# 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?**

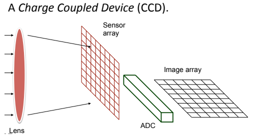
- A picture or pattern of a value varying in space and/or time
- Representation of a function $f: \mathbb{R}^n \to S$
- In digital form, e.g.: $I: \{1,\dots,X\}\times\{1,\dots,Y\}\to S$.
- For greyscale CCD images, $n=2, S=\mathbb{R}^+$

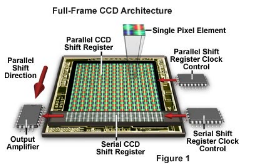
**What is a pixel?**

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

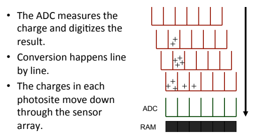
## 1.2 Image sources

**digital camera (CCD)**




**analog to digital Conversion**




**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 returns 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**

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

**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 $= 2^{24} = 16.7$M 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. → high-end cameras

**Filter mosaic** Coat filter directly on sensor. "Demosaicing" to obtain full colour & full resolution image. → low-end cameras

**Filter wheel** rotate multiple filters in front of lens. Allows more than 3 colour bands. → static scenes

**new color CMOS sensor, foveon's X3** blue, green, red sensor, one above the other (descending) → 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

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

**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 region of interest by comparison of the greylevel with a threshold $T$:

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

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

**Chromakeying** Control Lighting! "Plain" distance measure (e.g.)

$$I_\alpha = |I-g| > T$$

$T\sim 20$, $g=\begin{pmatrix}0 & 255 & 0\end{pmatrix}^T$. 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 $\sigma\to I_\Sigma$. $I_\alpha = |I-I_{bg}| > T$, $T=\begin{pmatrix}20 & 20 & 10\end{pmatrix}$, $I_{bg}$ = background image. Or better (e.g.)

$$I_\alpha=\sqrt{(I-I_{bg})^T\Sigma^{-1}(I-I_{bg})}>T$$

**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 = true positive count$/P$,

$P=TP+FN$ and *false positive fraction*

FP fraction = false positive count$/N$,

$N=FP+TN$. 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
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 $\mu$ and $\sigma$ 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

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

$\spadesuit = i,j\in$ 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.

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

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

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

$S_z$ translation of $S$ by vector z.

**Dilation** is $I\oplus S=\bigcup_{b\in S}I_b$.

**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 $S_1,\dots,S_n$ and sequential thinning/thickening $\spadesuit$

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

Several sequences of structuring elements are useful in practice. These are usually the set of rotations of a single stnucturing 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 $S(X)=\bigcup_{n=1}^{\infty}S_n(X)$.

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

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

**Applications and problems** The skeleton/MAT provies a stick figure representing the region 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 $(2M+1)\times(2N+1)$,

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

→ $(2M+1)+(2N+1)$ 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 $\sigma$ and window size. Width $>3\sigma$.

**Scale space** Convolution of a Gaussian with $\sigma$ 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. → Neighbor's influence decreases monotonically. *(3)* Still one lobe in frequency domain. → No corruption from high frequencies *(4)* Simple relationship to $\sigma$ *(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 $(-1\ 1)$

**Filters and templates** Filters at some point can be seen 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

$$\mathscr{I}(x,y) = \int_0^x dx' \int_0^y dy'\, I(x',y')$$

$$A = \mathscr{I}(1) \qquad A + C = \mathscr{I}(3)$$

$$A + B = \mathscr{I}(2) \qquad A + B + C + D = \mathscr{I}(4)$$

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

Also possible along diagonal.

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

# 4 Image features

## 4.1 Template matching

**Problem** Locate an object, described by a template $t(x,y)$, in the image $s(x,y)$. *Example:* Passport photo as image and eyes to detect.

**Method** Search for the best match by minimizing mean -squared error $E(p,q)$

$$= \sum_{x,y=-\infty}^{\infty} [s(x,y) - t(x-p,y-q)]^2$$

$$= \sum_{x,y=-\infty}^{\infty} |s(x,y)|^2 + |t(x,y)|^2$$

$$-2 \cdot \sum_{x,y=-\infty}^{\infty} s(x,y) \cdot t(x-p,y-q)$$

Equivalently, maximize *area correlation*

$$r(p,q) = \sum_{x,y=-\infty}^{\infty} s(x,y) \cdot t(x-p,y-q)$$

$$= s(p,q) * t(-p,-q)$$

$$\leq \sqrt{\left[\sum |s(x,y)|^2\right] \cdot \left[\sum |t(x,y)|^2\right]},$$

where in the last step the Cauchy-Schwarz inequality was used. Equality $\Leftrightarrow$

$$s(x,y) = \alpha \cdot t(x-p,y-q) \quad \text{with } \alpha \geq 0$$

Area correlation is equivalent to convolution of image $s(x,y)$ with impulse response $t(-x,-y)$.

**Diagram of template matcher**

$\xrightarrow{s(x,y)}$ $t(-x,-y)$ $\xrightarrow{r(x,y)}$ search peak(s) $\rightarrow$ object location(s) $p,q$

Remove mean before template matching to avoid bias towards bright image areas.

## 4.2 Edge detection

Idea (continuous-space): Detect local gradient

$$\|\nabla(f(x,y))\| = \sqrt{(\partial_x f)^2 + \left(\partial_y f\right)^2}$$

Digital image: Use finite differences instead:
**difference** $(-1\ 1)$, **central difference** $(-1\ [0]\ 1)$; **Prewitt** $\begin{pmatrix} -1 & 0 & 1 \\ -1 & [0] & 1 \\ -1 & 0 & 1 \end{pmatrix}$, $\begin{pmatrix} -1 & -1 & 1 \\ 0 & [0] & 0 \\ -1 & -1 & 1 \end{pmatrix}$; **Sobel** $\begin{pmatrix} -1 & 0 & 1 \\ -2 & [0] & 2 \\ -1 & 0 & 1 \end{pmatrix}$, $\begin{pmatrix} -1 & -2 & 1 \\ 0 & [0] & 0 \\ -1 & -2 & 1 \end{pmatrix}$; **Roberts** $\begin{pmatrix} [0] & 1 \\ -1 & 0 \end{pmatrix}$, $\begin{pmatrix} [1] & 0 \\ 0 & -1 \end{pmatrix}$,

**Laplacian operator** Detects discontinuities by considering second derivative

$$\nabla^2 f(x,y) = \partial_x^2 f(x,y) + \partial_y^2 f(x,y).$$

Isotropic (rotationally invariant) operator, zero-crossings mark edge location, discrete-space approximation by convolution with $3 \times 3$ impulse response $\begin{pmatrix} 0 & 1 & 0 \\ 1 & [-4] & 1 \\ 0 & 1 & 0 \end{pmatrix}$, or $\begin{pmatrix} 0 & 1 & 0 \\ 1 & [-8] & 1 \\ 0 & 1 & 0 \end{pmatrix}$,

**Laplacian of Gaussian** The Laplacian operator is very sensity to fine detail and noise, so blur it first with Gaussian.$\rightarrow$ do it in one operator Laplacian of Gaussian (LoG)

$$\mathrm{LoG}(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

### 4.2.1 Canny edge detector

*(1)* Smooth image with a gaussian filter *(2)* Compute gradient magnitude and angle (Sobel, Prewitt,…)

$$M(x,y) = \sqrt{(\partial_x f)^2 + \left(\partial_y f\right)^2},$$

$$\alpha(x,y) = \arctan\left(\partial_y f / \partial_x f\right)$$

*(3)* Apply nonmaxima suppression to gradient magnitude image *(4)* Double tresholding to detect strong and weak edge pixels *(5)* Reject weak edge pixels not connected with strong edge pixels

**Canny nonmaxima suppression** Quantize edge normal to one of four directions: horizontal, $-45°$, vertical, $45°$. If $M(x,y)$ is smaller than either of its neighbors in edge normal direction $\rightarrow$ suppress; else keep

**Double-thresh. of grad. magn.**

**strong edge:** $M(x,y) \geq \theta_{\text{high}}$

**weak edge:** $\theta_{\text{high}} > M(x,y) \geq \theta_{\text{low}}$

Typical setting: $\theta_{\text{high}}, \theta_{\text{low}} = 2, 3$. Region labeling of edge pixels. Reject regions without strong edge pixels.

## 4.3 Feature detection

### 4.3.1 Hough transform

*Problem:* fit a straight line (or curve) to a set of edge pixels. Hough transform (1962): generalized template matching technique. *(1)* Consider detection of straight lines $y = mx + c$. *(2)* draw a line in the parameter space $m,c$ for each edge pixel $x,y$ and increment bin counts along line. Detect peak(s) in $(m,c)$ plane. *(3)* Alternative parametrization avoids infinite-slope problem $x\cos\theta + y\sin\theta = \rho$

**circle detection** find circles of fixed radius $r$. For circles of undetermined radius, use 3d Hough transform for parameters $(x_0, y_0, r)$

### 4.3.2 Detecting corner points

Many applications benefit from features localized in $(x,y)$. Edges well localized only in one direction $\rightarrow$ detect corners. Desirable properties of corner detector: *(1)* Accurate localization, *(2)* invariance against shift, rotation, scale, brightness change, *(3)* robust against noise, high repeatability

### 4.3.3 Most accurately localizable patterns

**Local displacement sensitivity**

$$S(\Delta x, \Delta y) = \sum_{x,y \in \text{window}} [f(x,y) - f(x-\Delta x, y+\Delta y)]$$

**Linear approximation for small $\Delta x, \Delta y$**

$$f(x+\Delta x, y+\Delta x)$$

$$\approx f(x,y) + \partial_x f(x,y)\Delta x + \partial_y(x,y)\Delta y$$

$$S(\Delta x, \Delta y) \approx \sum_{(x,y) \in \text{window}} \left[\begin{pmatrix} \partial_x f & \partial_y f \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}\right]$$

$$= \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

**Feature point extraction**

$$\mathrm{SSD} \approx \Delta^T M \Delta$$

Find points for which the following is large

$$\min \Delta^T M \Delta$$

for $\|\Delta\| = 1$. i.e. maximize eigenvalues of $M$.

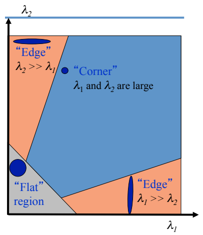
**Keypoint detection** Often based on eigenvalues $\lambda_1, \lambda_2$ of $M$ ("structure matrix"/"normal matrix"/second-moment matrix")

$$M = \sum_{(x,y) \in \text{window}} \begin{pmatrix} (\partial_x f)^2 & \partial_x f \partial_y f \\ \partial_x f \partial_y f & \left(\partial_y f\right)^2 \end{pmatrix},$$

Measure of "cornerness"

$$C(x,y) = \det(M) - k \cdot (\text{trace } M)^2$$

$$= \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)$$




**Corner importance weight** Give more importance to central pixels by using Gaussian weighting function

$$M = \sum_{x,y \in \text{window}} G(x - x_0, y - y_0, \sigma) \cdot \begin{pmatrix} (\partial_x f)^2 & \partial_x f \partial_y f \\ \partial_x f \partial_y f & \left(\partial_y f\right)^2 \end{pmatrix}$$

Compute subpixel localization by fitting parabola to *cornerness function*

**Robustness of Harris corner detector** *(1)* Invariant to brightness offset: $f(x,y) \rightarrow f(x,y) + c$ *(2)* Invariant to shift and rotation *(3)* Not invariant to scaling

### 4.3.4 Lowe's SIFT features

Recover features with position, orientation and scale.

**Position** *(1)* Look for strong responses of DoG filter, *(2)* only consider local maxima.

**Scale** *(1)* Look for strong responses of DoG filter over scale space. *(2)* only consider local maxima in both position and scale. *(3)* Fit quadratic around maxima for subpixel accuracy.

**Orientation** *(1)* Create histogram of local gradient directions computed at selected scale. *(2)* Assign canonical orientation at peak of smoothed histogram. *(3)* Each key specifies stable 2D coordinates $(x,y,\text{scale},\text{orientation})$

**SIFT descriptor** *(1)* Thresholded image gradients are sampled over $16 \times 16$ array of locations in scale space. *(2)* Create array of orientation histograms *(3)* 8 orientations $\times$ 4 $\times$ 4 histogram array = 128 dimensions

# 5 Fourier Transform

## 5.1 Aliasing

One can't shrink an image by taking every second pixel. If we do, characteristic errors appear. Typically, small phenomena look bigger; fast phenomena can look slower. Common phenomenons *(1)* Wagon wheels rolling the wrong way in movies. *(2)* Checkerboards misrepresented in ray tracing *(3)* Striped shirts look funny on color television.

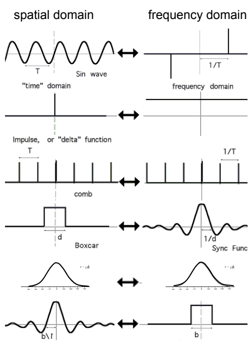
## 5.2 Definition

Represent function on a new basis. Basis elements have the form $e^{-i2\pi(ux+vy)}$. The Fourier transform is

$$\hat{f}(u,v) = \iint_{\mathbb{R}^2} dx dy\, f(x,y) e^{-i2\pi(ux+vy)}$$

Basis functions of Fourier transform are eigenfunctions of linear systems.

**Important functions**




**Convolution theorem** *(1)* The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\hat{f} \cdot \hat{g} = \widehat{f * g}$$

The Fourier transform of the product of two functions is the convolution of the Fourier transforms

$$\hat{f} * \hat{g} = \mathscr{F}(f \cdot g)$$

## 5.3 Sampling

Go from continuous world to discrete world, from function to vector. Samples are typically measured on regular grid. We want to be able to approximate integrals sensibly $\rightarrow$ Delta function

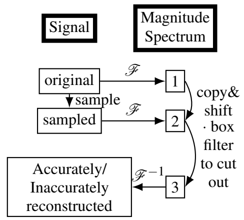
$$\mathscr{S}_{2D}(f(x,y))$$

$$= \sum_{i,j=-\infty}^{\infty} f(x,y)\delta(x-i, y-j)$$

$$= f(x,y) \sum_{i,j=-\infty}^{\infty} \delta(x-i, y-j),$$

with $\mathscr{S}$ = Sampler operator.

**FT of sampled signal**




In the figure above the accuracy depends on the overlapping wave functions in "2". The box filter then can't cut out appropriately the magnitude spectrum to get a proper result in "3". This leads to an inaccurately reconstructed signal.

**Proper sampling** To avoid this effect, this is the procedure:

$\boxed{\text{original signal}} \xrightarrow{\text{lp filtering}} \boxed{\text{lp filt. sign}} \xrightarrow{\text{sample}} \boxed{\text{sampl.sign.}} \xrightarrow{\text{reconstr.}} \boxed{\text{reconstr.sign}}$

**Smoothing as low-pass filtering** The message of the FT is that high frequencies lead to trouble with sampling. Solutionsppress high frequencies before sampling. A filter whose FT is a box is bad, because the filter kernel has infinite support. Common solution: use a Gaussian.

**Nyquist sampling theorem** Nyquist theorem: The sampling frequency must be at least twice the highest frequency. $\omega_s \geq 2\omega$. If this is no the case, the signal needs to be bandlimited before sampling, e.g. with a low-pass filter.

## 5.4 Image Restoration

**Pixelization** Possibilities: Square pixels, Gaussian reconstruction filter, Bilinear interpolation, perfect reconstruction filter.

**Motion blurring** Each light dot is transformed into a short line along the $x_1$-axis:

$$h(x_1, x_2) = \frac{1}{2\ell}\left[\theta(x_1 + \ell) - \theta(x_1 - \ell)\right]\delta(x_2)$$

**Noise** Gaussian blurring kernel:

$$h(x_1, x_2) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x_1^2 + x_2^2}{2\sigma^2}\right)$$

**Problem** $f(\mathrm{x}) \xrightarrow{h(\mathrm{x})} g(\mathrm{x}) \xrightarrow{\tilde{h}(\mathrm{x})} f(\mathrm{x})$. The "inverse" kernel $\tilde{h}(\mathrm{x})$ should compensate the effect of the image degradation $h(\mathrm{x})$, i.e.,

$$\left(\tilde{h} * h\right)(\mathrm{x}) = \delta(\mathrm{x})$$

$\tilde{h}(x)$ may be determined more easily in Fourier space:

$$\mathscr{F}\left[\tilde{h}\right](u,v) \cdot \mathscr{F}[h](u,v) = 1$$

To determine $\mathscr{F}\left[\tilde{h}\right]$, we need to estimate *(1)* the distortion model $h(\mathrm{x})$ (point spread function) or $\mathscr{F}[h](u,v)$ (modulation transfer function) *(2)* the parameters of $h(\mathrm{x})$, e.g. for defocussing

**Motion Blur FT**

$$\mathscr{F}[h](u,v) = \frac{1}{2\ell}\int_{-\ell}^{\ell} dx_1 \exp(-i2\pi u x_1) \cdot \underbrace{\int_{-\infty}^{\infty} dx_2\, \delta(x_2) \exp(-i2\pi v x_2)}_{=1}$$

$$= \mathrm{sinc}(2\pi u \ell)$$

Problem: $\mathscr{F}[\tilde{h}](u) = 1/\hat{h}(u)$. sinc has many zeroes and these frequencies can't be recovered! Solution: Regularized reconstruction filter

$$\tilde{F}[\tilde{h}](u,v) = \frac{\mathscr{F}[h]}{\|\mathscr{F}\|^2 + \varepsilon}$$

Singularities are avoided by the rugelarization $\varepsilon$.

**Space-time super-resolution** One can put two movies of the same thing and merge their frames for space and time super-resolution.

**Spatial super-resolution** • lens + pixel = low-pass filter (edisered to avoid aliasing) • Low-res images $= D * H * G *$ (desired high-res-image). D:decimate, H:lens+pixel, G: Geometric warp • Smplified case for translation: $LR = (D * G) * (H * HR)$. G is shift-invariant and commutes with H. First compute H HR, then deconvolve HR with H. • Super-resolution needs to restore attenuated frequencies. Many images improve $S/N$ ration $\sim \sqrt{n}$, which helps. Eventually Gaussian's double exponential always dominates.

# 6 Unitary transforms

Digital image as a matrix:

$$f = \begin{pmatrix} f(0,0) & \cdots & f(N-1,0) \\ \vdots & \ddots & \vdots \\ f(0,L-1) & \cdots & f(N-1,L-1) \end{pmatrix}$$

$$= f_{yx}$$

or as a vector

$$\mathrm{f} = \begin{pmatrix} f(0,0) \\ \vdots \\ f(N-1,L-1) \end{pmatrix}$$

**General approach** *(1)* Sort samples $f(x,y)$ of an $M \times N$ image (or rectangular block in the image) into column vector of length $M \times N$. *(2)* Compute transform coefficients $\mathrm{c} = A\mathrm{f}$ where $A$ is a matrix of size $(MN)^2$. *(3)* Transform $A$ is unitary, iff $A^{-1} = A^*$ *(4)* If $A$ is real-valued, i.e. $A = \overline{A}$, transform is orthonormal.

**Energy conservation** $\|\mathrm{c}\|^2 = \mathrm{c}^*\mathrm{c} = \mathrm{f}^* A^* A \mathrm{f} = \|\mathrm{f}\|^2$

**Image collection** $f_i$ one image, $F = f_1, \ldots, f_n$,

**Auto-correltion function**

$$R_{ff} = E[f_i f_i^*] = FF^*/n$$

**energy distribution** Energy is conserved, but often will be unevely distributed among coefficients. Autocorrelation matrix:

$$R_{\mathrm{cc}} = E[\mathrm{cc}^*] = E[A\mathrm{ff}^* A^*] = AR_{\mathrm{ff}}A^*$$

Mean squared values ("average energies") of the coefficients $c_i$ are on the diagonal of $R_{\mathrm{cc}}$.

$$E[c_i^2] = [R_{\mathrm{cc}}] = [AR_{\mathrm{ff}}A^*]_{ii}$$

**Eigenmatrix of autocorrelation matrix** Definition: Eigenmatrix $\Phi$ of autocorrelation matrix $R_{ff}$. *(1)* $\phi$ is unitary *(2)* The columns of $\Phi$ form a set of eigenvectors of $R_{ff}$, i.e.,

$$R_{ff}\Phi = \Phi\Lambda,$$

where $\Lambda = \mathrm{diag}(\lambda_0, \ldots, \lambda_{MN-1})$. *(3)* $R_{ff}$ is symmetric nonnegative definite, hence $\lambda_i \geq 0$ for all $i$ *(4)* $R_{ff}$ is normal matrix, i.e. $R_{ff}^* R_{ff} = R_{ff} R_{ff}^*$, hence unitary eigenmatrix exists.

## 6.1 Karhunen-Loeve Transform

Strongly correlated samples with equal energies $\xrightarrow{A}$ uncorrelated samples, most of the energy in first coefficient.

**Properties** *(1)* Unitary transform with matrix $A = \Phi^*$ where the columns of $\phi$ are ordered according to decreasing eigenvalues. *(2)* Transform coefficients are pairwise uncorrelated

$$R_{\mathrm{cc}} = AR_{\mathrm{ff}}A^* = \Phi^* R_{\mathrm{ff}} = \Phi^* \Phi \Lambda = \Lambda$$

*(3)* Energy concentration property: No other unitary transform packs as much energy into the first $J$ coefficients, where $J$ is arbitrary. Mean squared approximation error by choosing only first $J$ coefficients is minimized.

**Optimal energy concentration** *(1)* To show optimum energy concentration property, consider the truncated coefficient vector $\mathrm{b} = I_j\mathrm{c}$, where $I_J$ contain ones on the first $J$ diagonal positions, else zeros. *(2)* Energy in first $J$ coefficients for arbitrary transform $A$

$$E = \mathrm{tr}(R_{\mathrm{bb}}) = \mathrm{tr}(I_J R_{cc} I_J)$$

$$= \mathrm{tr}(I_j A R_{\mathrm{ff}} A^* I_j) = \sum_{k=0}^{J-1} a_k^T R_{\mathrm{ff}} \overline{a}_k$$

where $a_k^T$ is the $k$-th row of $A$. *(3)* Lagrangian cost function to enforce unit-length basis vectors

$$L = E + \sum_{k=0}^{J-1} \lambda_k \left(1 - a_k^T \overline{a}_k\right)$$

$$= \sum_{k=0}^{J-1} a_k^T R_{\mathrm{ff}} \overline{a}_k + \sum_{k=0}^{J-1} \lambda_k \left(1 - a_k^T \overline{a}_k\right)$$

Differentiating L with respect to $a_j$ yields necessary condition

$$R_{\mathrm{ff}} \overline{a}_j = \lambda_j \overline{a}_j, \qquad \forall j < J$$

## 6.2 Basis images and eigenimages (EI)

For a unitary transform, the inverse transform $\mathrm{f} = A^*\mathrm{c}$ can be interpreted in terms of the superpositions of "basis images" (columns of $A^*$) of size $MN$. If the trasnform is a KL transform, the basis images, which are the eigenvectors of the autocorrelation matrix $R_{\mathrm{ff}}$, are called "eigenimages". If energy concentration works well, only a limited number of eigenimages is needed to approximate a set of images with small error. These eigenimages form an optimal linear subspace of dimensionality $J$.

**EI for recogition** To recognize complex patterns (e.g., faces), large portions of an image (say of size $MN$) might have to be considered. High dimensionality of "image space" means high computational burden for many recognition techniques. Transform $\mathrm{c} = W\mathrm{f}$ can reduce dimensionality from $MN$ to $J$ by representing the image by $J$ coefficients. Idea: tailor a KLT to the specific set of images of the recognition task to preserve the salient features.

**Simple recognition** Simple Euclidean distance (SSD) between images. Best match wins

$$\operatorname*{argmin}_i D_i = \|I_i - I\|$$

Computationally expensive, i.e. requires presented image to be correlated with every image in the database!

**Eigenspace matching** Let $I_i$ be the input image, $I$ the database. The "character" of the face $\hat{J} = J - \langle I \rangle$, with $J$ being any image (set). Do KLT (aka PCA) transformation

$$\hat{I}_i \mapsto p_i, \quad E^* \hat{I}_i = p_i.$$

$$\leadsto \hat{I}_i \approx E p_i,$$

$$\leadsto I_i - I = \hat{I}_i - \hat{I} \approx E(p_i - p),$$

$$\leadsto \|I_i - I\| \approx \|p_i - p\|,$$

with closest rank-k approximation property of SVD. Approximate

$$\operatorname*{argmin}_i D_i = \|I_i - I\| \approx \|p_i - p\|.$$

## 6.3 Eigenfaces (EF)

Concatenate face pixels into "observation vector", x.

**EI for recognition** *(1)* Input image, *(2)* normalize, *(3)* subtract mean face, *(4)* KLT, *(5)* Find most similar $p_i$, *(6)* similarity measure, *(7)* rejection system, *(8)* result of identificiation.

**Limitations of EFs** Differences due to varying illumination can be much larger than differences between faces!

## 6.4 Fisherfaces/LDA

Training data: For eigenfaces distance of difference of illumination are within indivual variance. Key idea: Find directions where ratio of between/within individual variance are maximized. Linearly project to basis where dimension with good signal to nois ration are maximized.

**Fisher linear discriminant analysis** Eigenimage method maximizes "scatter" within the linear subspace over the entire image set - regardless of classification task

$$E_{\text{opt}} = \underset{E}{\operatorname{argmax}}(\det(ERE^*)),$$

Fisher linear discriminant analysis: Maximize between-class scatter, while minimizing within-class scatter,

$$F_{\text{opt}} = \underset{F}{\operatorname{argmax}} \left( \frac{\det(FR_B W^*)}{\det(FR_W F^*)} \right),$$

$$R_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^*,$$

$$R_W = \sum_{\substack{i=1,\ldots,c \\ \Gamma_\ell \in \text{Class}(i)}} (\Gamma_\ell - \mu_i)(\Gamma_\ell - \mu_i)^*.$$

$N_i$ are the samples in class $i$ and $\mu_i$ is the mean in class $i$. Solution: Generalized eigenvectors $w_i$ corresponding to the $k$ largest eigenvalues $\{\lambda_i \mid i = 1,\ldots,k\}$, i.e.

$$R_B w_i = \lambda_i R_w w_i, i = 1,\ldots,k$$

. Problem: within-class scatter matrix $R_W$ at most of rank $L-c$, hence usually singular. Apply KLT first to reduce dimension of feature space to $L-c$ (or less), proceed with Fisher LDA in low-dimensional space.

**Eigenfaces vs. Fisherfaces** Eigenfaces conserve energy but the two classes e.g. in 2D are no longer distinguishable. FLD (Fisher LDA) separates the classes by choosing a better 1D subspace. Fisher faces are much better in varying illuminations.

**Varying illumination (FF)** All images of same Lambertian surface with different illumination (without shadows) ile in a 3D linear subspace. Single point source at infinity

$$f(x,y) = a(x,y) \left( \ell^T n(x,y) \right) L,$$

$a(x,y)$ surface albedo, $L$ light source intensity. Superposition of arbitrary number of point sources at infinity is still in same 3D linear subspace, due to linear superposition of each contribution to image. Fisher images can eliminate withi-class scatter.

**Appearance manifold approach**

For everyobject, *(1)* sample the set of viewing conditions *(2)* use these images as feature vectors *(3)* apply a PCA over all the images *(4)* keep the dominant PCs *(5)* sequence of views for one object represent a manifold in space of projections *(6)* what is the nearest manifold for a given view?

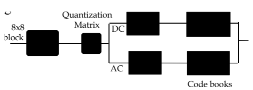
**Object-pose manifold** Appearance changes projected on PCs (1D pose changes). Sufficient characterization for recognition and pose estimation.

### 6.5 JPEG image compression

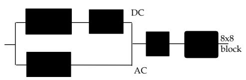
We don't resolve high frequencies too well. . . let's use this to compress images . . . JPEG!

**Concept** Block-based discrete cosine transform (DCT)

**JPEG Encoding and Decoding**
Encoding:




Decoding:




**DCT** A variant of discrete fourier transform: Real numbers, fast implementation. Block sizes: *(1)* small block: faster, correlation exists between neighboring pixels *(2)* better compression in smooth regions The first coefficient $B(0,0)$ is the DC component, the average intensity. The top-left coefficients represent low frequencies, the bottom right hight frequencies.

**Entropy Coding (Huffman code)**

| symbol | prob. | code | binary frac. |
|---|---|---|---|
| $Z$ | 0.5 | 1 | 0.1 |
| $Y$ | 0.25 | 01 | 0.01 |
| $X$ | 0.125 | 001 | 0.001 |
| $W$ | 0.125 | 000 | 0.000 |

The code words, if regarded as a binary fraction, are pointers to the particular interval being coded. In Huffman code, the code words point to the base of each interval. The average code length is $H = -\sum p(s) \log_2 p(s) \to$ optimal.

## 7 Scale-space representations

From an original signal $f(x)$ generate a parametric family of signals $f^t(x)$, where fine-scale information is successively suppressed.

### 7.1 Image pyramid

*Level 0:* $1 \times 1$, *Level 1:* $2 \times 2$, *Level 2:* $4 \times 4$, Level $J-1$: $N/2 \times N/2$, Level $J$ (base): $N \times N$.

### 7.2 Applications

*(1)* Search for correspondence: look at coarse scales, then refine with finer scales *(2)* Edge tracking: a "good" edge at a fine scale has parents at a coarser scale *(3)* Control of detail and computational cost in matching: e.g. finding stripes; terribly important in texture representation

### 7.3 Pyramids

**Gaussian pyramid** Smooth with gaussians, because "gaussian$^2$" = another gaussian. Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

**Laplacian Pyramid** ¡++¿ Shown the information added in gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

**Wavelet/QMF** Bandpassed representation, complete, but with aliasing and some non-oriented subbands. Recursive application of a two-band filter bank to the lowpass band of the provious stage yields octave band splitting.

**Steerable pyramid** Shown components at each scale and orientation separately, Non-aliased subbands. Good for texture and feature analysis.

### 7.4 Haar Transform

Two major sub-operations: *(1)* Scaling captures info at different frequencies *(2)* Translation captures info at different locations Can be represented by filtering and downsampling. Relatively poor energy compaction.

## 8 Optical Flow

### 8.1 Applications

*(1)* tracking *(2)* structure from motion *(3)* stabilization *(4)* compression *(5)* Mosaicing

### 8.2 Brightness constancy

**Definition of Optical Flow**
"Apparent motion of brightness patterns". Ideally, the optical flow is the projection of the three-dimensional velocity vectors on the image.

**Caution required** *(1)* Uniform, rotating sphere $\mathcal{OF} = 0$ *(2)* No motion, but changing lighting $\mathcal{OF} \neq 0$

### 8.3 Mathematical formulation

$I(x,y,t) =$ brightness at $(x,y)$ at time $t$.

**Brightness constancy assumption:**

$$I\left(\frac{dx}{dt}\delta t, y + \frac{dy}{dt}\delta t, t + \delta t\right) = I(x,y,t)$$

**Optical flow constraint equation:**

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

### 8.4 The aperture problem