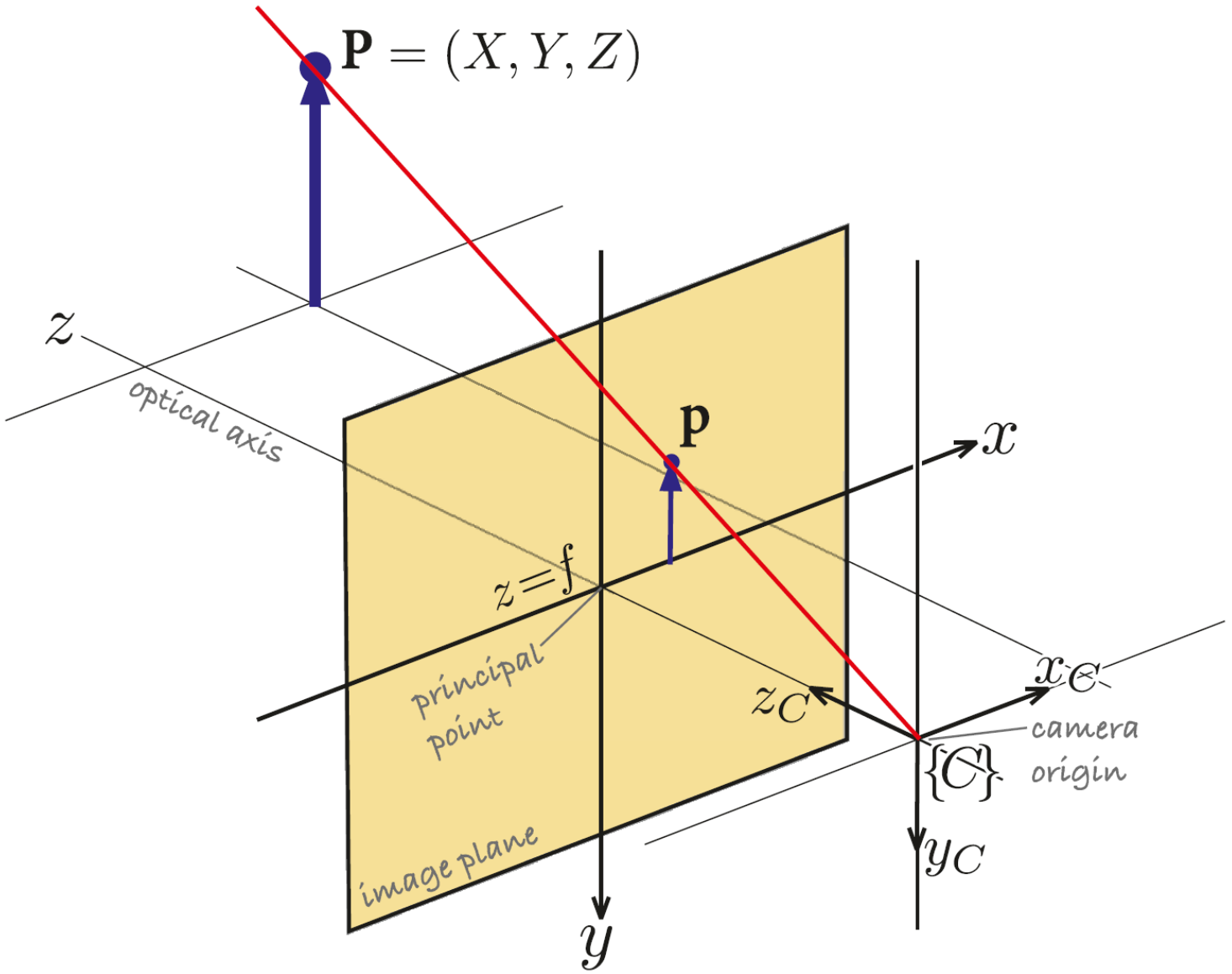


Fig. 11.3.
The central-projection model. The image plane is at a distance f in front of the camera's origin, and on which a noninverted image is formed. The camera's coordinate frame is right-handed with the z-axis defining the center of the field of view

2-D images

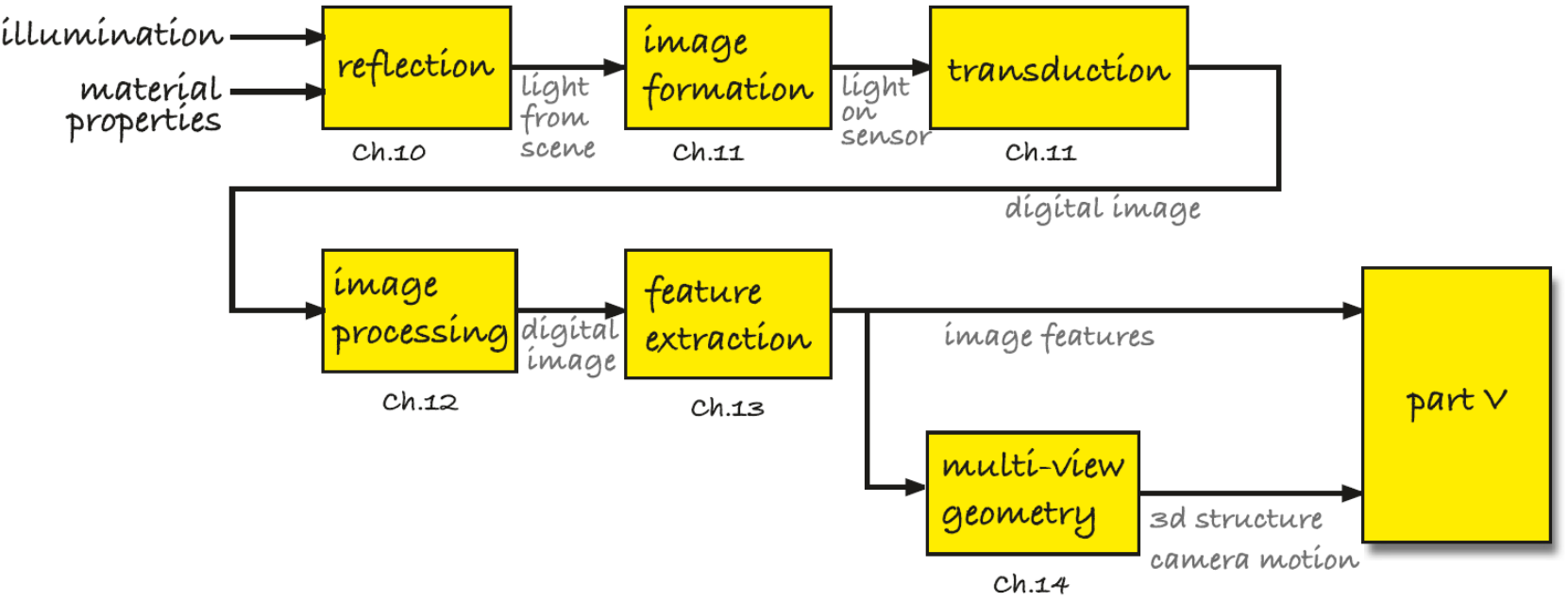
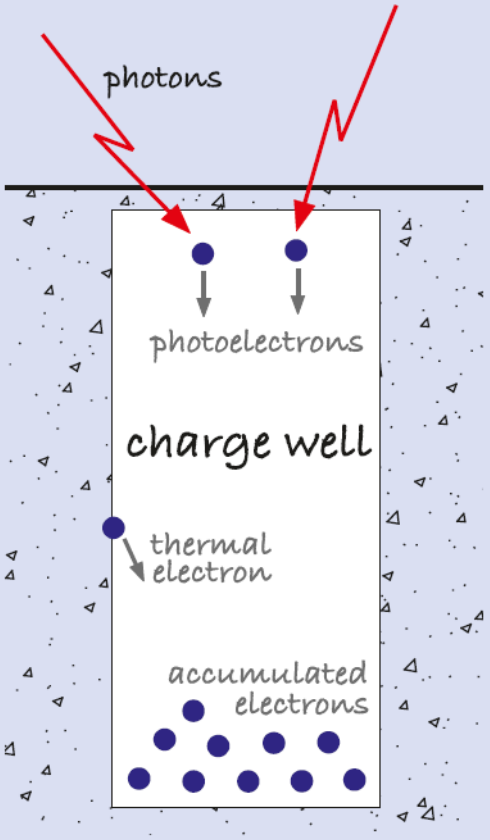


Fig. IV.3.
Steps involved in image processing

2-D images

The charge well also accumulates thermally generated electrons, the dark current, which is proportional to temperature and is a source of noise – extreme low light cameras are cooled. Another source of noise is pixel nonuniformity due to adjacent pixels having a different gain or offset – uniform illumination therefore leads to pixels with different values which appears as additive noise. The charge well has a maximum capacity and with excessive illumination surplus electrons can overflow into adjacent charge wells leading to flaring and bleeding.

At the end of the exposure interval the accumulated charge (thermal- and photo-electrons) is read. For low-cost CMOS sensors the charge wells are connected sequentially via a switching network to one or more on-chip analog to digital converters. This results in a rolling shutter and for high speed relative motion this leads to tearing or jello effect as shown to the right. More expensive CMOS and CCD sensors have a global shutter – they make a temporary snapshot copy of the charge in a buffer which is then digitized sequentially.



The exposure on the sensor is

$$H = qL\frac{T}{N^2}\ \mathrm{lx\,s}$$

where L is scene luminance (in nit), T is exposure time, N is the f-number (inverse aperture diameter) and $q \approx 0.7$ is a function of where S_{SOS} is the ISO rating – standard output sensitivity (SOS) – of the digital camera. Higher ISO increases image brightness by greater amplification of the measured charge but the various noise sources are also amplified leading to increased image noise which is manifested as graininess.

In photography the camera settings that control image brightness can be combined into an exposure value (EV)

$$\mathrm{EV} = \log_2 \frac{N^2}{T}$$

and all combinations of f-number and shutter speed that have the same EV value yield the same exposure. This allows a tradeoff between aperture (depth of field) and exposure time (motion blur). For most low-end cameras the aperture is fixed and the camera controls exposure using T instead of relying on an expensive, and slow, mechanical aperture. A difference of 1 EV is a factor of two change in exposure which photographers refer to as a *stop*. Increasing EV results in a darker image – most DSLR cameras allow you to manually adjust EV relative to what the camera's lightmeter has determined.

[a] Which is backward compatible with historical scales (ASA, DIN, ISO) devised to reflect the sensitivity of chemical films for cameras – a higher number reflected a more sensitive or "faster" film.

[b] 18% saturation, middle grey, of 8-bit pixels with gamma of 2.2.

2-D images

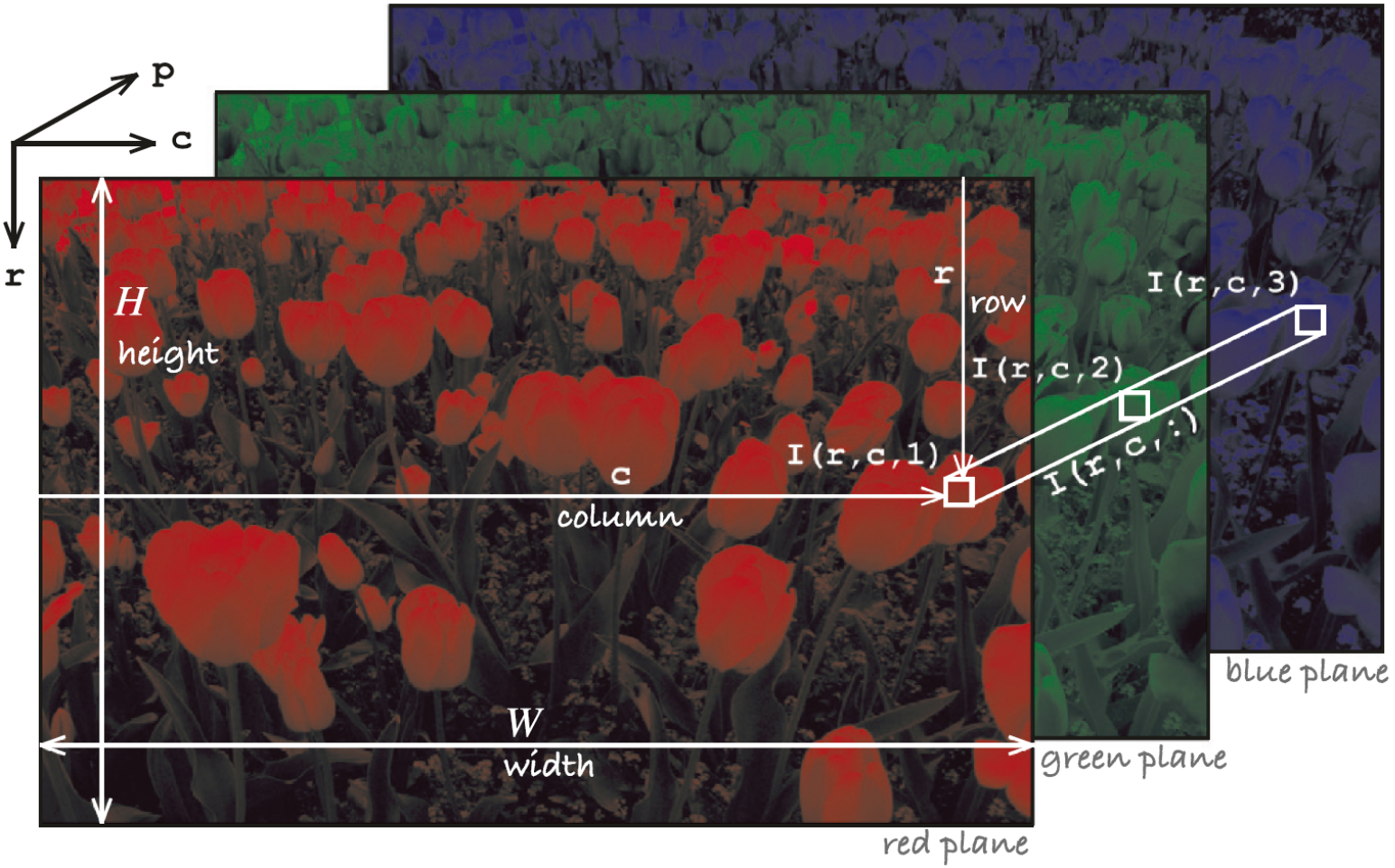


Fig. 12.2.
Color image shown as a 3-dimensional structure with dimensions: row, column, and color plane

Image processing

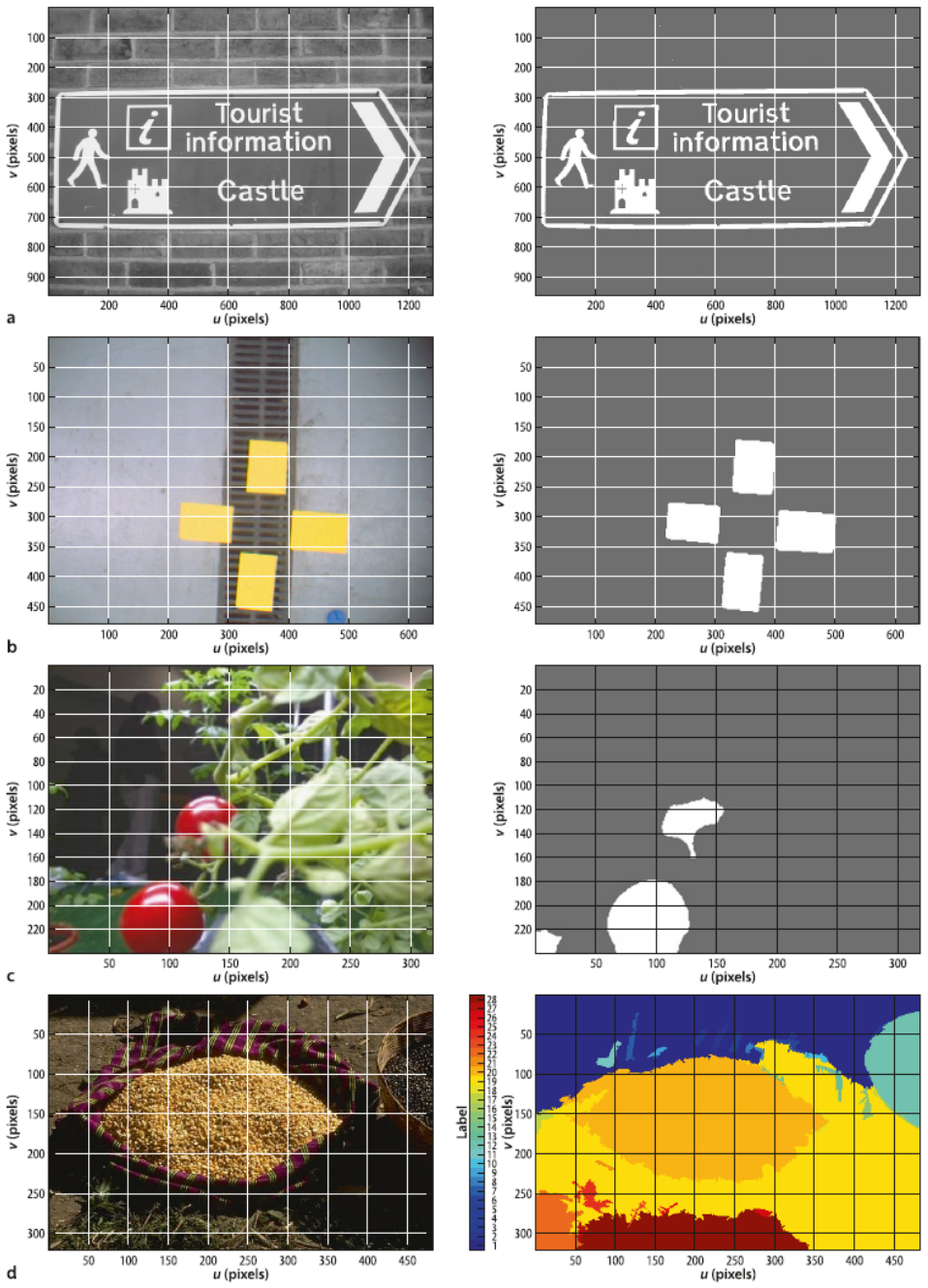


Fig. 13.1.
Examples of pixel classification. The left-hand column is the input image and the right-hand column is the classification. The classification is application specific and the pixels have been classified as either object (*white*) or not-object (*black*). The objects of interest are **a** the individual letters on the sign; **b** the yellow targets; **c** the red tomatoes. **d** is a multi-level segmentation where pixels have been assigned to 28 classes that represent locally homogeneous groups of pixels in the scene

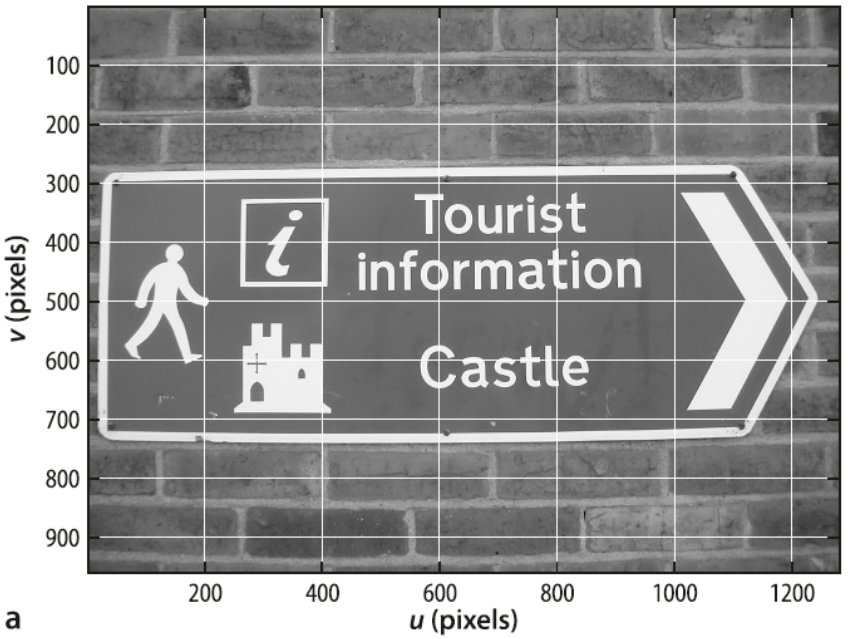
Classification

A common approach to binary classification of pixels is the monadic operator

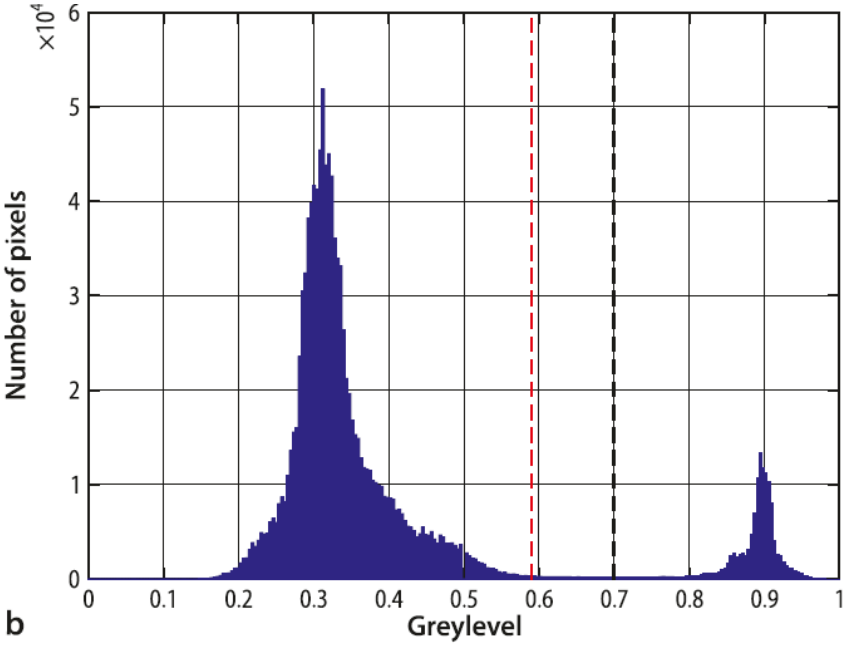
$$c[u,v] = \begin{cases} 0, & \text{if } I[u,v] < t \\ 1, & \text{if } I[u,v] \geq t \end{cases} \quad \forall (u,v) \in I$$

where the decision is based simply on the value of the pixel I. This approach is called thresholding and t is referred to as the *threshold*.

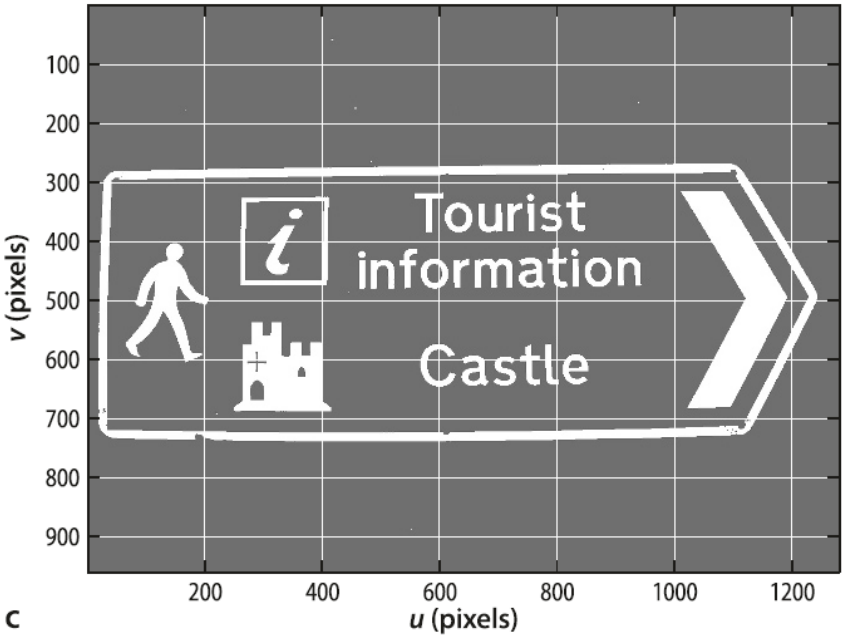
Classification



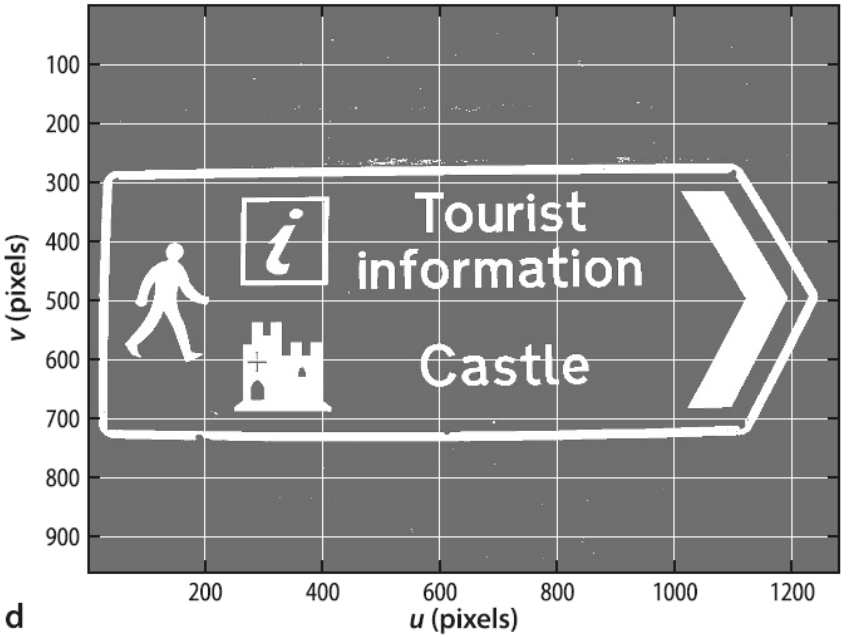
a



b

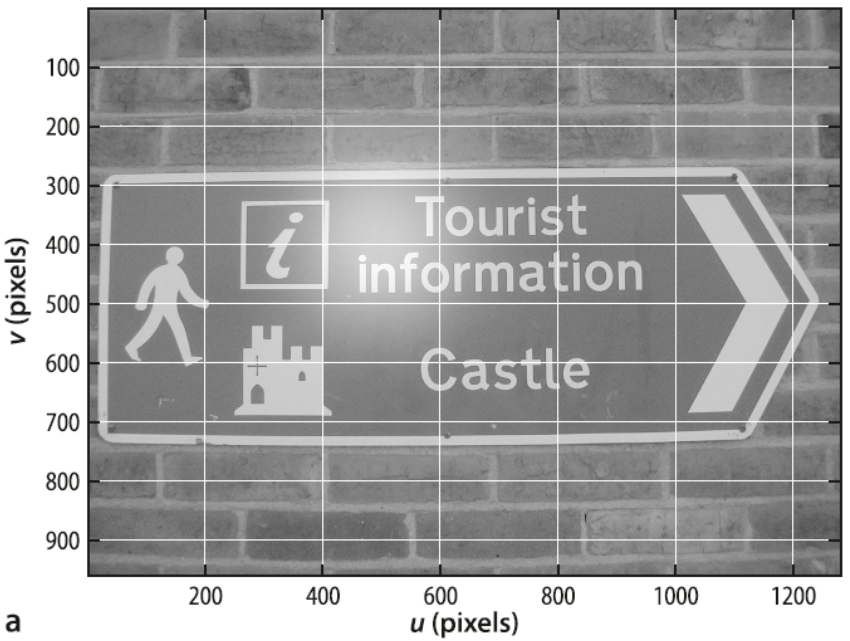


c

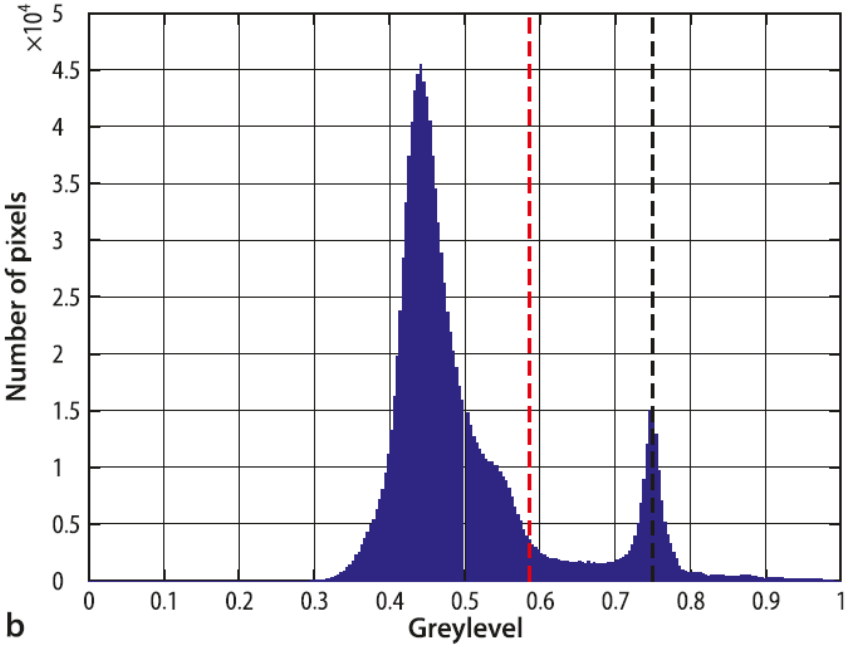


d

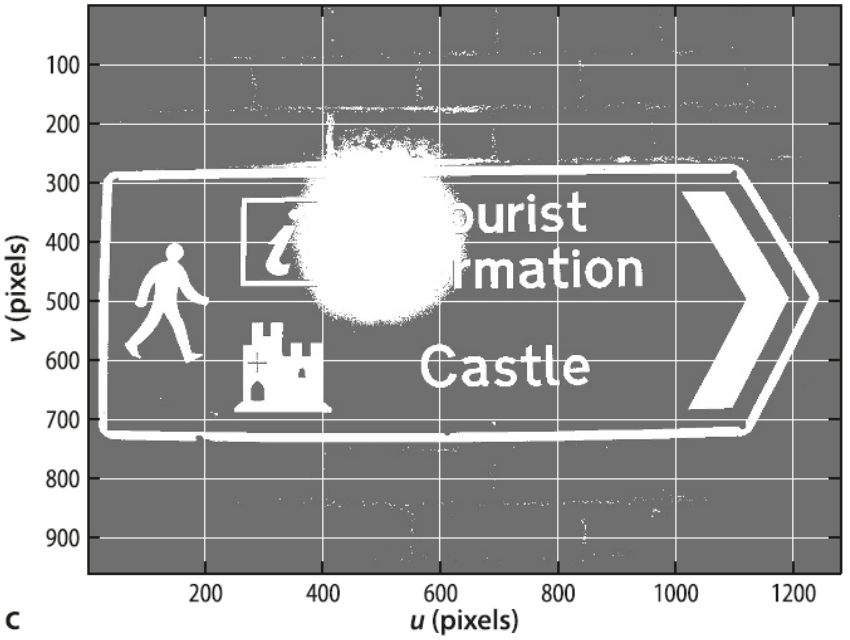
Classification



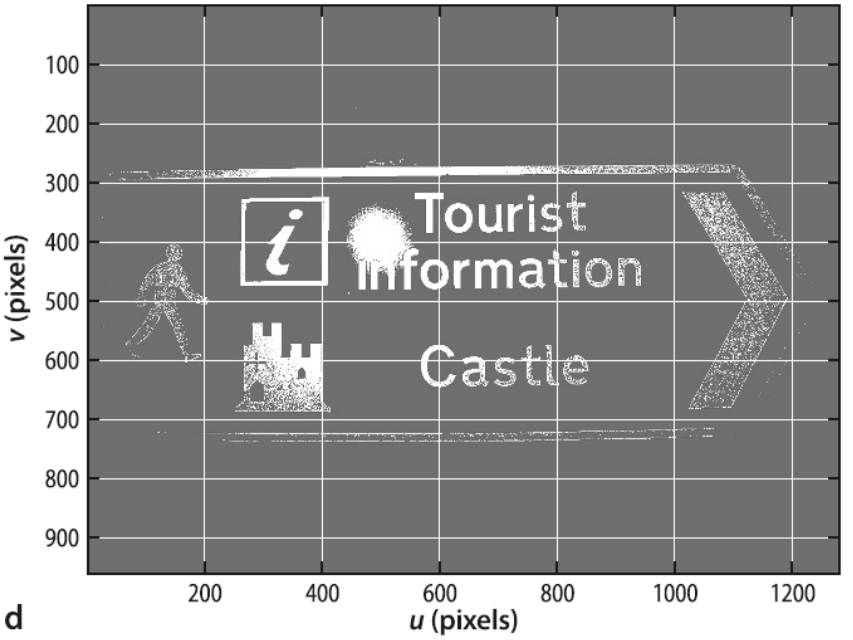
a



b

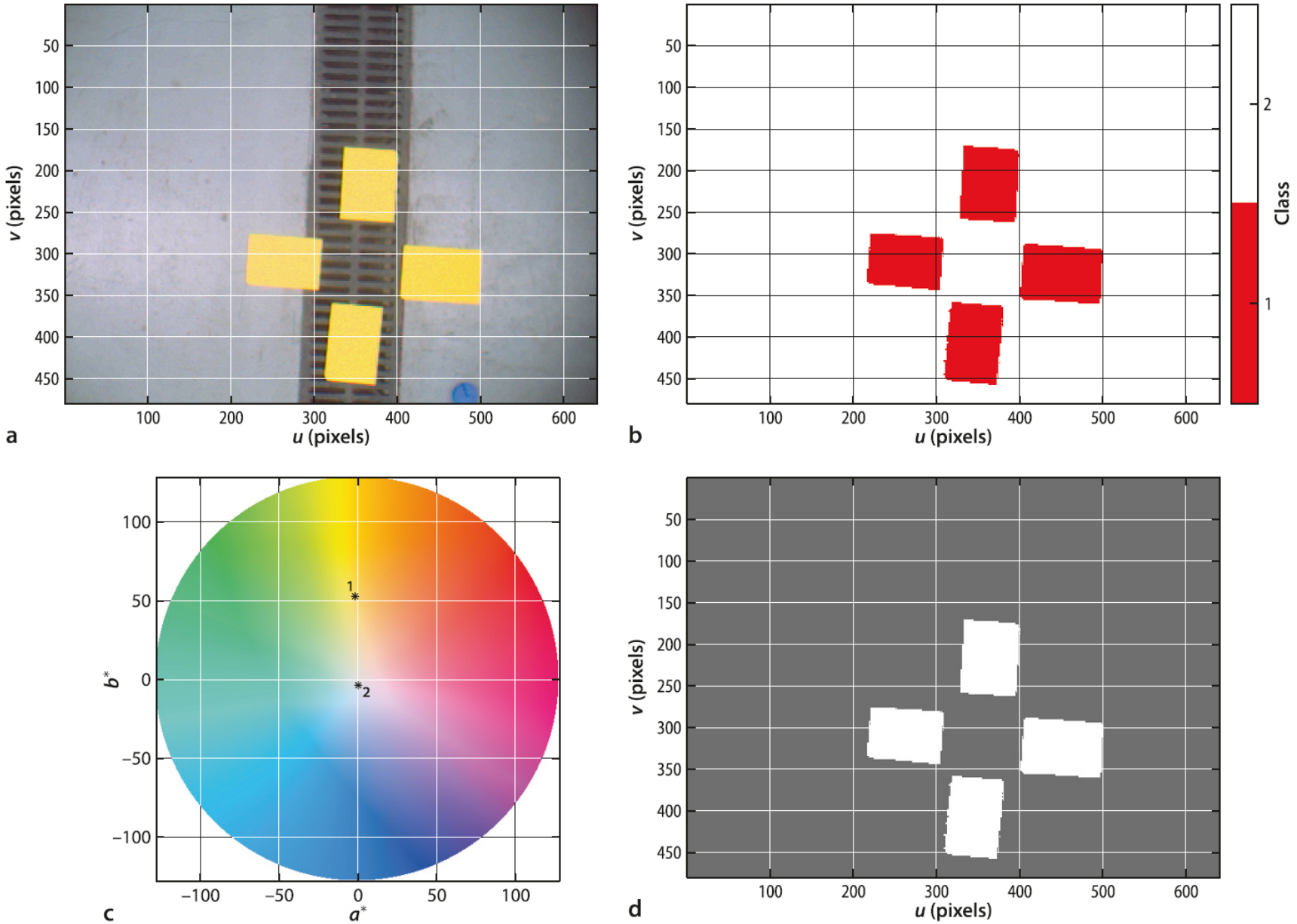


c



d

Clustering



Clustering

***k*-means clustering** is an iterative algorithm for grouping n-dimensional points into k spatial clusters. Each cluster is defined by a center point which is an n-vector $\boldsymbol{c}_i$, $i \in [1, k]$. At each iteration all points are assigned to the *closest* cluster center, and then each cluster center is updated to be the mean of all the points assigned to the cluster.

The algorithm is implemented by the Toolbox function `kmeans`. The distance metric used is Euclidean distance. The k-means algorithm requires an initial estimate of the center of each cluster and this can be provided in various ways, see the documentation. By default `kmeans` randomly selects k of the provided points, and this means the algorithm will return different results at each invocation.

To demonstrate we choose 500 random 2-dimensional points

```
>> a = rand(2,500);
```

where `a` is a 2×500 matrix with one point per column. We will cluster this data into three sets

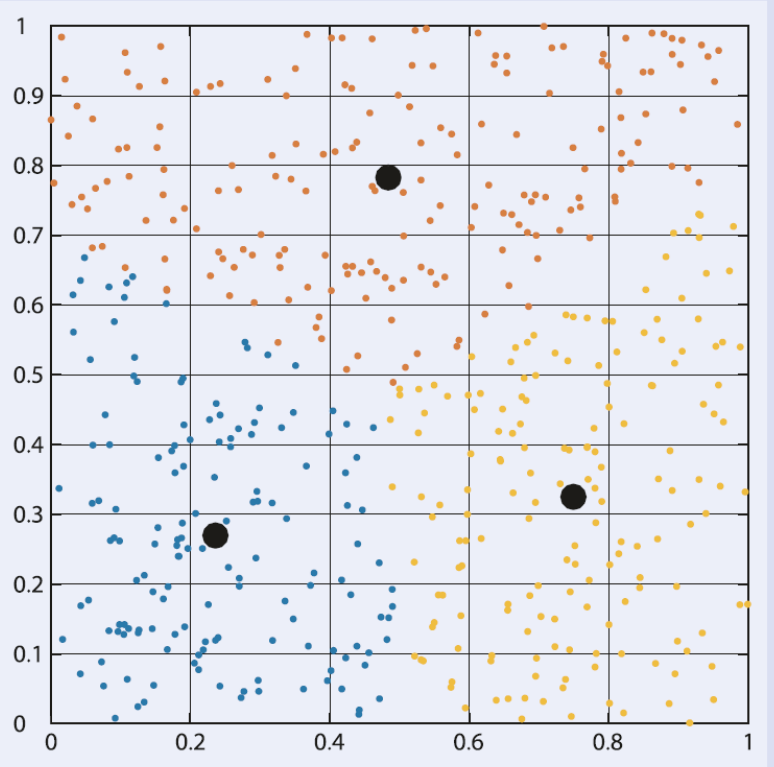
```
>> [cls,centre,r] = kmeans(a, 3);
```

where `cls` is a 500-vector whose elements specify the class of the corresponding column of `a`. `center` is a 2×3 matrix whose columns specify the center of each 2-dimensional cluster and `r` is the residual – the norm of the distance of every point from its assigned cluster centroid.

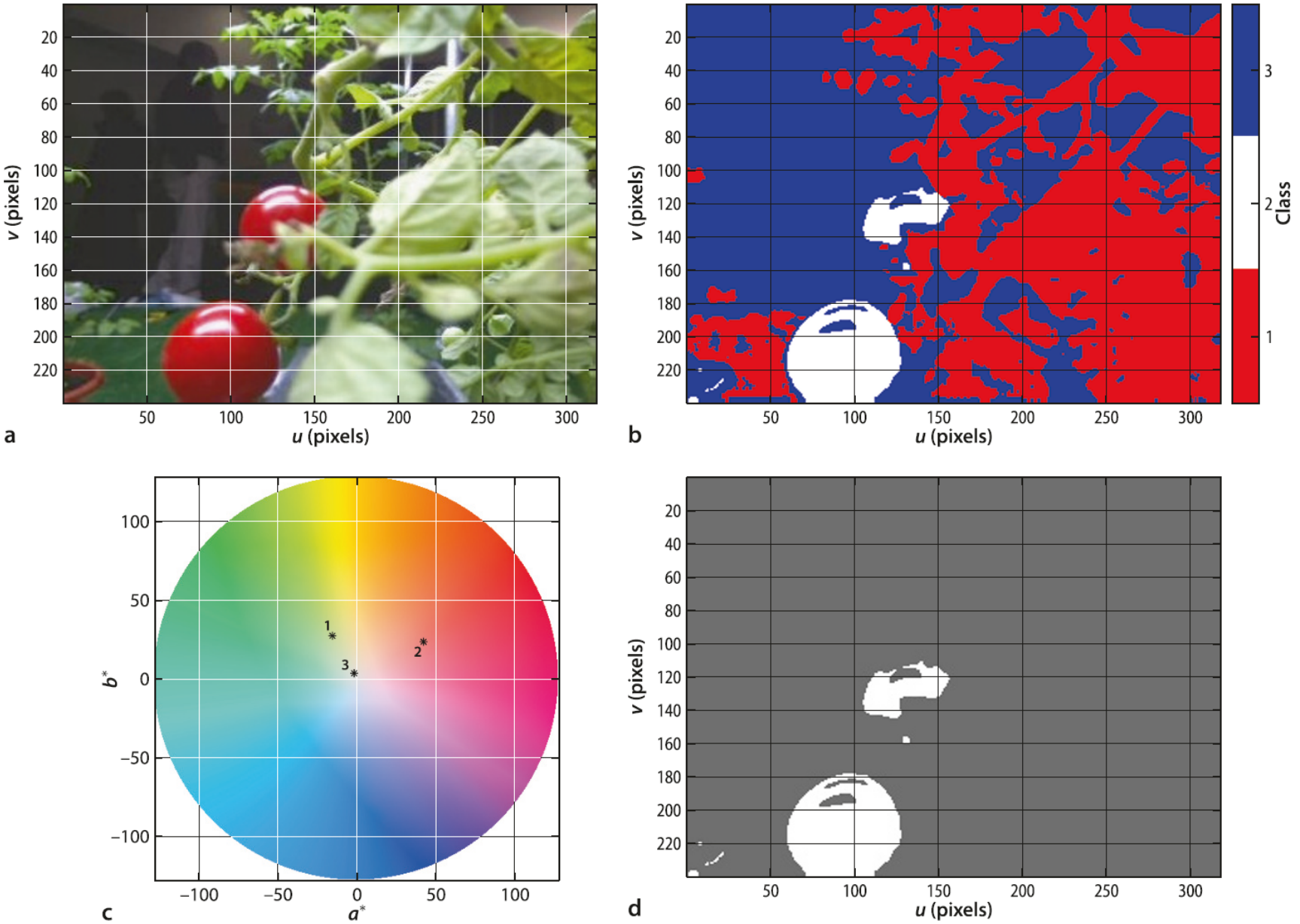
We plot the points in each cluster with different colors

```
>> hold on
>> for i=1:3
     plot( a(1,cls==i), a(2,cls==i), '.' );
   end
```

and it is clear that the points have been sensibly partitioned. The centroids center have been superimposed as black dots.



Clustering



Filtering

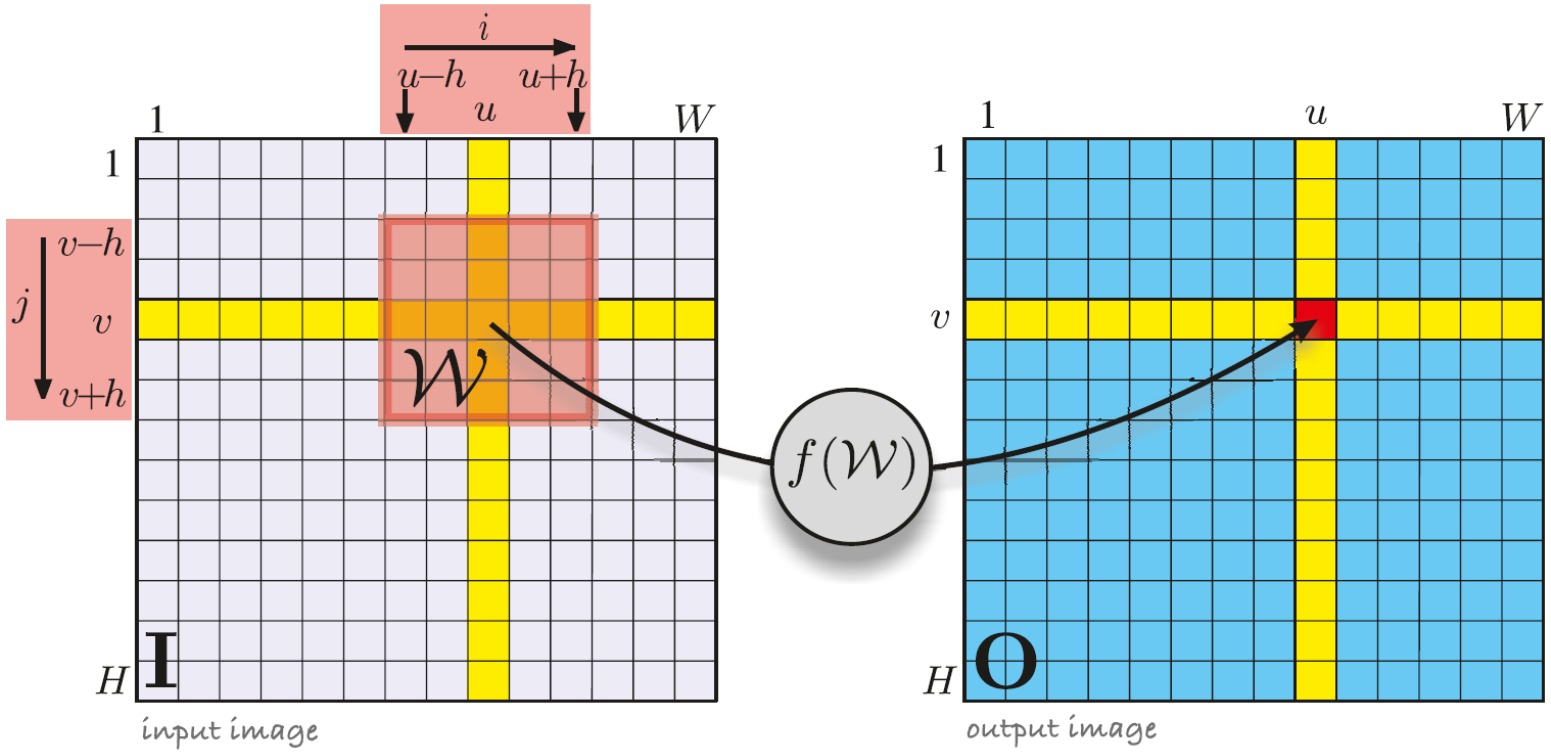


Fig. 12.12. Spatial image processing operations. The *red shaded region* shows the window $\mathcal{W}$ that is the set of pixels used to compute the output pixel (show in *red*)

Filtering

12.5.1 Linear Spatial Filtering

A very important linear spatial operator is correlation

$$\mathbf{O}[u,v] = \sum_{(i,j)\in\mathcal{W}} \boldsymbol{I}[u+i, v+j]\boldsymbol{K}[i,j],\ \forall(u,v)\in\mathbf{I} \tag{12.1}$$

where $\boldsymbol{K} \in \mathbb{R}^{w\times w}$ is the kernel and the elements are referred to as the filter coefficients. For every output pixel the corresponding window of pixels from the input image $\mathcal{W}$ is multiplied element-wise with the kernel $\boldsymbol{K}$. The center of the window and kernel is considered to be coordinate (0, 0) and $i, j \in [-h, h] \subset \mathbb{Z}\times\mathbb{Z}$. This can be considered as the weighted sum of pixels within the window where the weights are defined by the kernel $\boldsymbol{K}$. Correlation is often written in operator form as

$$\mathbf{O} = \boldsymbol{K} \otimes \mathbf{I}$$

A closely related operation is convolution

where $\boldsymbol{K} \in \mathbb{R}^{w\times w}$ is the convolution kernel. Note that the sign of the i and j indices has changed in the first term. Convolution is often written in operator form as

$$\mathbf{O} = \boldsymbol{K} * \mathbf{I}$$

As we will see convolution is the workhorse of image processing and the kernel $\boldsymbol{K}$ can be chosen to perform functions such as smoothing, gradient calculation or edge de-

Filtering

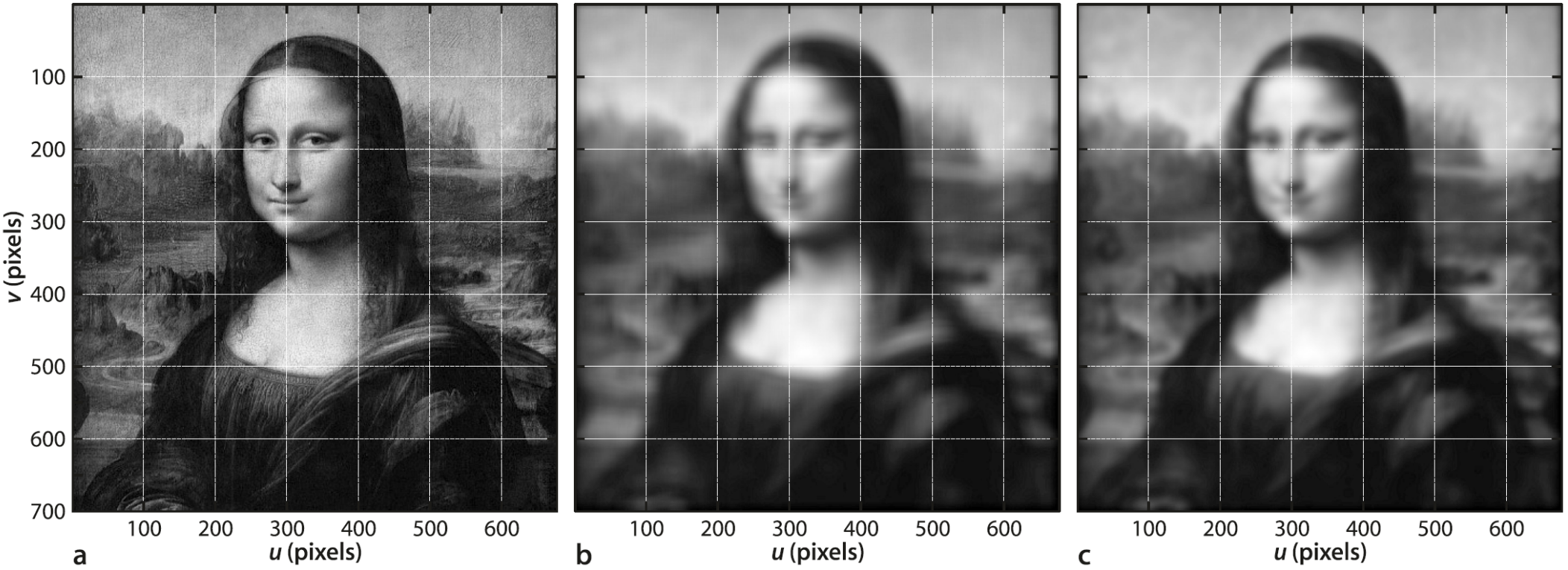


Fig. 12.13. Smoothing. **a** Original image; **b** smoothed with a 21×21 averaging kernel; **c** smoothed with a 31×31 Gaussian $G(\sigma = 5)$ kernel

Filtering

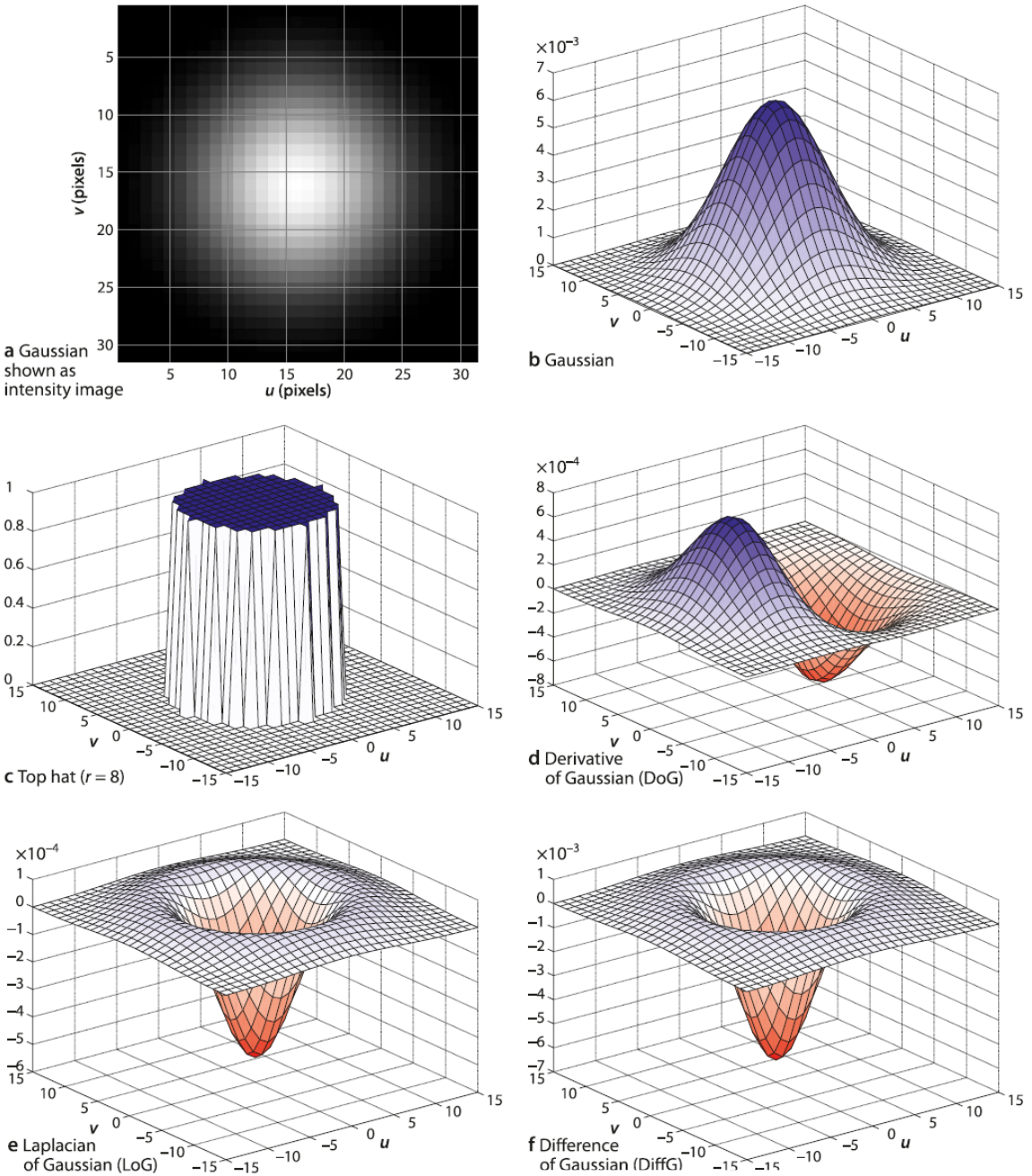
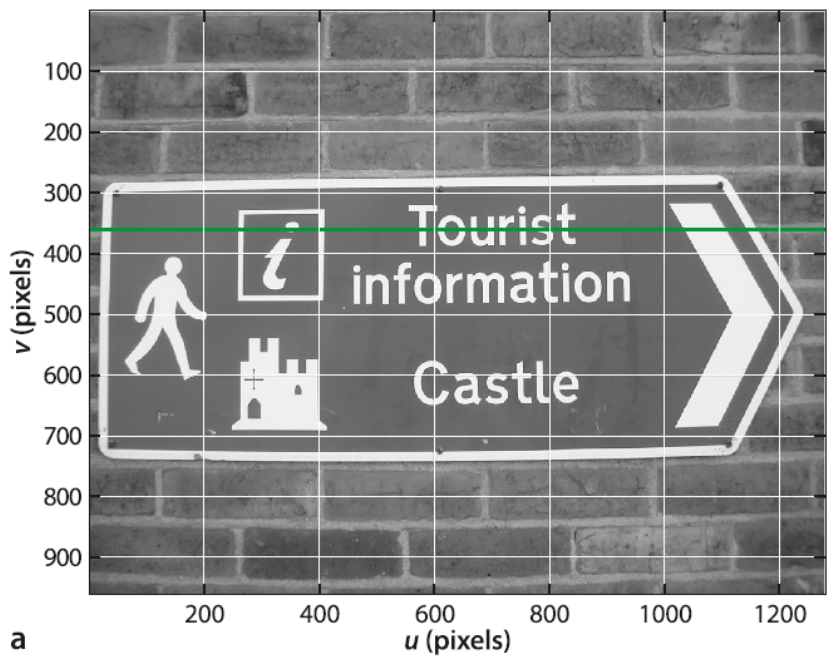
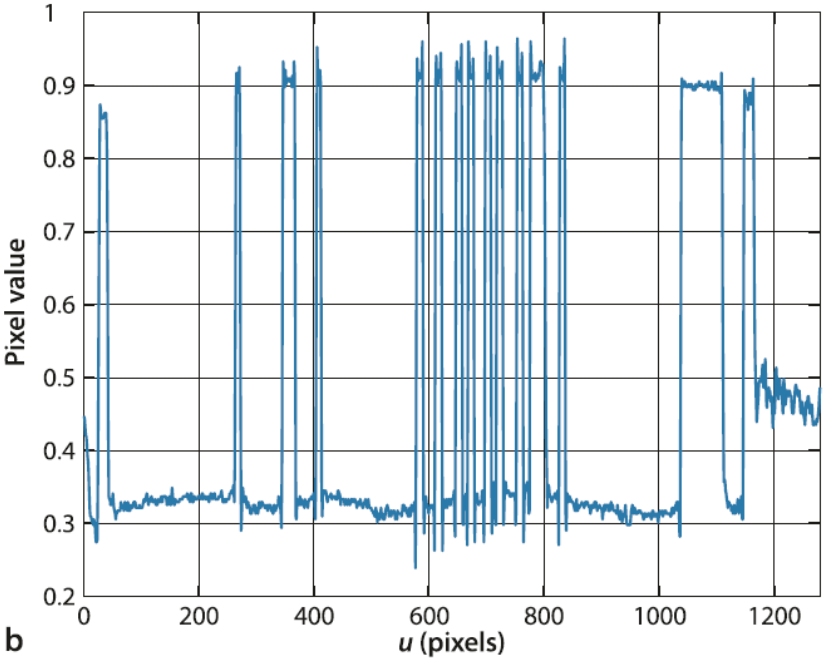
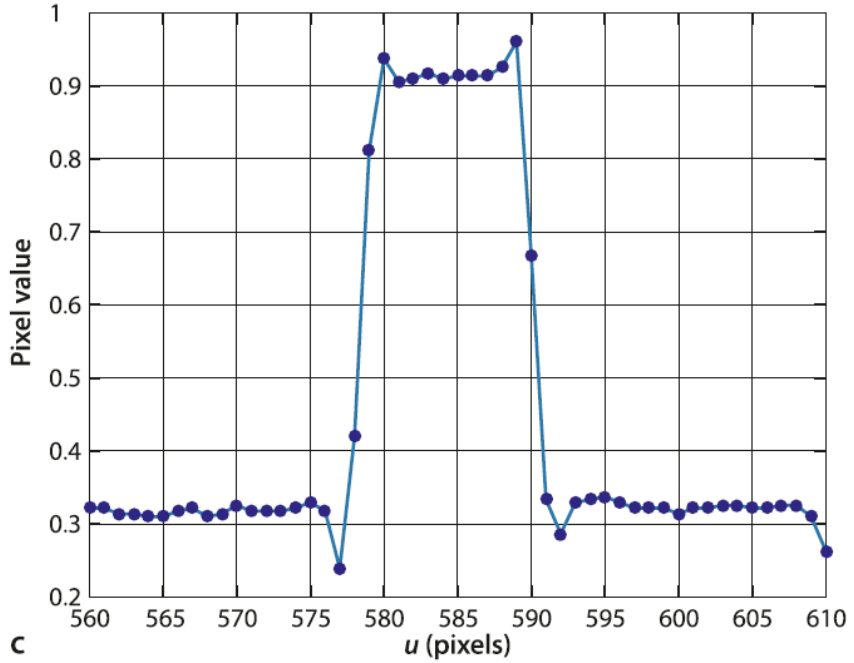


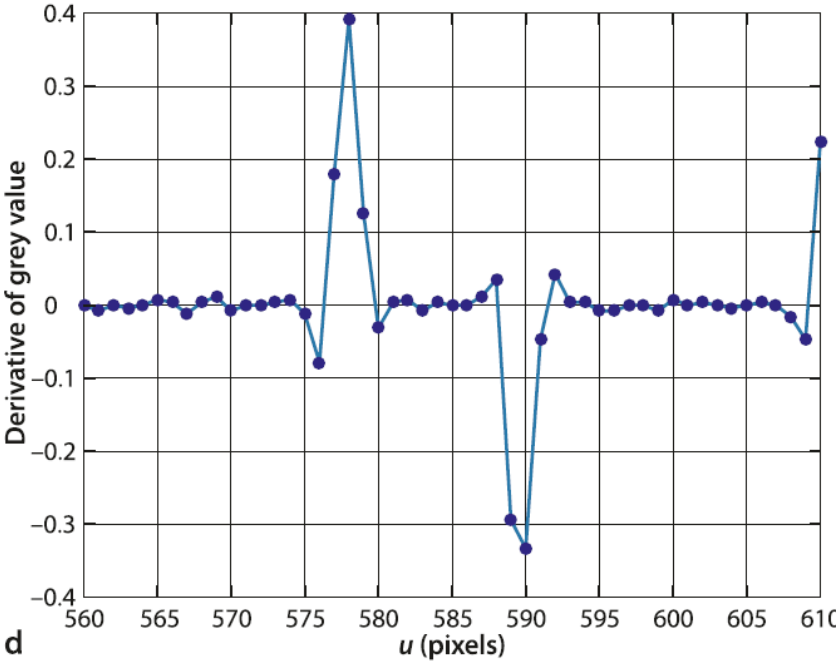
Fig. 12.14. Gallery of commonly used convolution kernels. $h = 15$,

Edge detection

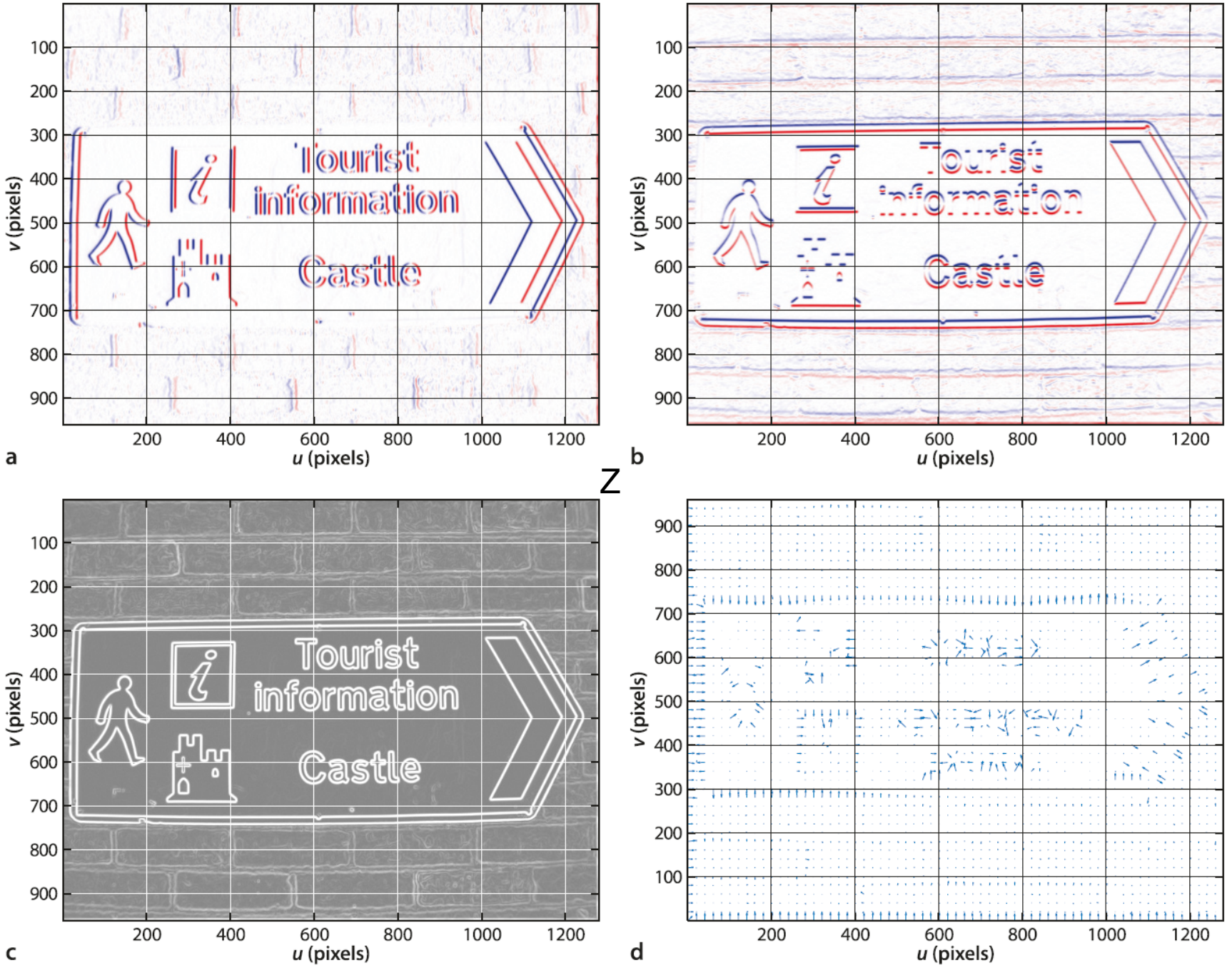








Edge detection



Edge detection

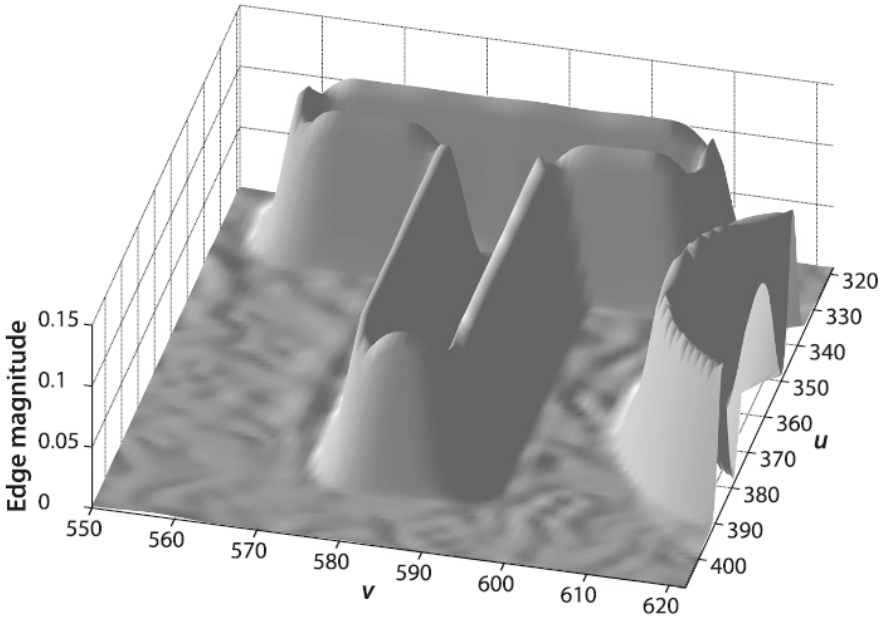


Fig. 12.18. Closeup of gradient magnitude around the letter T shown as a 3-dimensional surface

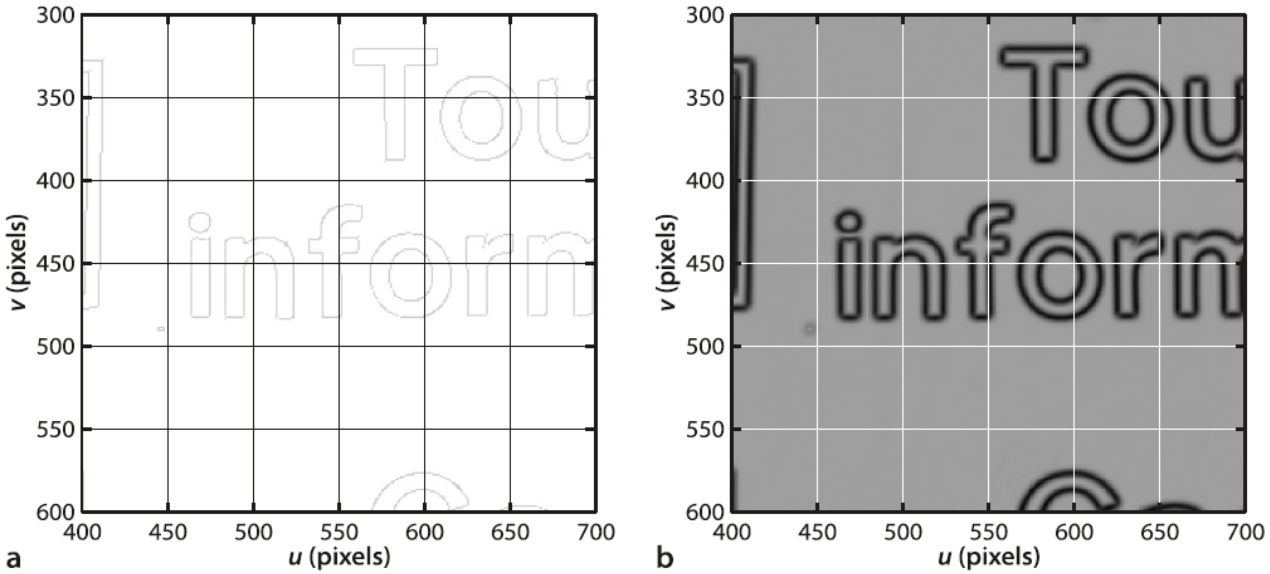


Fig. 12.19. Comparison of two edge operators: **a** Canny operator with default parameters; **b** Magnitude of derivative of Gaussian kernel ($\sigma = 2$). The |DoG| operator requires less computation than Canny but generates thicker edges. For both cases results are shown *inverted*, *white* is zero

Template matching

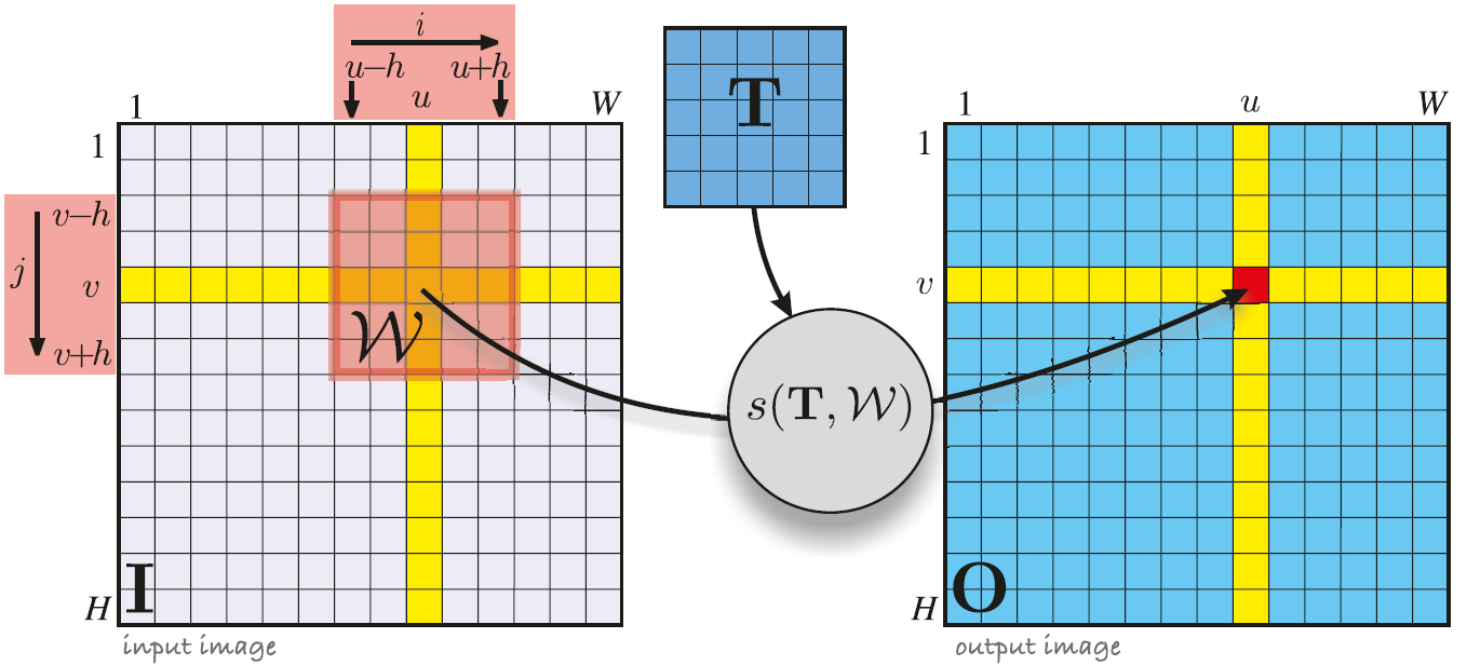


Fig. 12.21.
Spatial image processing operations. The *red shaded region* shows the window $\mathcal{W}$ that is the set of pixels used to compute the output pixel (shown in *red*)

Template matching

Table 12.1. Similarity measures for two equal sized image regions I_1 and I_2. The Z-prefix indicates that the measure accounts for the zero-offset or the difference in mean of the two images (Banks and Corke 2001). $\bar{I}_1$ and $\bar{I}_2$ are the mean of image regions I_1 and I_2 respectively. Toolbox functions are indicated in the last column

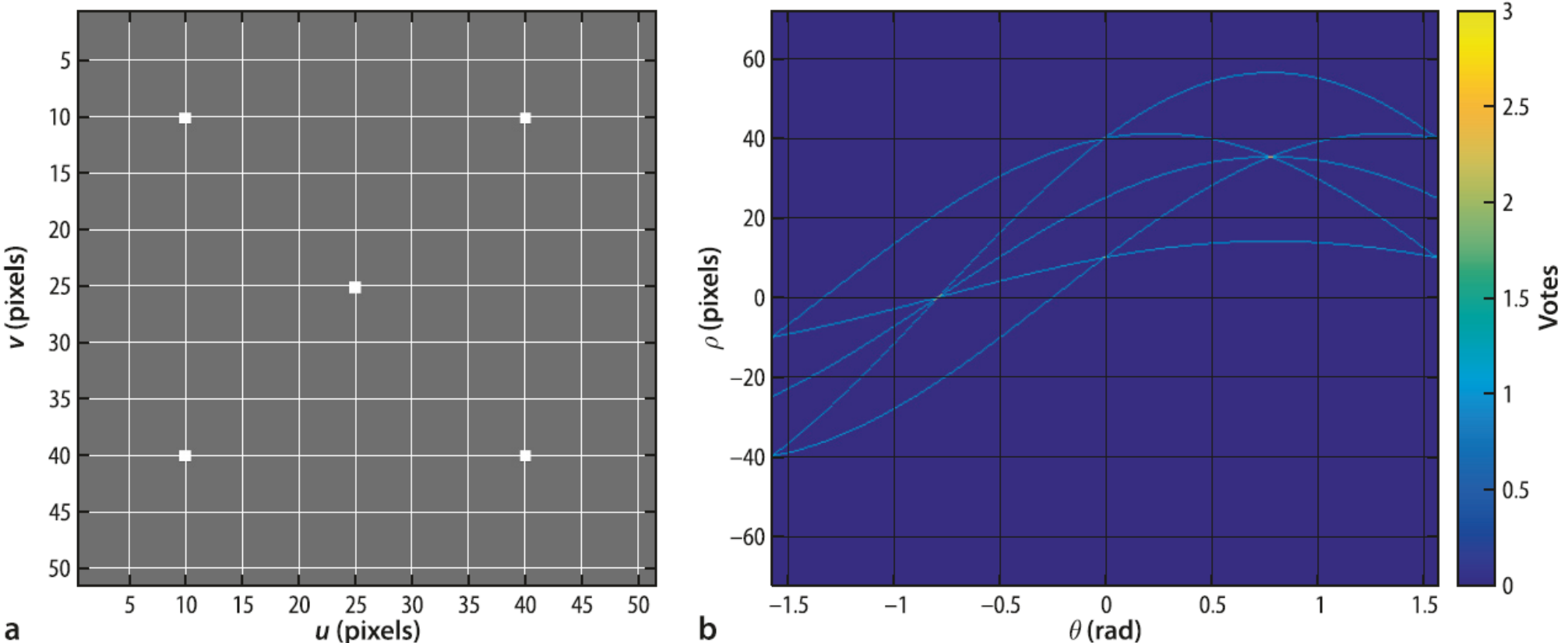
Sum of absolute differences		
SAD	$s = \sum_{(u,v)\in I_1} \left\lvert I_1[u,v] - I_2[u,v] \right\rvert$	`sad`
ZSAD	$s = \sum_{(u,v)\in I_1} \left\lvert (I_1[u,v] - \bar{I}_1) - (I_2[u,v] - \bar{I}_2) \right\rvert$	`zsad`
Sum of squared differences		
SSD	$s = \sum_{(u,v)\in I_1} \left(I_1[u,v] - I_2[u,v]\right)^2$	`ssd`
ZSSD	$s = \sum_{(u,v)\in I_1} \left((I_1[u,v] - \bar{I}_1) - (I_2[u,v] - \bar{I}_2)\right)^2$	`zssd`
Cross correlation		
NCC	$s = \frac{\sum_{(u,v)\in I_1} I_1[u,v]\ I_2[u,v]}{\sqrt{\sum_{(u,v)\in I_1} I_1^2[u,v]\ \sum_{(u,v)\in I_1} I_2^2[u,v]}}$	`ncc`
ZNCC	$s = \frac{\sum_{(u,v)\in I_1} (I_1[u,v] - \bar{I}_1)\ (I_2[u,v] - \bar{I}_2)}{\sqrt{\sum_{(u,v)\in I_1} (I_1[u,v] - \bar{I}_1)^2\ \sum_{(u,v)\in I_1} (I_2[u,v] - \bar{I}_2)^2}}$	`zncc`

Hough transform

Consider any one of these points – there are an infinite number of lines that pass through that point. If the point could vote for these lines, then each possible line passing through the point would receive one vote. Now consider another point that does the same thing, casting a vote for all the possible lines that pass through it. One line (the line that both points lie on) will receive a vote from each point – a total of two votes – while all the other possible lines receive either zero or one vote.

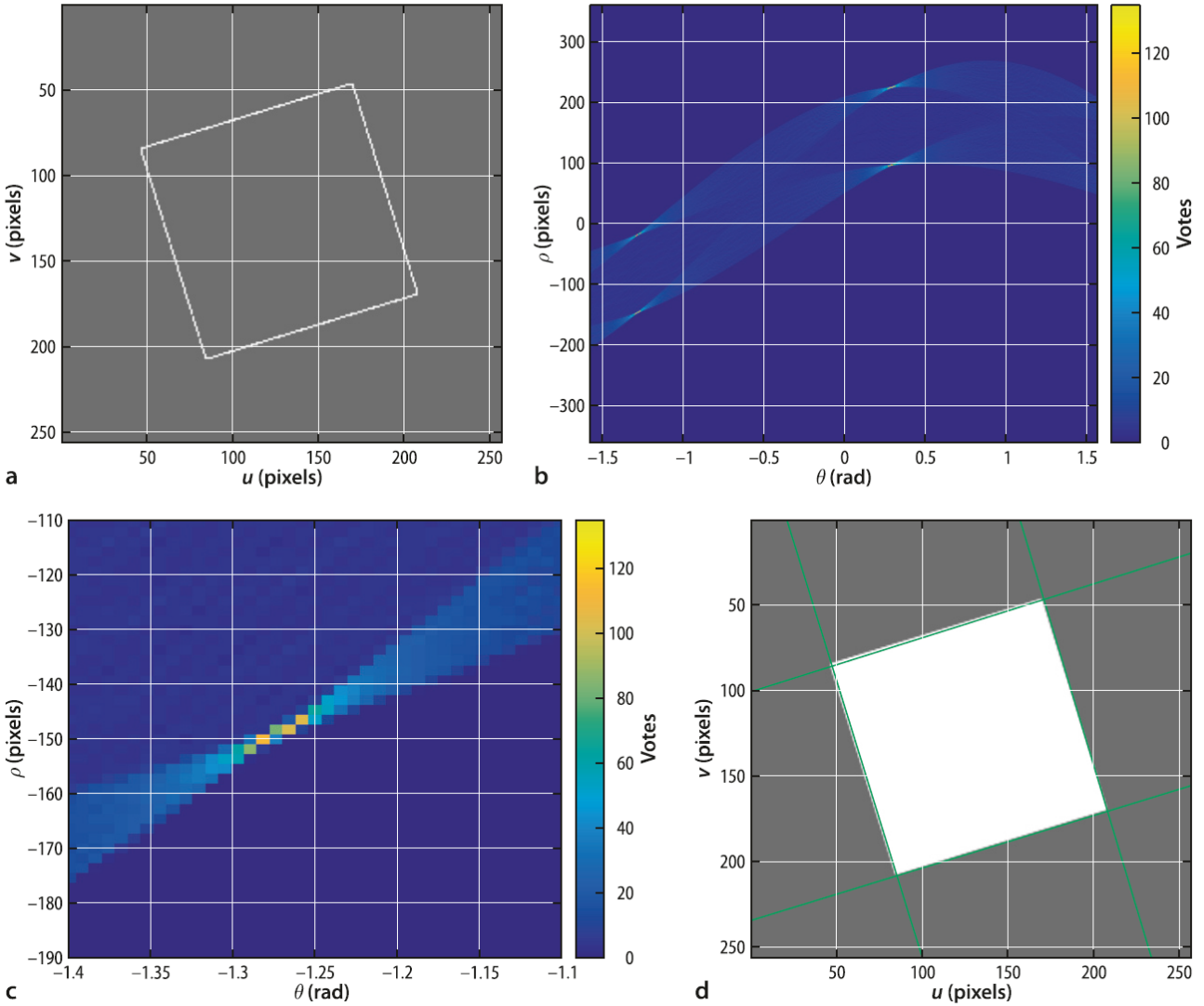
At the end of the process those elements of A with the largest number of votes correspond to dominant lines in the scene.

Fig. 13.17. Hough transform fundamentals. **a** Five points that define six lines; **b** the Hough accumulator array. The horizontal axis is an angle $\theta \in \mathbb{S}^1$ so we can imagine the graph wrapped around a cylinder and the left- and right-hand edges joined. The sign of ρ also changes at the *join* so the curve intersections on the left- and right-hand edges are equivalent



Hough transform

Fig. 13.19. Hough transform for a rotated square. **a** Edge image; **b** Hough accumulator; **c** closeup view of the Hough accumulator; **d** estimated lines overlaid on the original image



Hough transform

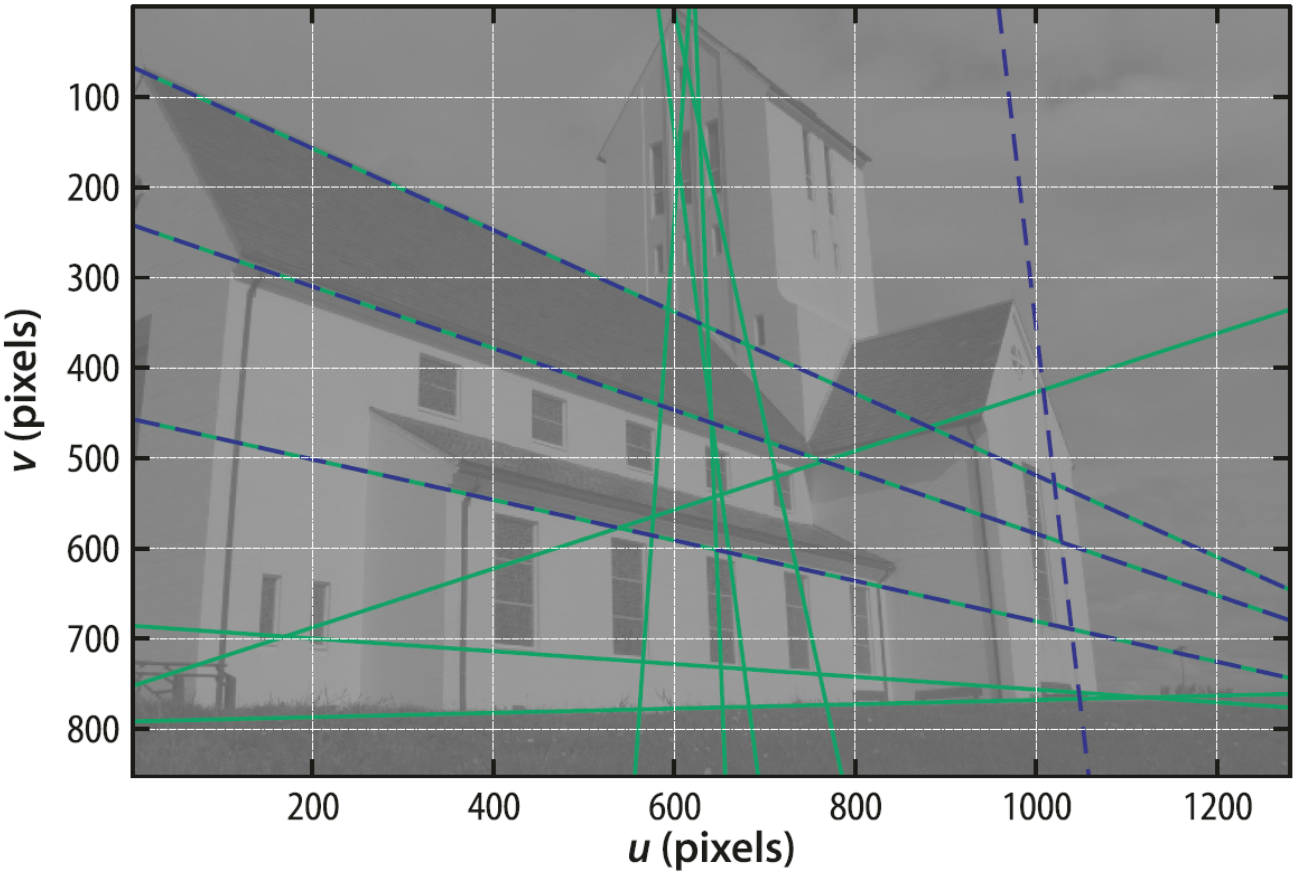


Fig. 13.20.
Hough transform of a real image. The *green lines* correspond to the ten strongest voting peaks. The overlaid *dashed blue lines* are those with an edge segment length of at least 80 pixels. Three lines meet both criteria

Corner detection

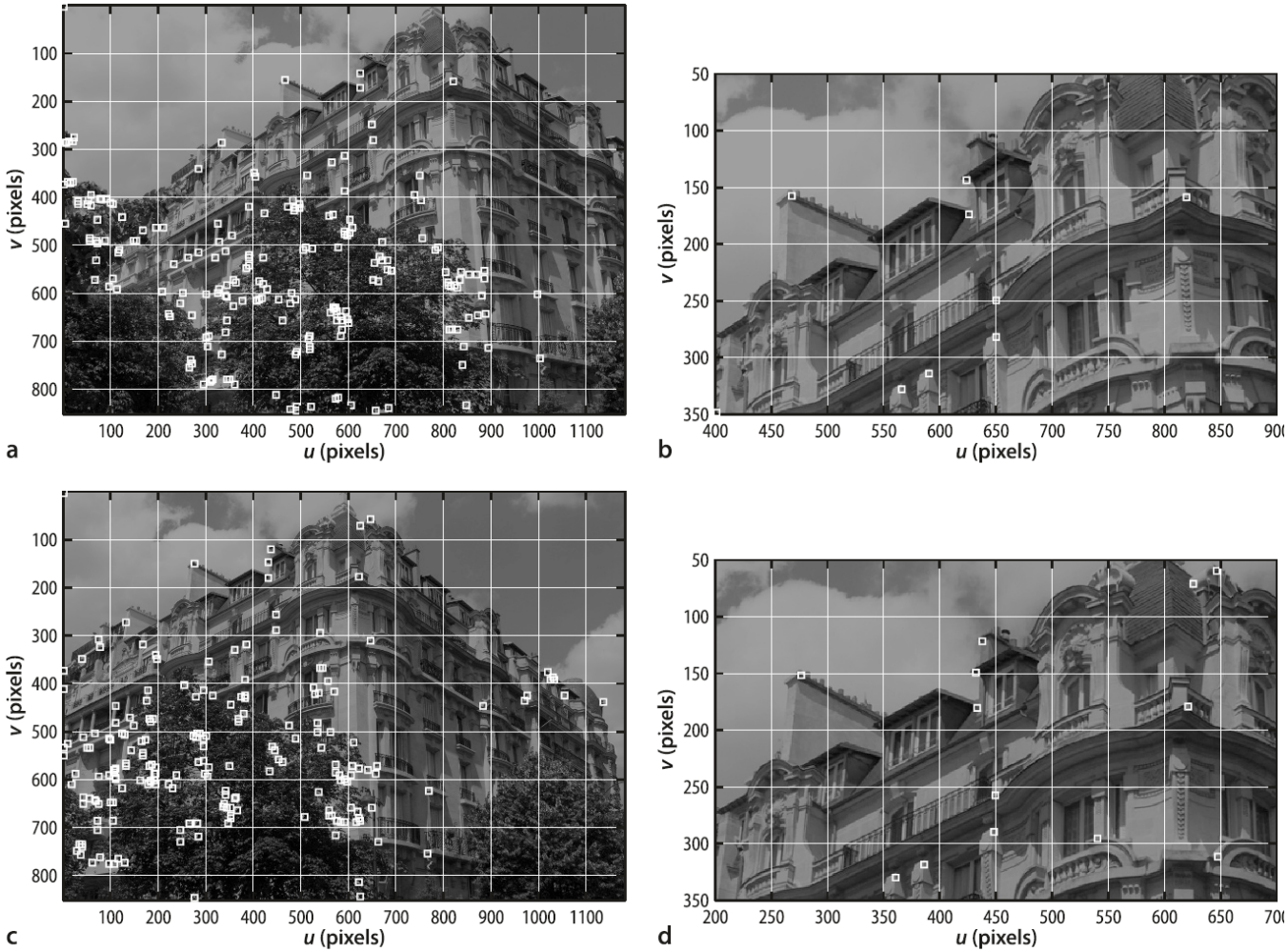


Fig. 13.21. Harris corner detector applied to two views of the same building. **a** View one; **b** zoomed in view one; **c** view two; **d** zoomed in view two. Notice that quite a number of the detected corners are attached to the same world features in the two views

Corner detection

An interest point (u, v) is one for which $s(\cdot)$ is high for *all* directions of the vector (δ_u, δ_v). That is, in whatever direction we move the window it rapidly becomes dissimilar to the original region. If we consider the original image $\boldsymbol{I}$ as a surface the eigenvalues of $\boldsymbol{A}$ are the principal curvatures of the surface at that point. If both eigenvalues are small then the surface is flat, that is the image region has approximately constant local intensity. If one eigenvalue is high and the other low, then the surface is ridge shaped which indicates an edge. If both eigenvalues are high the surface is sharply peaked which we consider to be a corner.◀

$$s\left(u, v, \delta_u, \delta_v\right) = \sum_{(i,j)\in\mathcal{W}} \left(\boldsymbol{I}\left[u+\delta_u+i, v+\delta_v+j\right] - \boldsymbol{I}\left[u+i, v+j\right]\right)^2$$

Corner detection

The Shi-Tomasi detector considers the strength of the corner, or *cornerness*, as the minimum eigenvalue

$$C_{\mathrm{ST}}(u, v) = \min(\lambda_1, \lambda_2) \tag{13.15}$$

where λ_i are the eigenvalues of $\boldsymbol{A}$. Points in the image for which this measure is high are referred to as "*good features to track*". The Harris detector◀ is based on this same insight but defines corner strength as

$$C_{\mathrm{H}}(u, v) = \det(\boldsymbol{A}) - k\,\mathrm{tr}(\boldsymbol{A}) \tag{13.16}$$

and again a large value represents a strong, distinct, corner. Since $\det(\boldsymbol{A}) = \lambda_1\lambda_2$ and $\mathrm{tr}(\boldsymbol{A}) = \lambda_1 + \lambda_2$ the Harris detector responds when both eigenvalues are large and elegantly avoids computing the eigenvalues of $\boldsymbol{A}$ which has a somewhat higher computational cost.◀ A commonly used value for k is 0.04.

Feature descriptors

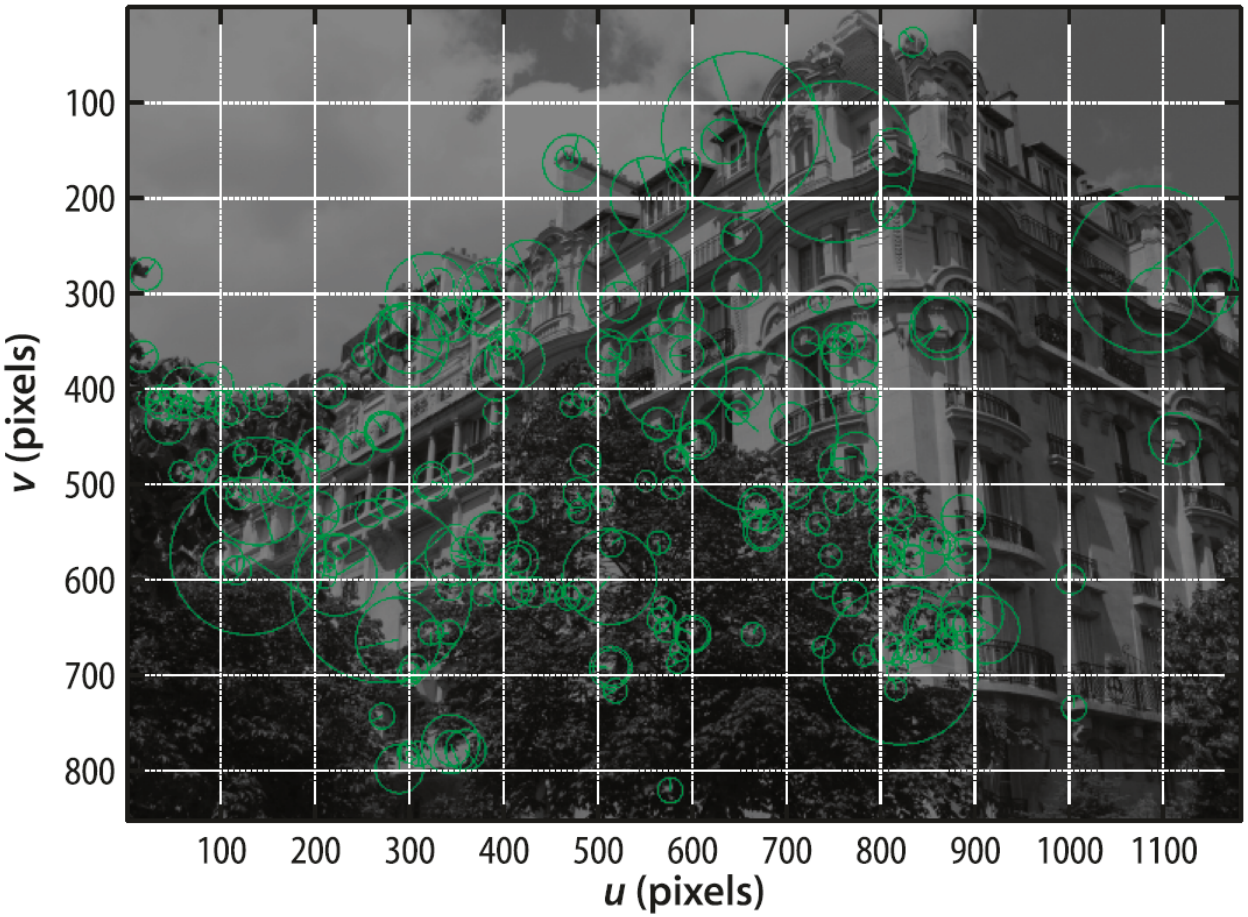


Fig. 13.28.
SURF descriptors showing the support region (scale) and orientation as a radial line

Feature descriptors

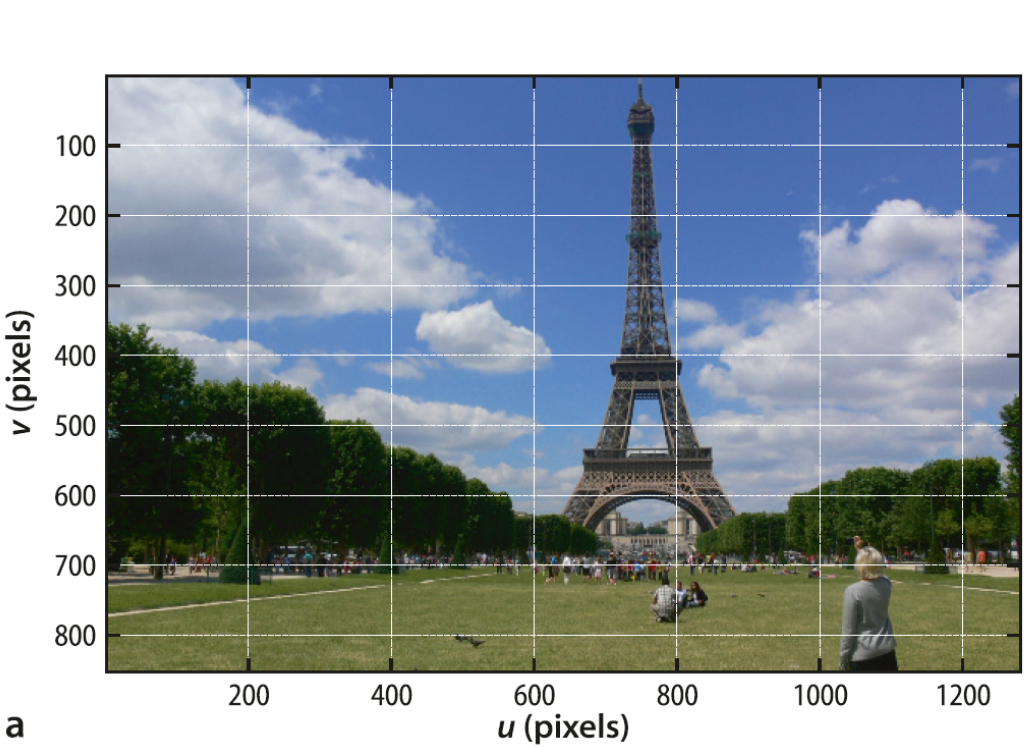
The SURF algorithm is more than just a scale-invariant feature detector, it also computes a very robust *descriptor*. The descriptor is a 64-element vector that encodes the image gradient in subregions of the support region in a way which is invariant to brightness, scale and rotation. This enables feature descriptors to be unambiguously matched to a descriptor of the same world point in another image even if their scale and orientation are quite different. The difference in position, scale and orientation of the matched features gives some indication of the relative camera motion between the two views. Matching features between scenes is crucial to the problems that we will address in the next chapter.

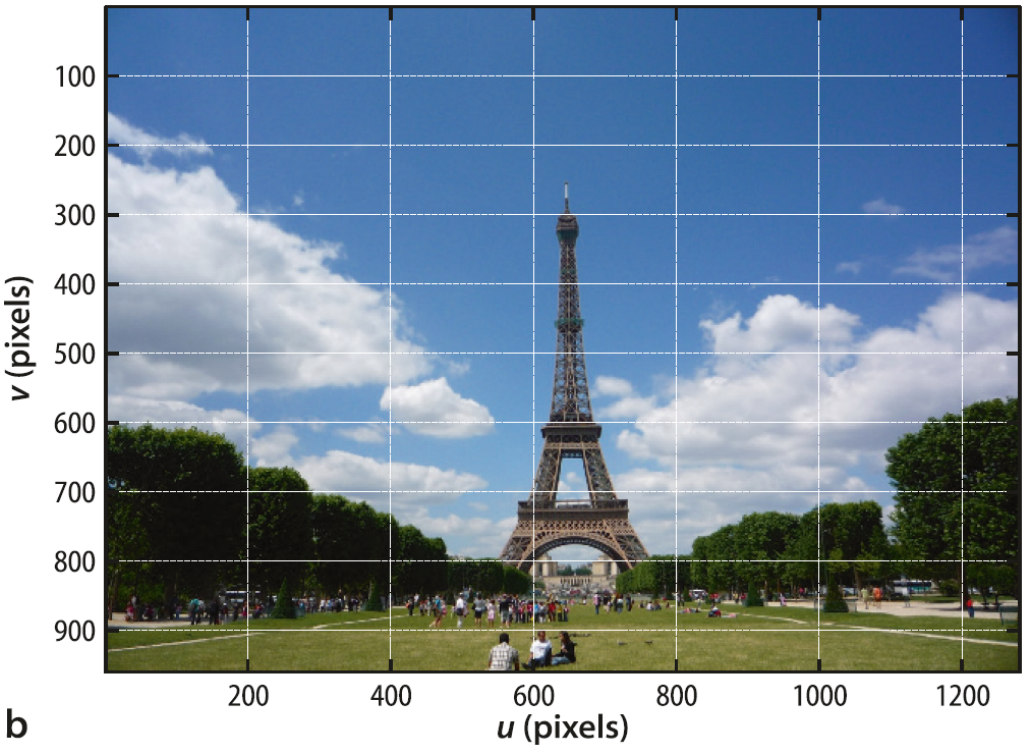
Feature descriptors

The SURF algorithm is more than just a scale-invariant feature detector, it also computes a very robust *descriptor*. The descriptor is a 64-element vector that encodes the image gradient in subregions of the support region in a way which is invariant to brightness, scale and rotation. This enables feature descriptors to be unambiguously matched to a descriptor of the same world point in another image even if their scale and orientation are quite different. The difference in position, scale and orientation of the matched features gives some indication of the relative camera motion between the two views. Matching features between scenes is crucial to the problems that we will address in the next chapter.

Detectors versus descriptors. When matching world feature points, or landmarks, between different views we must first *find* points that are distinctive. This is the job of the detector and results in a coordinate (u, v) and perhaps a scale factor or orientation. The second task is to *describe* the region around the point in a way that allows it to be matched as decisively as possible with the region around the corresponding point in the other view. This is the descriptor which is typically a long vector formed from pixel values, histograms, gradients, histograms of gradient and so on. There are many detectors to choose from: Harris and variants, Shi-Tomasi, FAST, AGAST, MSER etc.; as well as many descriptors: ORB, BRISK, FREAK, CenSurE (aka STAR), HOG, ACF etc. Some algorithms such as SIFT and SURF define both a detector and a descriptor. The SIFT descriptor is a form of HOG descriptor.

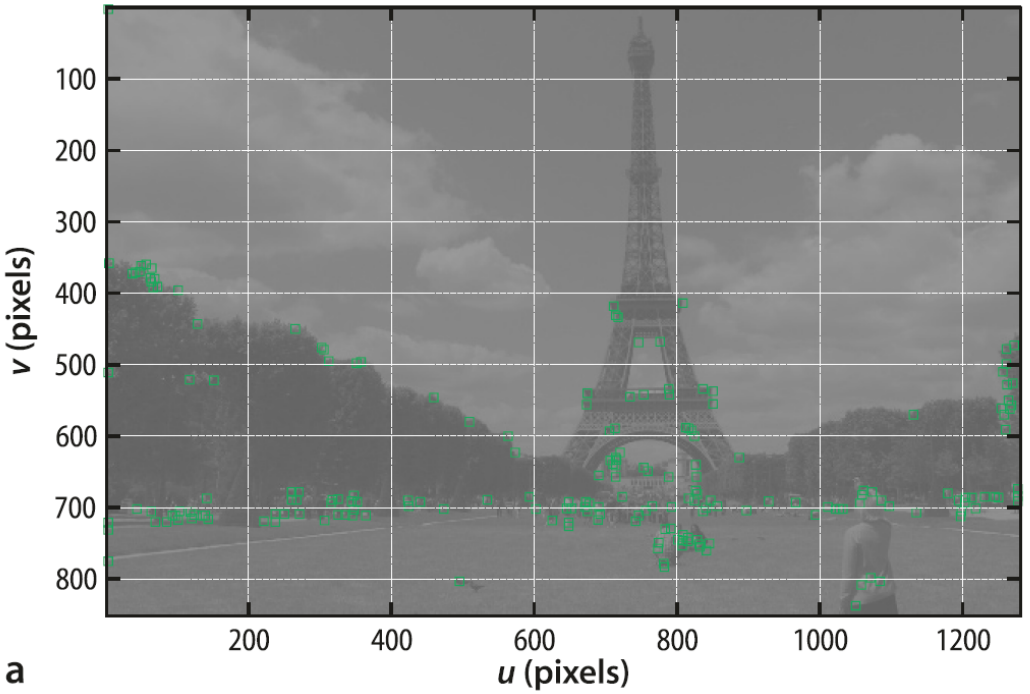
Feature matching

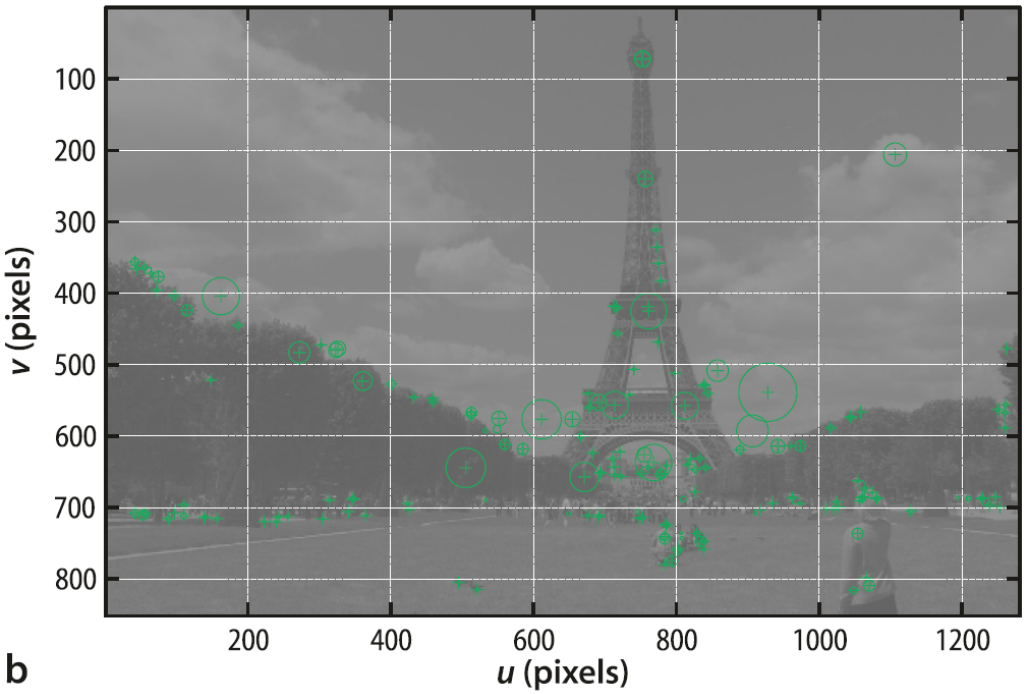




Feature matching

Fig. 14.2. Corner features computed for Fig. 14.1a. **a** Harris corner features; **b** SURF corner features showing scale





Feature matching

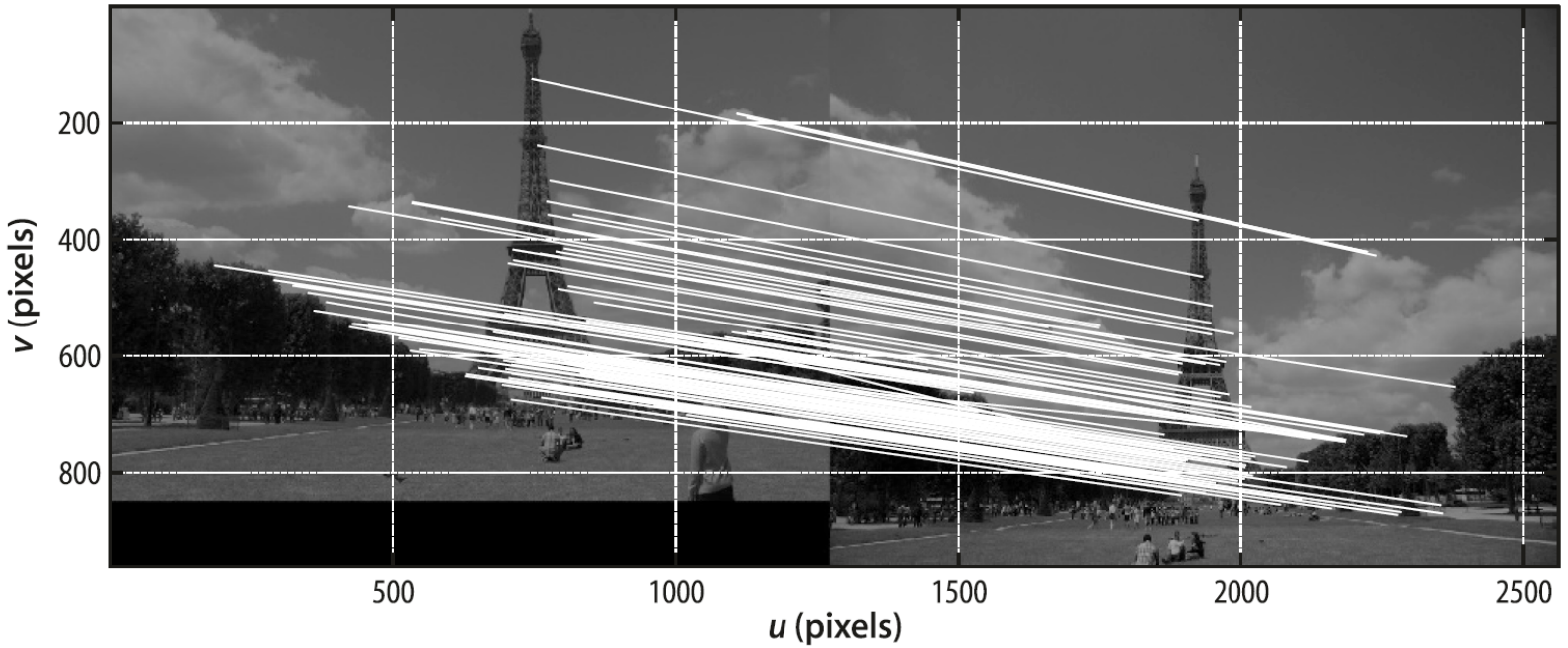


Fig. 14.3.
Feature matching. Subset (100 out of 1 664) of matches based on SURF descriptor similarity. We note that a few are clearly incorrect

For more

Read RVC Part IV

See videos on https://robotacademy.net.au

Feedback

Piazza thread: 3/23 Lec 16 Feedback

Please post your answers to the following anonymously.

1. What did you like today?

2. What was unclear?

3. How was spring break?

4. What did you think of the ethics lecture on Monday?

5. Any additional feedback / comments?