1 The digital image

Problems of digital cameras sensors

- transmission interference
- compression artefacts
- spilling
- scratches, sensor noise
- bad contrast

1.1 Image as 2D signal

Signal: function depending on some variable with physical meaning

Image: continuous function

2 variables: xy-coordinates

3 variables: xy+time

Brightness is usually the value of the function, but other physical values are: Temperature, pressure, depth…

What is an image?

- A picture or pattern of a value varying in space and/or time
- Representation of a function $f : \mathbb{R}^n \to S$
- In digital form, e.g.:

$$I : \{1, \dots, X\} \times \{1, \dots, Y\} \to S.$$

- For greyscale CCD images, $n = 2, S = \mathbb{R}^+$

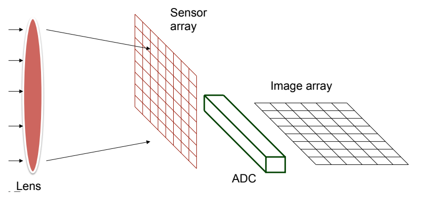
What is a pixel?

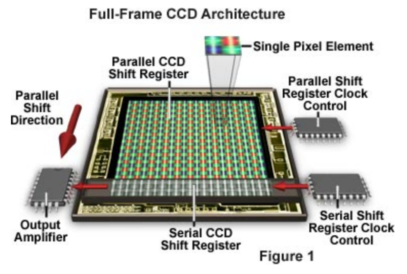
Not a little square! E.g. gaussian or cubic reconstruction filter.

1.2 Image sources

digital camera (CCD)

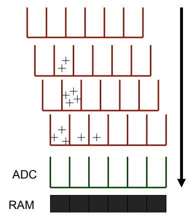
A *Charge Coupled Device* (CCD).




Full-Frame CCD Architecture
Single Pixel Element
Parallel CCD Shift Register
Parallel Shift Register Clock Control
Parallel Shift Direction
Output Amplifier
Serial CCD Shift Register
Serial Shift Register Clock Control
Figure 1

analog to digital Conversion

- The ADC measures the charge and digitizes the result.
- Conversion happens line by line.
- The charges in each photosite move down through the sensor array.



Blooming Buckets have finite capacity. Photosite saturation causes blooming.

bleeding or smearing during transit buckets still accumulate some charges. Due to tunneling and CCD Architecture. (Influenced by time “in transit” versus integration time. Effect is worse for short shutter times)

dark current CCDs produce thermally-generated charge. They give non-zero output even in darkness. Partly, this is the *dark current* and it fluctuates randomly. One can reduce it by cooling the CCD.

CMOS Has same sensor elements as CCD. Each photo sensor has its *own amplifier*. This leads to more nois (reduced by subtracting “black” image) and lower sensitivity (lower fill rate). The uses of standard CMOS technology allows to put other components on chip and “smart” pixels.

CCD vs. CMOS *CCD:* mature technology, specific technology, hight production cost, high power consumption, higher fill rate, blooming, sequential readout.
CMOS: recent technology, standard IC technology, cheap, low power, less sensitive, per pixel amplification, random pixel access, smart pixels, on chip integration with other components, rolling shutter (sequential read-out of lines)

1.3 Sampling

1D Sampling takes a function and returs a vector whose elements are values of that function at the sample points.

Undersampling “Missing” things between samples. Information lost

aliasing signals “traveling in disguise” as other frequencies. (Can happen in undersampling.)

1.4 Reconstruction

Inverse of sampling. Making samples back into continuous function. For output (need realizable method), for analysis or processing (need mathematical method), amounts to “guessing” what the function did in between.

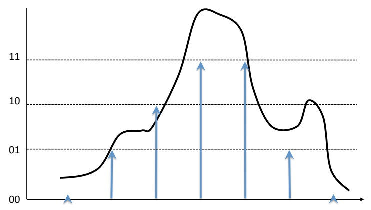
Bilinear interpolation

$$\begin{aligned} f(x,y) = {} & (1-a)(1-b)f[i,j] \\ & + a(1-b)f[i+1,j] \\ & + abf[i+1,j+1] \\ & + (1-a)bf[i,j+1] \end{aligned}$$

Nyquist frequency Half the sampling frequency of a discrete signal processing system. Signal’s max frequency (bandwidth) must be *smaller* than this.

sampling grids cartesian sampling, hexagonal sampling and non-uniform sampling

1.5 Quantization



real valued function will get digital values - integer values. Quantization is lossy and can’t be reconstructed. Simple quantization uses equally spaced levels with k intervals.

usual quantization intervals *Grayscale image:* 8 bit$= 2^8 = 256$ grayvalues. *Color image RGB (3 channels):* 8 bit/channel $= 2^{24} =$ 16.7M colors. Nonlinear, for example log-scale.

1.6 Image Properties

Image resolution: Clipped when reduced. *Geometric resolution:* Whole picture but crappy when reduced. *Radiometric resolution:* Number of colors.

1.7 Image Noise

additive Gaussian Noise Common model $I(x,y) = f(x,y) + c$, where $c \sim \mathcal{N}(0, \sigma^2)$. So that $p(c) = (2\pi\sigma^2)^{-1} e^{-c^2/2\sigma^2}$.

Poisson noise: **(shot noise)**

$$p(k) = \lambda^k e^{-\lambda}/k!$$

Rician noise: (appears in MRI)

$$p(I) = \frac{I}{\sigma^2} \exp\left(\frac{-(I^2 + f^2)}{2\sigma^2}\right) I_0\left(\frac{If}{\sigma^2}\right)$$

Multiplicative noise: $I = f + fc$

Signal to noise ration (SNR) $s = F/\sigma$ is an index of image quality, where

$$F = \frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} f(x,y)$$

Often used instead: *Peak Signal to Noise Ratio (PSNR)* $s_{\text{peak}} = F_{\max}/\sigma$

1.8 Colour Images

Consist of red, green and blue channel.

Prism (with 3 sensors) Separate light in three beams using dichroic prism. Requires 3 sensors and precise alignment. Gives good color separation. $\to$ high-end cameras

Filter mosaic Coat filter directly on sensor. “Demosaicing” to obtain full colour & full resolution image. $\to$ low-end cameras

Filter wheel rotate multiple filters in front of lens. Allows more than 3 colour bands. $\to$ static scenes

new color CMOS sensor, foveon’s X3 blue, green, red sensor, one above the other (descending) $\to$ better image quality

2 Image segmentation

“Segmentation is the ultimate classification problem. Once solved, Computer Vision is solved.”

Interim Summary Segmentation is hard. It is easier if you define the task carefully: *(1)* Segmentation task binary or continuous? *(2)* What are regions of interest? *(3)* How accurately must the algorithm locate the region boundaries?

Definition it partitions an image into *regions of interest*. It is the first stage in many automatic image analysis systems. A *complete segmentation* of an image I is a finite set of regions $R_1, \dots, R_N$, such that

$$I = \bigcup_{i=1}^{N} \text{ and } R_i \cap R_j = \varnothing, \qquad \forall i \neq j.$$

segmentation quality the quality of a segmentation depends on what you want to do with it. Segmentation algorithms must be chosen and evaluated with an application in mind.

2.1 Thresholding

Is a simple segmentation process, produces a binary image B. It labes each pixen *in* or *out* of the rogion of interest by comparison of the greylevel with a threshold T:

$$B(x,y) = \begin{cases} 1 & \text{if } I(x,y) \geq T \\ 0 & \text{if } I(x,y) < T. \end{cases}$$

Choosing T By trial and error. Compare results with ground truth. Automatic methods. (ROC curve)

Chromakeying Control Lighting! “Plain” discance measure (e.g.)

$$\mathbf{I}_\alpha = |\mathbf{I} - \mathbf{g}| > T$$

$T \sim 20$, $\mathbf{g} = \begin{pmatrix} 0 & 255 & 0 \end{pmatrix}^T$. Problems: Variation is *not* the same in all 3 channels. Hard alpha maske

$$I_{\text{comp}} = I_\alpha I_a + (1 - I_\alpha) I_b$$

Gaussian model per pixel (Like chromakeying.) mean $\mu \to I_\mu$, standard deviation $\sigma \to I_\Sigma$. $\mathbf{I}_\alpha = \left|\mathbf{I} - \mathbf{I}_{\text{bg}}\right| > \mathbf{T}$, $\mathbf{T} = \begin{pmatrix} 20 & 20 & 10 \end{pmatrix}$, $\mathbf{I}_{\text{bg}}$ = background image. Or better (e.g.)

$$\mathbf{I}_\alpha = \sqrt{\left(\mathbf{I} - \mathbf{I}_{\text{bg}}\right)^T \Sigma^{-1} \left(\mathbf{I} - \mathbf{I}_{\text{bg}}\right)} > \mathbf{T} = 4$$

ROC Analysis Receiver operating Characteristic. An ROC curve characterizes the performance of a binary classifier. A binary classifier distinguishes between two different types of things.

Classification error Binary classifiers make errors. Two types of input to a binary classifier: Positives, negatives. Four possible outcomes in any test: True positive, true negative, false negative, false positive.

ROC Curve Characterizes the error trade-off in binary classification tasks. It plots the *true positive fraction* (TP fraction)

$$= \text{true positive count}/P, \qquad P = TP + FN$$

and *false positive fraction* (FP fraction)

$$= \text{false positive count}/N, \qquad N = FP + TN.$$

ROC curve always passes through $(0,0)$ and $(1,1)$.

MAP (Maximum A Posteriori) detector

Operating points choose an *operating point* by assigning relative costs and values to each outcome, $V_{TN}, V_{TP}, C_{FN}, C_{FP}$. V and C being values and costs. For simplicity, often $V_{TN} = V_{TP} = 0$.

Performance Assessment In real-life, we use two or even three separate sets of test data: *(1)* A *training set*, for tuning the algorithm, *(2)* A *validation* set for tuning the performance score, *(3)* An unseen *test set* to get a final performance score on the tuned algorithm.

Pixel connectivity Define neighbors, e.g. (for 2D) 4-neighborhood or 8-neighborhood

Pixel paths There are e.g. 4- and 8-connected paths. (p_i neighbor of p_{i+1}).

Connected regions A region is 4- or 8-connected if it contains a(n) 4- or 8-connected path between any two of its pixels.

2.2 Region Growing

(1) Start from a seed point or region. *(2)* Add neighboring pixels that satisfy the criteria defining a region. *(3)* Repeat until we can include no more pixels.

```
function B = RegionGrow(I, seed)
  [X,Y] = size(I);
  visited = zeros(X,Y);
  visited(seed) = 1;
  boundary = emptyQ;
  boundary.enQ(seed);
while(~boundary.empty())
  nextPoint = boundary.deQ();
  if(include(nextPoint, seed))
    visited(nextPoint) = 2;
      Foreach (x,y) in N(nextPoint)
        if(visited(x,y) == 0)
          boundary.enQ(x,y);
          visited(x,y) = 1;
        end
      end
  end
end
```

2.2.1 Variations

seed selection *(1)* One seed point, *(2)* Seed region, *(3)* Multiple seeds.

seed selection *(1)* Greylevel thresholding, *(2)* Greylevel distribution model. E.g. include if $(I(x,y)-\mu)^2 < (n\sigma)^2$, $n=3$. Can update μ and σ after every iteration, *(3)* color or texture information.

snakes A snake is an *active contour*. It's a polygon. Each point on contour moves away from seed while its image neighborhood satisfies an inclusion criterion. Often the contour has smoothness constraints. the algorithm iteratively minimizes an energy function:

$$E = E_{\text{tension}} + E_{\text{stiffness}} + E_{\text{image}}$$

2.3 Spatial relations

Markov Random Fields Markov chains have 1D structure. At every time, there is one state. This enabled use of dynamic programming. *Markov Random fields* break this 1D structure: • Field of sites, each of which has a label, simultaneously. • Label at one site dependend on others, no 1D structure to dependencies. • This means no optimal, efficient algorithms, except for 2-label problems. Minimize

$$\text{Energy}(\mathbf{y};\theta,\text{data}) = \sum_i \psi_1(y_i;\theta,\text{data}) + \sum_{\spadesuit} \psi_2(y_i,y_j;\theta,\text{data})$$

$\spadesuit = i,j \in$ edges

FG-BG segmentation The code does the following: • background RGB Gaussian model training (from many images) • shadow modeling (hard shadow and soft shadow) • graphcut foreground-background segmentation

2.4 Morphological Operations

They are local pixel transformations for processing region shapes. Most often used on binary images. Logical transformations based on comparison of pixel neighborhoods with a pattern.

8-neighbor erode (Minkowsky subtraction) Erase any foreground pixel that has one eight-connected neighbar that is background

8-neighbor dilate (Minkowsky addition) Paint any background pixel that has one eight-connected neighbor that is foreground. *Applications*: Smooth region boundaries for shape analysis, remove noise and artefacts from an imperfect segmentation, match particular pixel configurations in an image for simple object recognition

structuring elements morphological operations take two arguments 1. a binary image 2. a structuring element Compare the structuring element to the neighborhood of each pixel. This determines the output of the morphological operation. The structuring element is also a binary array and has an origin.

$$I_1 \cup I_2 = \{\mathbf{x} : \mathbf{x} \in I_1 \text{ or } \mathbf{x} \in I_2\}\,,$$
$$I_2 \cap I_2 = \{\mathbf{x} : \mathbf{x} \in I_1 \text{ and } \mathbf{x} \in I_2\}\,,$$
$$I^C = \{\mathbf{x} : \mathbf{x} \notin I\}\,,$$
$$I_1 \setminus I_2 = \{\mathbf{x} : \mathbf{x} \in I_2 \text{ and } \mathbf{x} \notin I_2\}\,.$$

Erosion of binary image I by the structuring element S is defined by

$$I \ominus S = \{\mathbf{z} \in E \mid S_{\mathbf{z}} \subset I\}$$

$S_{\mathbf{z}}$ translation of S by vector $\mathbf{z}$.

Dilation is $I \oplus S = \bigcup_{\mathbf{b}\in S} I_{\mathbf{b}}$.

Opening $I \circ S = (I \ominus S) \oplus S$.

Closing $I \bullet S = (I \oplus S) \ominus S$.

To remove holes in the foreground and islands in the background, do both opening and closing. Thesize and shape of the structuring element determine which features survive. In the absence of knowledge about the shape of features to remove, use a circular structuring element.

Granulometry Provides a size distribution of distinct regions or "granules" in the image. We open (opening as above) the image with increasing structuring element size and count the number of regions after each operation. Creates "morphological sieve".

```
function gSpec = granulo(I, T, maxRad
  )
% Segment the image I.
B = (I>T);
%Open the image at each structuring element size up
%to a maximum and count the remaining regions.
for x=1:maxRad
O = imopen(B,strel('disk',x));
numRegions(x) = max(max(
      connectedComponents(O)));
end
gSpec = diff(numRegions);
```

Hit-and-miss transform $H = I \otimes S$ Searches for an exact match of the structuring element. Simple form of template matching.

Thinning $I \oslash S = I \setminus (I \otimes S)$

Thickening $I \odot S = I \cup (I \otimes S)$

Sequential thinning/thickening With structuring elements $S_1,\ldots,S_n$ and sequential thinning/thickening $\spadesuit$

$$I \spadesuit \{S_i : i=1,\ldots,n\} = ((I \spadesuit S_1)\cdots \spadesuit S_n)$$

Several sequences of structuring elements are useful in practice. These are usually the set of rotations of a single stnuctuning element, sometimes called the *Golay alphabet*. See bwmorph in matlab.

2.4.1 Medial Axis Transform (MAT, skeletonization)

The skeleton and MAT are stick-figure representations of a region $X \in \mathbb{R}^2$. Start a grassfire at the boundary of the region, theskeleton is the set of points at which two fire fronts meet.

Skeleton Use structuring element

$$B = \begin{pmatrix} 0&1&0\\ 1&1&1\\ 0&1&0 \end{pmatrix}.$$

The n-th skeleton subset is

$$S_n(X) = (X \ominus_n B) \setminus [(X \ominus_n B) \circ B]\,,$$

where $\ominus_n$ denotes n successive erosions. The skeleton is the union of all the skeleton subsets $S(X) = \cup_{n=1}^{\infty} S_n(X)$.

Reconstruction can reconstruct region X from its *skeleton subsets*.

$$X = \cup_{n=0}^{\infty} S_n(X) \oplus_n B$$

Applications and problems The skeleton/-MAT provies a stick figure representing the rogion shap. Used in object recognition, in particula, character recognition. *Problems:* Definition of a maximal disc is poorly defined on a digital grid and is sensive to noise on the boundary. Sequential thinning output sometimes preferred to skeleton/MAT.

3 Image filtering

Image filtering is modifying the pixels in an image based on some function of a local neighborhood of the pixels.

3.1 Linear Shift-Invariant Filtering

About modifying pixels based on *neighborhood*. Local methods simplest. Linear means *linear combination* of neighbors. Linear methods simplest. *Shift-invariant* means doing the same for each pixel. Same for all is simplest. Useful to: Low-level image processing operations, smoothing and noise reduction, sharpen, detect or enhance features.

Linear operation L is a *linear* operation if

$$L[\alpha I_1 + \beta I_2] = \alpha L[I_1] + \beta L[I_2]$$

Output I' of linear image operation is a weighted sum of each pixel in the input I

$$I'_j = \sum_{i=1}^{N} \alpha_{ij} I_i, \qquad j = 1\cdots N$$

Linear Filtering Linear operations can be written:

$$I'(x,y) = \sum_{i,j\in N(x,y)} K(x,y;i,j) I(i,j)$$

I = input image; I' = output of operation. k is *kernel* of the operation. $N(m,n)$ is a neighbourhood of (m,n).

Correlation e.g. template matching. Linear operation: $I' = KI$

$$I'(x,y) = \sum_{i,j\in N(x,y)} K(i,j) I(x+i, y+j)$$

Convolution e.g. point spread function

$$I'(x,y) = \sum_{i,j\in N(x,y)} K(i,j) I(x-i, y-j)$$

Edge The filter window falls off the edge of the image, we need to extrapolate, methods: *(1)* clip filter (black) *(2)* wrap around *(3)* copy edge *(4)* reflect across edge *(5)* vary filter near edge

Filter at boundary *(1)* ignore, copy or trucate. No processing of boundary pixels. Pad image with zeros (matlab). Pad image with copies of edge rows/columns *(2)* truncate kernel *(3)* reflected indexing *(4)* circular indexing

Separable Kernels Separable filters can be written

$$K(m,n) = f(m)g(n)$$

for a rectangular neighbourhood with size $(2M+1)\times(2N+1)$,

$$I'(m,n) = f * (g * I(N(m,n))),$$
$$I''(m,n) = \sum_{j=-N}^{N} g(j) I(m, n-j),$$
$$I'(m,n) = \sum_{i=-M}^{M} f(i) I''(m-i, n).$$

$\rightarrow (2M+1)+(2N+1)$ operations!

Smoothing kernels (low-pass filters) • Mean filter: $\frac{1}{9}\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}$, • Weighted smoothing filters: $\frac{1}{10}\begin{pmatrix}1&1&1\\1&2&1\\1&1&1\end{pmatrix}$, $\frac{1}{16}\begin{pmatrix}1&2&1\\2&4&2\\1&2&1\end{pmatrix}$.

Gaussian Kernel Idea: Weight contributions of neighboring pixels:

$$\mathcal{N}_{\mu=0,\sigma}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Smoothing with a Gaussian instead of a box filter removes the artefact of the vertical and horizontal lines. Gaussian smoothing Kernel is *separable*! $\mathcal{N}(x,y) = \mathcal{N}(x)\mathcal{N}(y)$. Amount of smoothing depends on σ and window size. Width $> 3\sigma$.

Scale space Convolution of a Gaussian with σ with itself is a gaussian with $\sigma\sqrt{2}$. Repeated convolution by a Gaussian filter produces the scale space of an image.

Gaussian filter top-5 *(1)* Rotationally symmetric. *(2)* Has a single lobe. $\rightarrow$ Neighbor's influence decreases monotonically. *(3)* Still one lobe in frequency domain. $\rightarrow$ No corruption from high frequencies *(4)* Simple relationship to σ *(5)* Easy to implement efficiently

Differential filters • Prewitt operator: $\begin{pmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{pmatrix}$, • Sobel operator: $\begin{pmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{pmatrix}$.

High-pass filters • Laplacian operator: $\begin{pmatrix}0&1&0\\1&-4&1\\0&1&0\end{pmatrix}$, • High-pass filter: $\begin{pmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{pmatrix}$.

Differentiation and convolution

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon\to 0}\left(\frac{f(x+\varepsilon,y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon}\right)$$
$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1},y) - f(x_n,y)}{\Delta x},$$

which is obviously a convolution $\begin{pmatrix}-1 & 1\end{pmatrix}$

Filters and templates Filters at some point can be sees as taking a ·-product between the image and some vector, the image is a set of dot products, filters look like the effects they are intended to find, filters find effects they look like.

Image sharpening Also known as enhancement. Increases the high frequency components to enhance edges.

$$I' = I + \alpha\,|K * I|\,,$$

where K is a high-pass filter kernel and $\alpha \in [0,1]$.

Integral images integral images (also known as summed-area tables) allow to efficiently compute the convolution with a constant rectangle

$$\mathcal{I}(x,y) = \int_0^x \mathrm{d}x' \int_0^y \mathrm{d}y'\, I(x',y')$$

$$\begin{aligned} A &= \mathcal{I}(1) & A + C &= \mathcal{I}(3) \\ A + B &= \mathcal{I}(2) & A + B + C + D &= \mathcal{I}(4) \end{aligned}$$

$$D = \mathcal{I}(4) - \mathcal{I}(2) - \mathcal{I}(3) + \mathcal{I}(1)$$

Also possible along diagonal.

Viola-Jones cascade face detection Very efficient face detection using integral images.

4 Questions

- $$I_{\text{comp}} = I_\alpha I_a + (1 - I_\alpha)\, I_b$$
- MAP, Maximum a posteriori detector.
- graph cuts
- Solve MRFs with graph cuts