Artificial Intelligence

Lecture 16: An overview of image processing and machine vision methods

Dr. Benjamin Inden

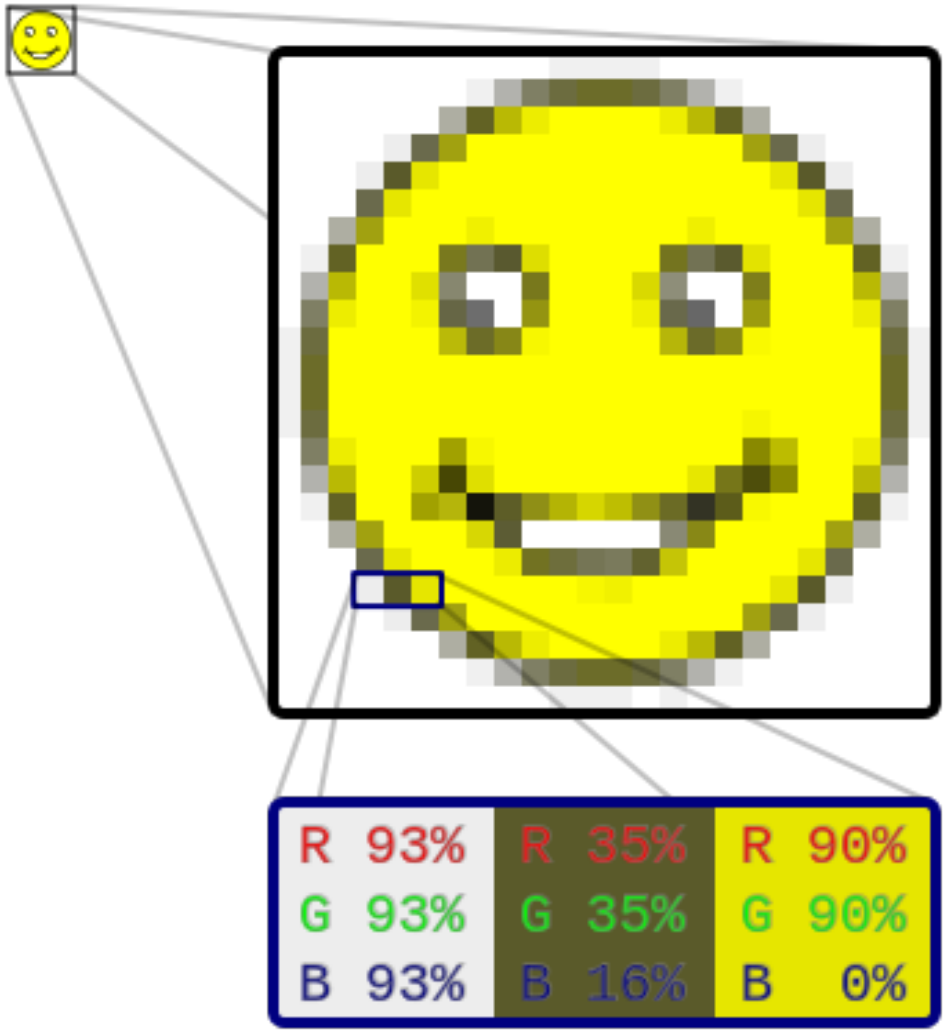
Department of Computing & Technology, Nottingham Trent University

September 28, 2017

Lecture for this week

- Slides based on Russel/Norvig, Chapter 24; `scikit-learn.org`; `scikit-image.org`; some images and other material from Wikipedia.
- Please do not post these slides to the internet or use other means of public distribution. They are for your personal use only.

How (raster) images are stored in the computer

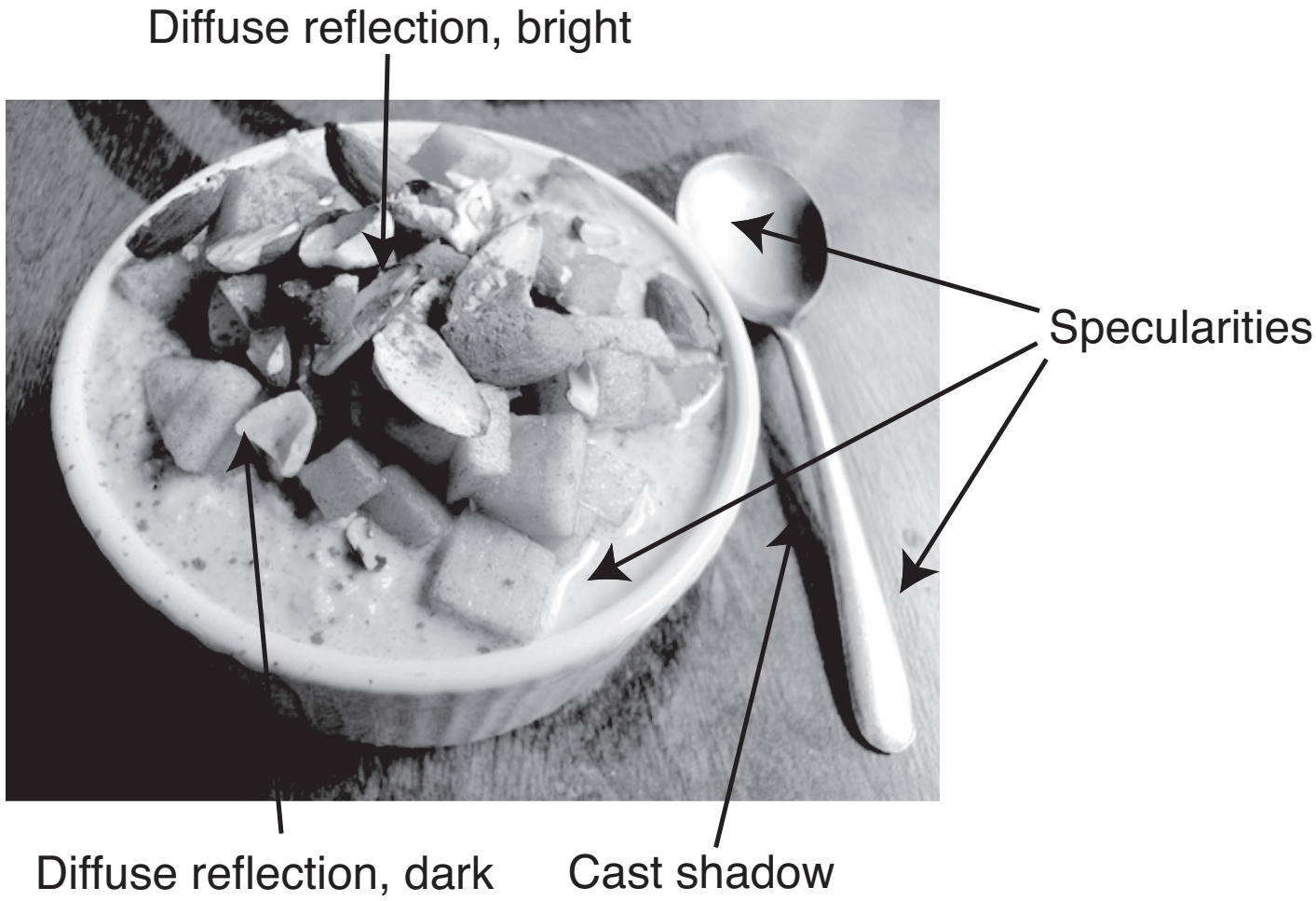


Definitions

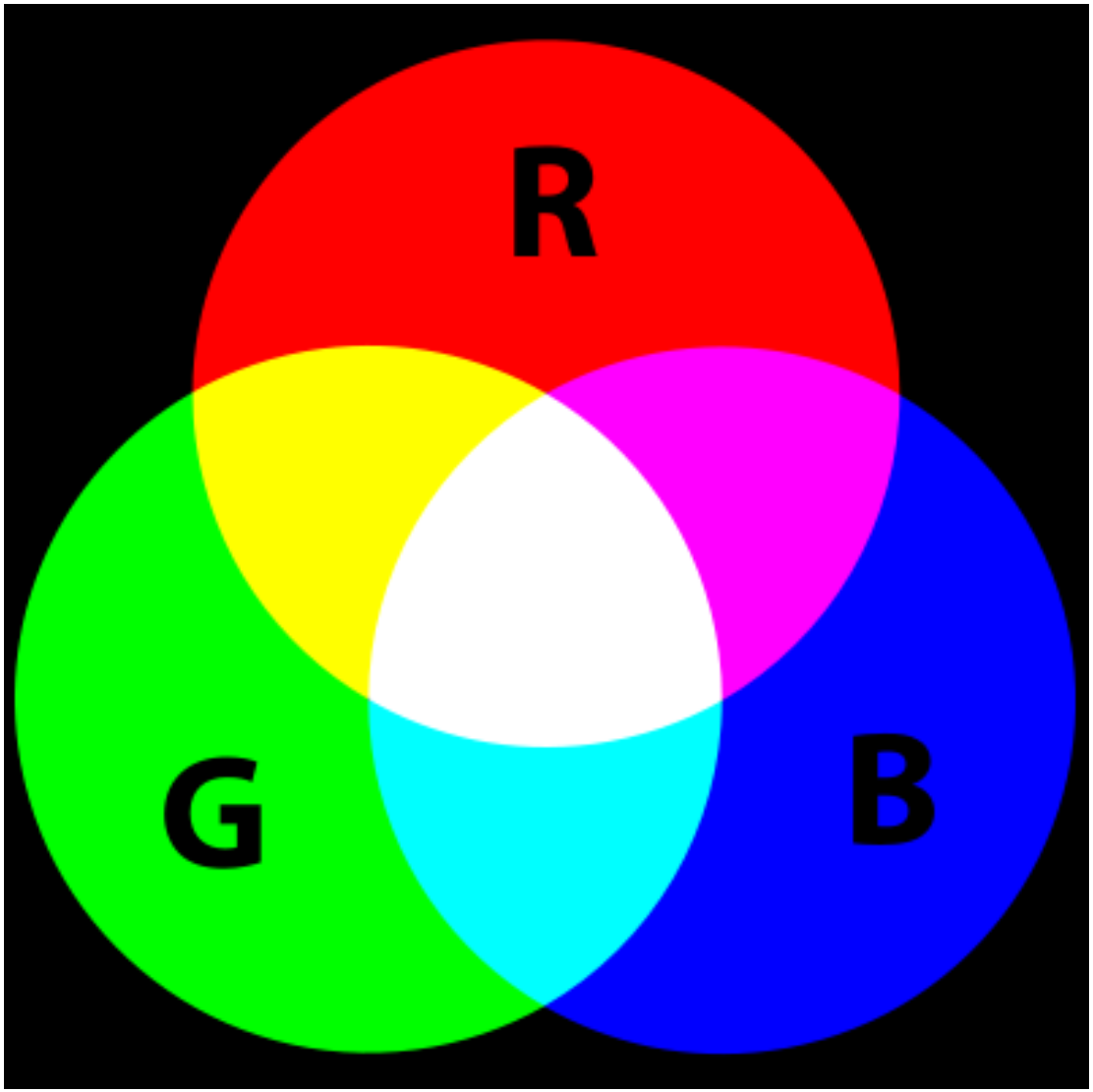
Pixel: an image element at given coordinates (x, y)

Color depth: number of bits used to store the color of a single pixel, expressed as bit per pixel or bit per color (there may be several colors per pixel ...)

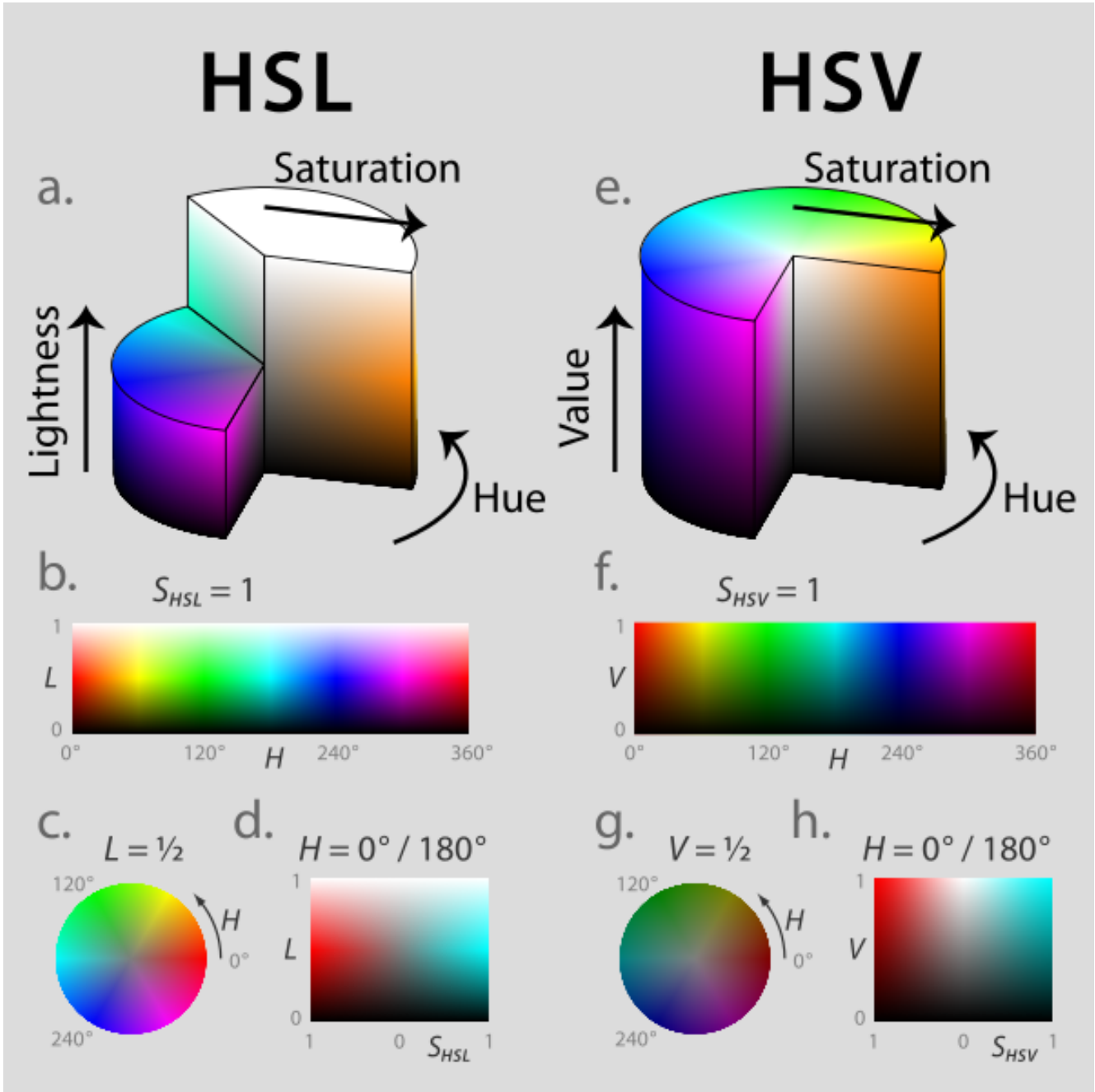
Challenges to recognising objects: Illumination



Color models: RGB



Color models: HSL and HSV



Advantages of HSV model for machine vision

- RGB values typically all depend on lighting, whereas H value is rather independent on lighting.
- “Look for the orange book”

Image preprocessing

- Object recognition can be done with supervised machine learning
- To make recognition easier, the images can be preprocessed:
  - Changing color space
  - Removing Colors
  - Filtering out noise
  - Focussing on relevant parts of the image
  - Decreasing resolution

Convolution

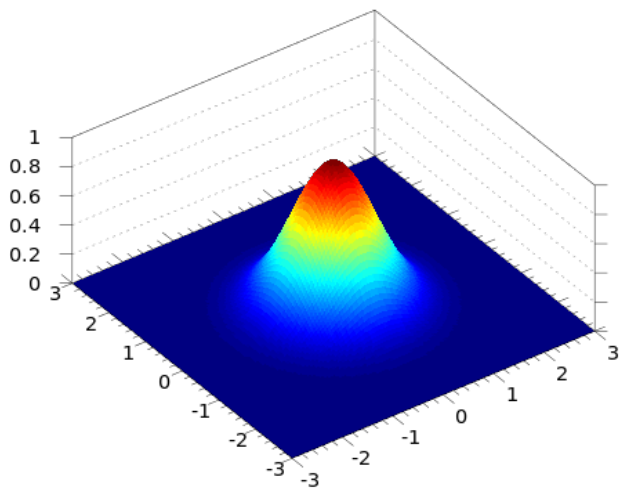
The convolution of two (discrete) functions f, g in one dimension:
$h(x) = (f * g)(x) = \sum_{u=-\infty}^{+\infty} f(u)g(x-u)$
In two dimensions:
$h(x,y) = (f * g)(x,y) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} f(u,v)g(x-u, y-v)$
For applications in image processing, f is the input image, h is the output image, and g is a kernel function used to transform he image

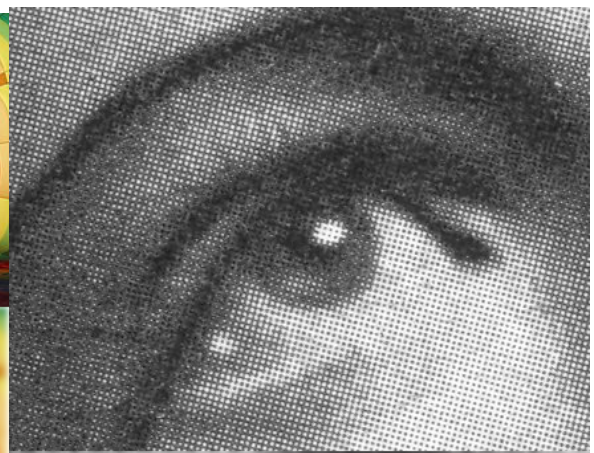
Blurring

Intuitively, take a (weighted) mean of pixels in the neighborhood to remove pixel noise

More formally, apply convolution with a Gaussian kernel

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



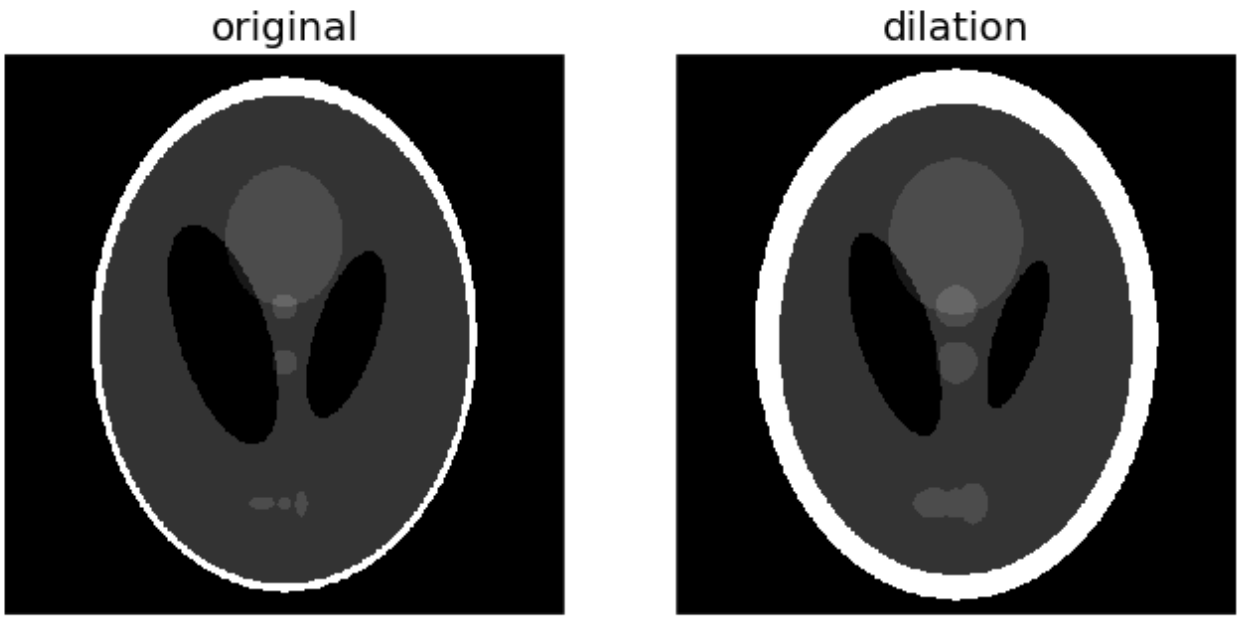


Erosion

Sets a pixel to the minimum of all pixels in its neighborhood

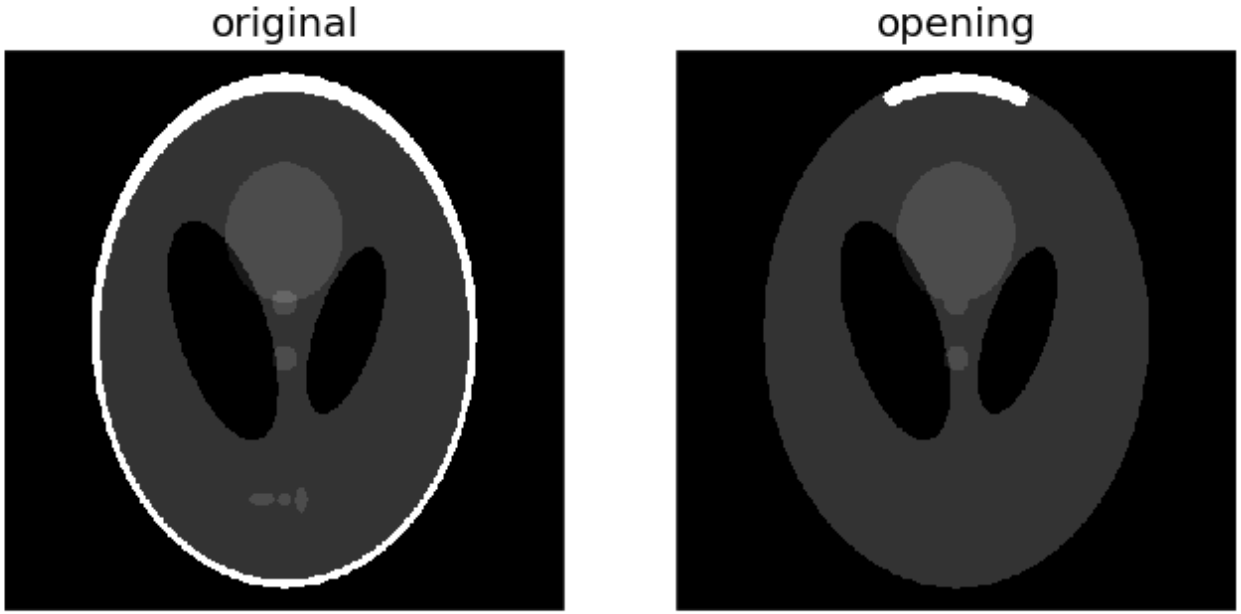
Dilation

Sets a pixel to the maximum of all pixels in its neighborhood



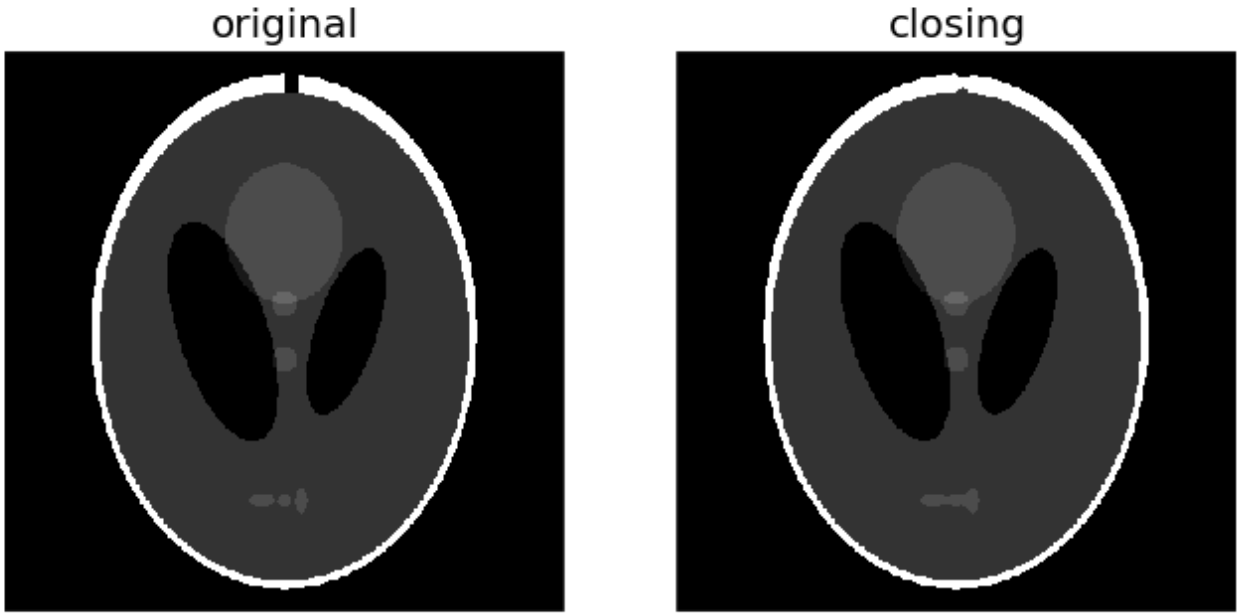
Opening

Applies an erosion followed by a dilation
Can remove small bright spots



Closing

Applies a dilation followed by an erosion
Can remove small dark spots



Cropping and Scaling

Original

Cropping

Scaling

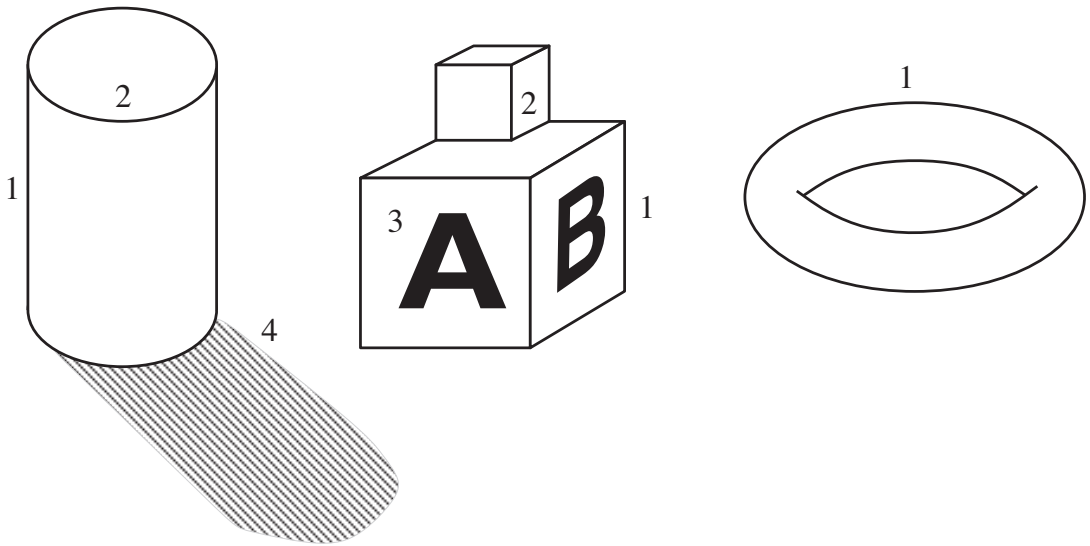
Feature detection / extraction

A feature is an "interesting part" of an image. It is characterized by a local pattern.
Typical features:

- Edges
- Corners
- Blobs (Regions of uniformity, e.g. uniform color)

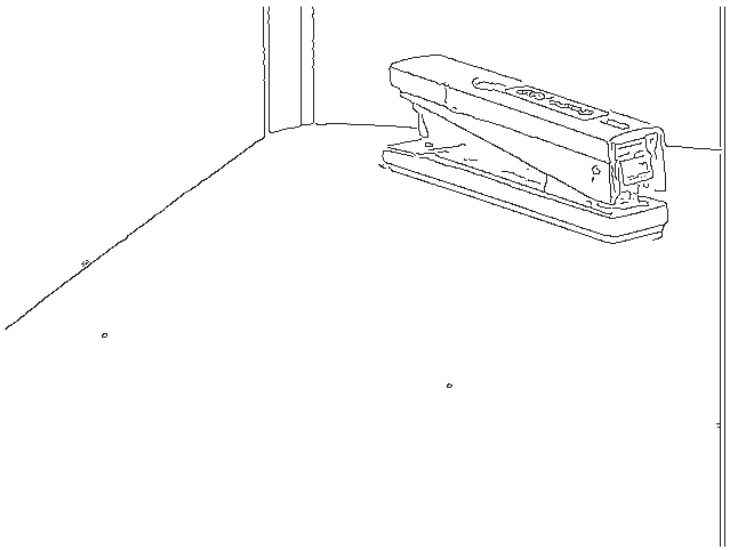
Various algorithms are used by computer vision libraries like OpenCV and scikit-image to detect various features. Features may be organized hierarcically (e.g., a face has a nose, a mouth ...). Previous knowledge can be used to narrow down the possibilities. Feature detection can serve as a preparatory step before object recognition, or may be a goal in itself.

Edge detection: Different kind of edges

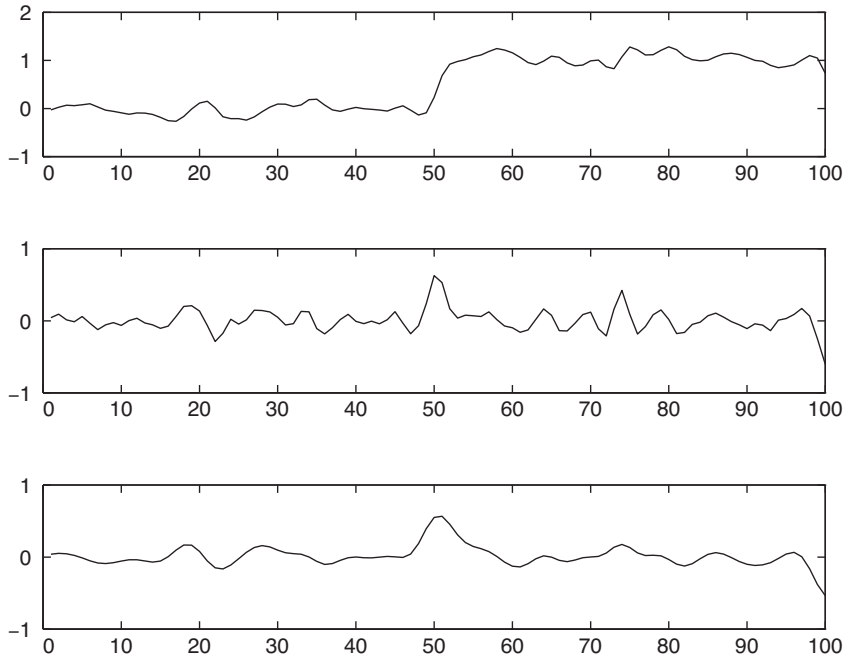


(1) depth discontinuities; (2) surface orientation discontinuities; (3) reflectance discontinuities; (4) illumination discontinuities (shadows).

Edge detection: Typical results



Edge detection: The intensity profile

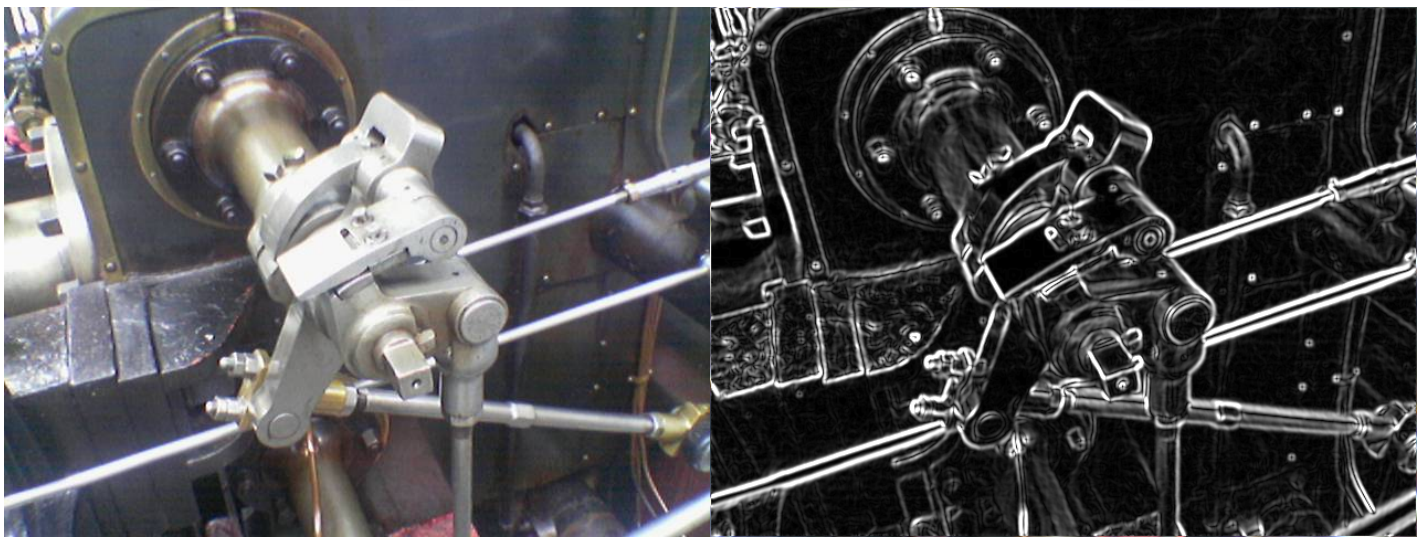


(1) image brightness along a spatial dimension; (2) differentiated brightness curve; (3) derivative of a smoothed version (i.e., after Gaussian blurring) of the image brightness function

Sobel edge detection

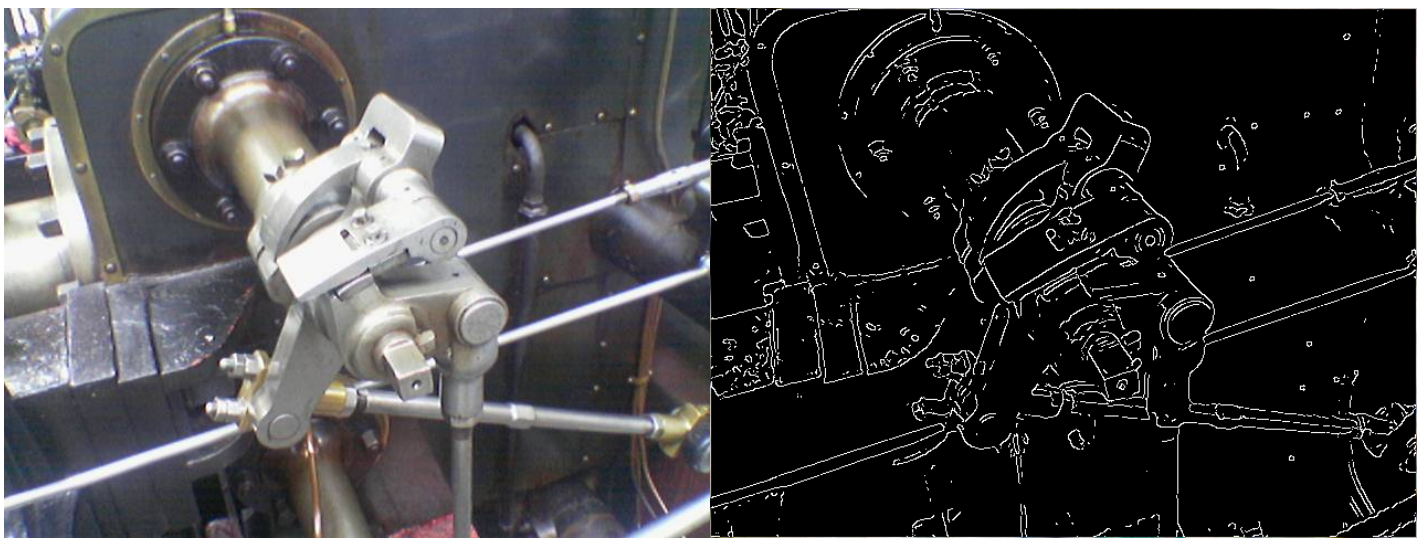
Image is convolved with two kernels that approximate blurring and differentiation in x and y dimensions

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$



Canny edge detector

1. Apply Gaussian filter to remove noise.
2. Calculate intensity gradients of the image using two kernels, discretize edge orientation to vertical, horizontal, or diagonal.
3. Apply non-maximum suppression to “thin” the edges.
4. Apply double threshold to determine whether edge points are strong, weak, or spurious, supress spurios edge points.
5. Suppress all the weak edge points that are not connected to strong edge points.



Face detection

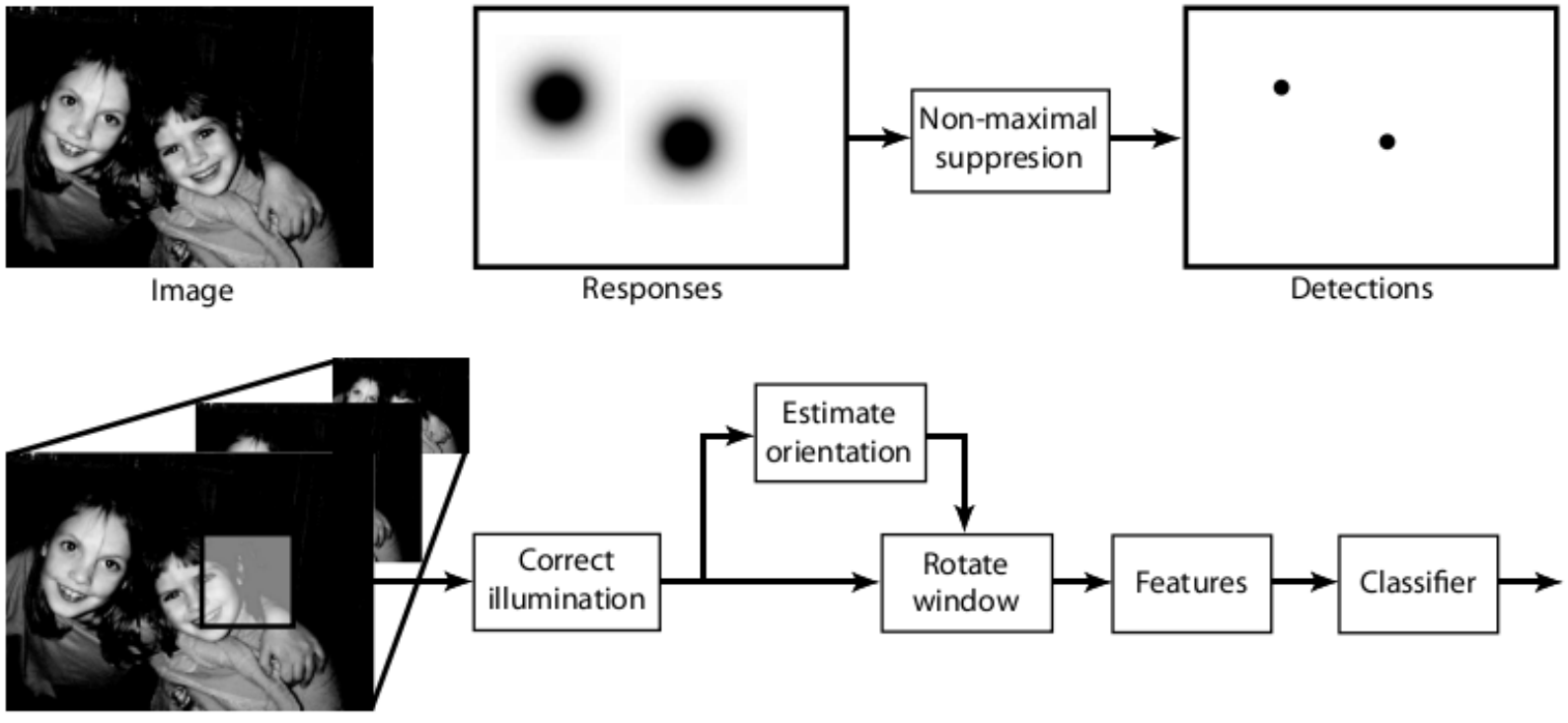


Figure 12 Face finding systems vary, but most follow the architecture illustrated in two parts here. On the top, we go from images to responses, then apply non-maximum suppression to find the strongest local response. The responses are obtained by the process illustrated on the bottom. We sweep a window of fixed size over larger and smaller versions of the image, so as to find smaller or larger faces, respectively. The illumination in the window is corrected, and then a regression engine (quite often, a neural net) predicts the orientation of the face. The window is corrected to this orientation and then presented to a classifier. Classifier outputs are then postprocessed to ensure that only one face is placed at each location in the image.

Motion detection

Typical process:

- Subtract an image from its predecessor.
- Binarize image (black/white) with a threshold set such that small changes (noise) are filtered out
- Calculate total /mean change (= brightness) over the whole image

Optical flow

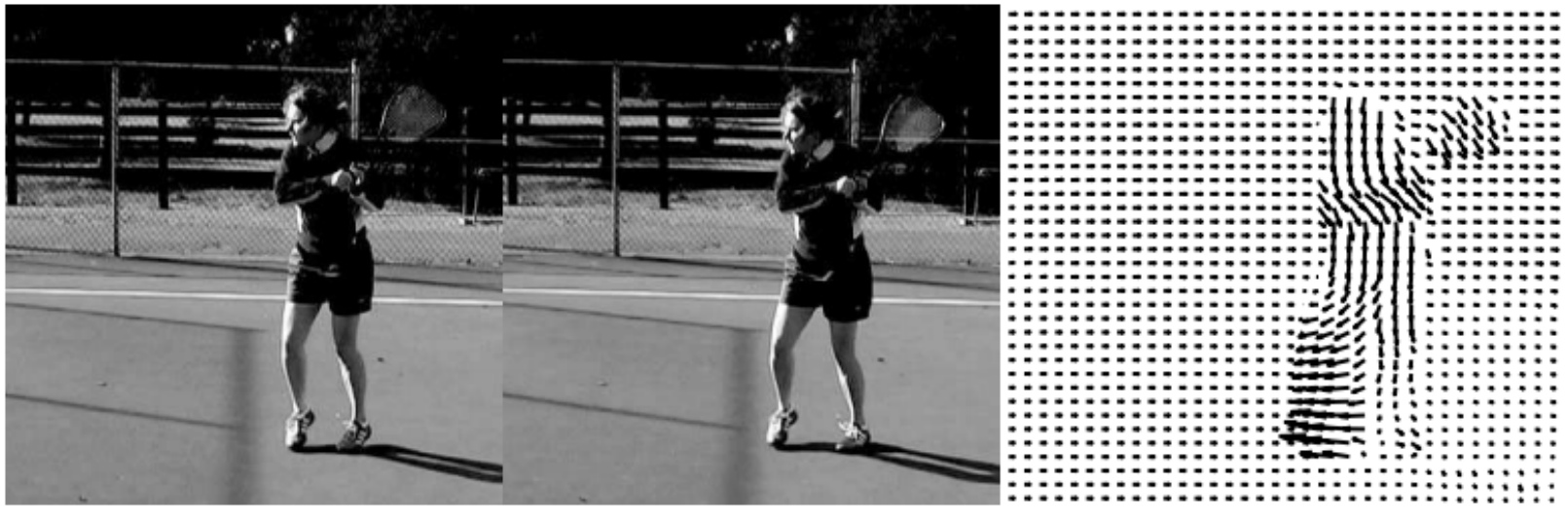
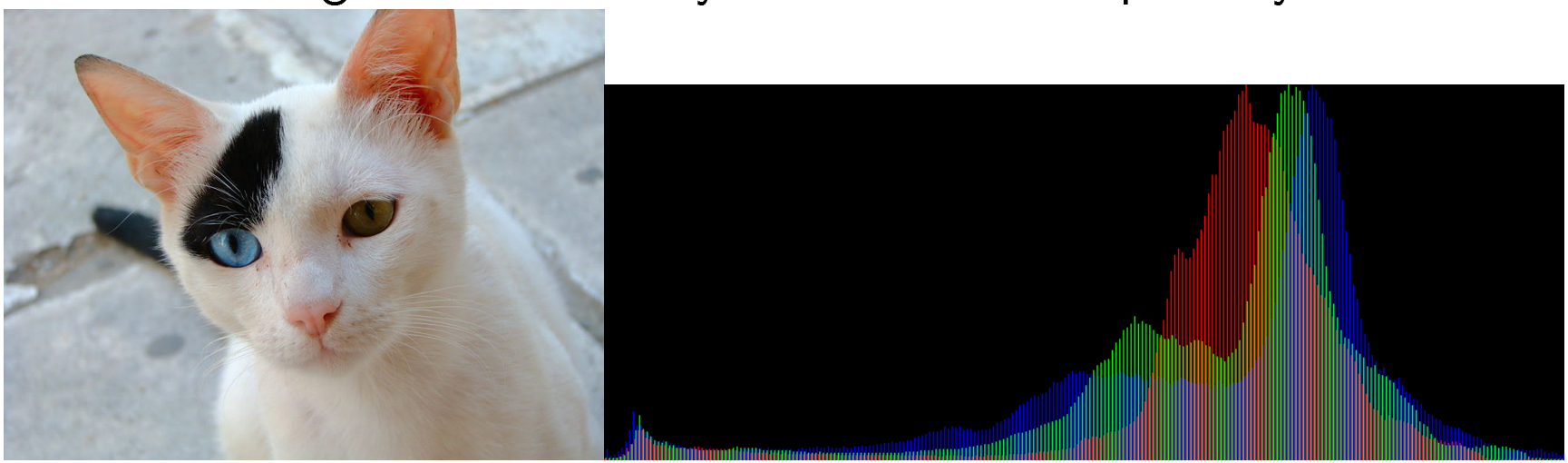


Figure 10 Two frames of a video sequence. On the right is the optical flow field corresponding to the displacement from one frame to the other. Note how the movement of the tennis racket and the front leg is captured by the directions of the arrows. (Courtesy of Thomas Brox.)

Algorithms for calculating optical flow between two images are built into libraries like OpenCV (e.g. Lucas-Kanade algorithm).
Optical flow provides also a cheap mechanism for obstacle avoidance and time-to-contact calculation in flying insects (and flying robots).

Image histograms

Image histograms show the distribution of brightness values over the whole image — if necessary, for each color separately.



Histogram Equalization and stretching for contrast enhancement

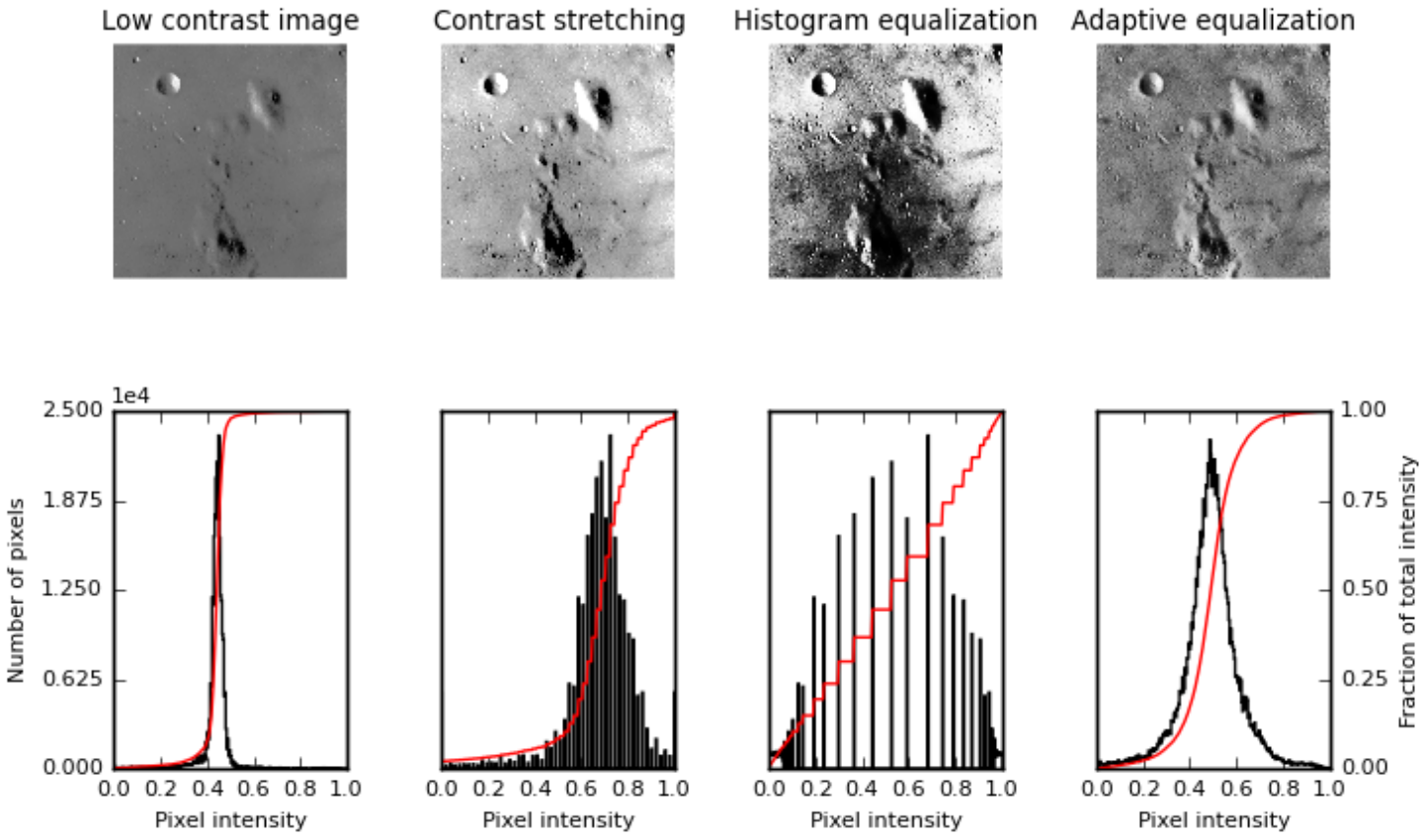
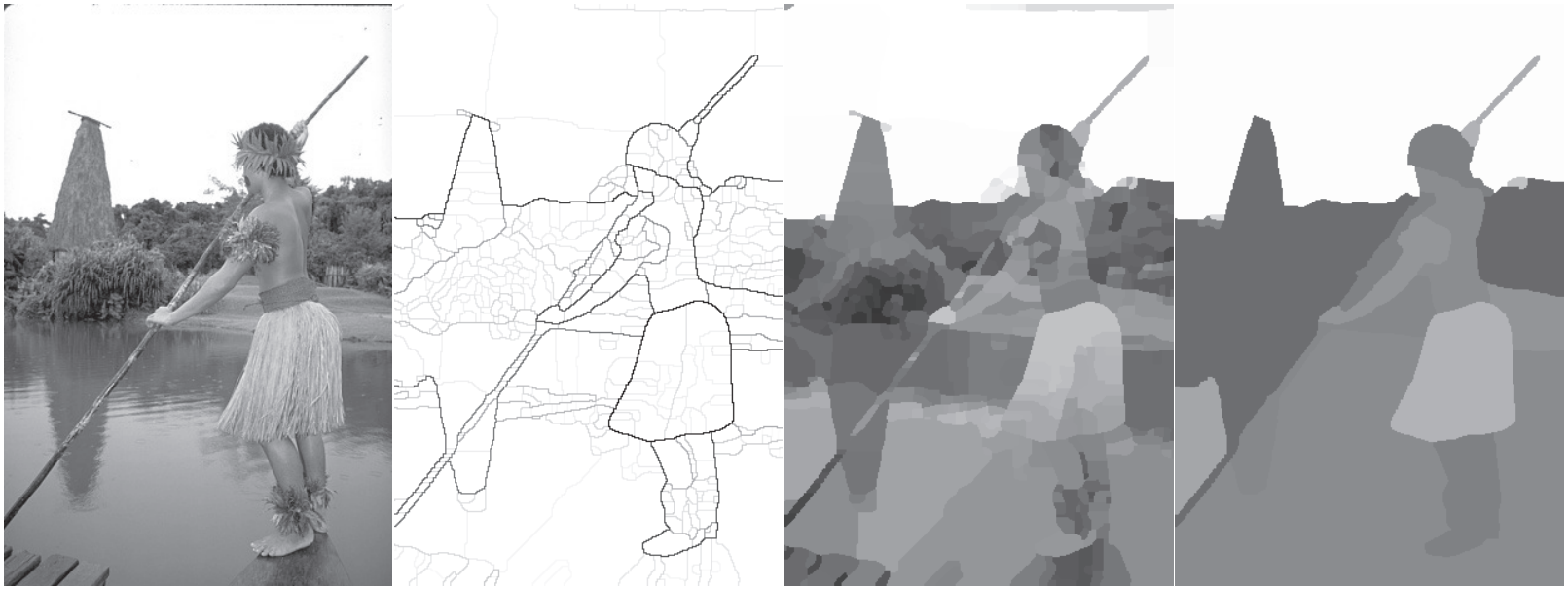


Image segmentation: Example



(a) original mage; (b) edges with different strengths; (c) fine segmented image; (d) coarse segmented image

Image segmentation: Methods

- k-means clustering based:
  1. Pick k cluster centers, either randomly or based on some heuristic
  2. Repeat until clusters are converged:
     - 2.1 Assign each pixel in the image to the cluster that minimizes the distance between the pixel and the cluster center (where distance is a measure based on combinations of color difference, spatial distance, etc.)
     - 2.2 Re-compute the cluster centers by averaging all of the pixels in the cluster
- Edge detection based (edges must be completed to become boundaries of closed regions)
- Histogram based (basically, look for valleys in the histogram and define them as borders between segments with different colours / brightnesses)
- Various other approaches, hybrid methods

scikit-image: Felzenzwalb's method, SLIC, Quickshift

Steps towards classification

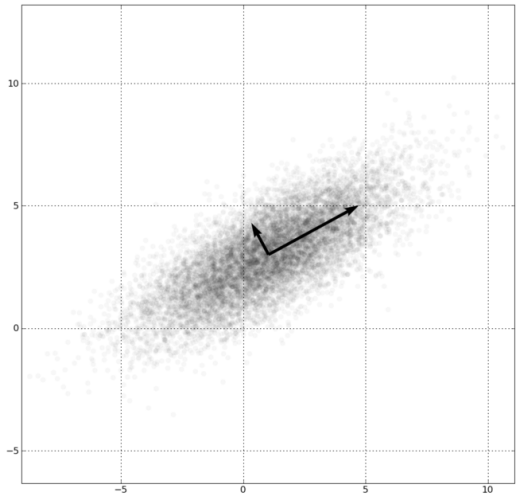
1. Apply one or more operators to get rid of unwanted information in the image:
   - 1.1 Filtering (e.g., blurring)
   - 1.2 Cropping / Decrease of resolution
   - 1.3 Feature detection / extraction
2. Apply dimension reduction to compactify the remaining information
3. Apply a supervised learning method (SVM, $k-$nearest neighbor, neural network) to the resulting input

While steps 1 and 2 are not necessary in principle, they are necessary in practice in most situations
Typical problems: recognizing a face, recognizing a car plate, recognizing a dangerous traffic situation, recognizing various tools / parts / plants (for autonomous robots), recognizing handwritten characters, ...

Dimensionality reduction

Imagine a small image 50×50 pixels $\times 3$ colors $= 7500$ inputs!
Learning with so many inputs takes a very long time.
Typical approach in machine learning: Feature selection: Either humans specify, or the computer learns itself, which inputs are important; the others are then ignored.
But at this stage of image recognition, we have aready removed most unimportant pixels; the remaining unimportant information (and the important information) is distributed over all pixels.
We can solve this problem by transforming the image into a different space (using a different basis) such that important and less important information become separated.

Principal component analysis

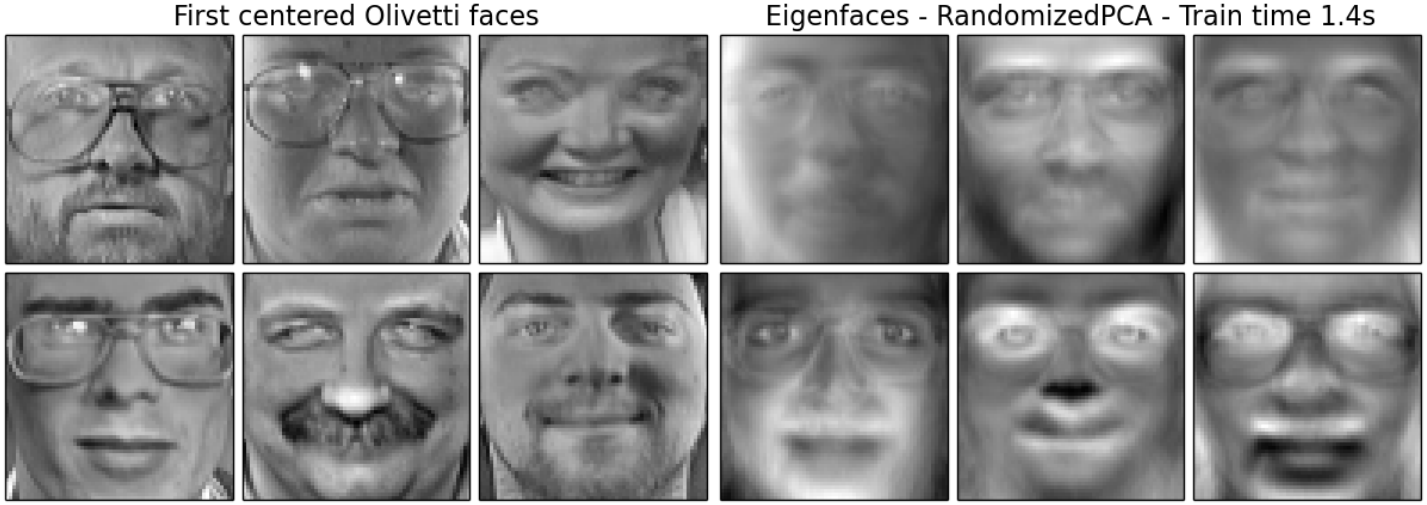


- An n-dimensional item can be understood as a point in n-dimensional vector space
- If you have a lot such items, you can re-arrange the coordinate system such that the first axis is in the direction with the most spread (variance), the second axis in the (orthogonal) direction with the second most spread, and so on
- The vectors corresponding to the new axes are called pricipal components of the data
- They can be calculated using techniques from linear algebra (an eigenvalue decomposition of the covariance matrix or a singular value decomposition of the data matrix)

Using PCA for dimensionality reduction

- The principal components are calculated.
- The data items are transformed into the new coordinate system
- Only the n components / dimensions with the greatest variance are kept, all other dimensions are ignored

Eigenfaces



- The images are preprocessed such that the faces are centered, and everything else is cropped.
- Images, like any data. can also be viewed as vectors in a high-dimensional space, and PCA can be applied.
- The resulting principal components can be displayed as images: the eigenfaces.
- All original images can now be expressed as linear combinations of the eigenfaces.
- Classification is done in the space of the coefficients of this linear combination.

Summary

- Color spaces
- Preprocessing
- Feature extraction
- Segmentation
- Dimensionality reduction
- Classification