Contents

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?

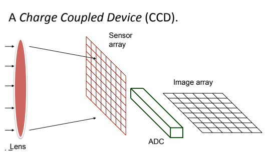
- A picture or pattern of a value varying in space and/or time
- Representation of a function $f : \mathbb{R}^n \to S$
- In digital form, e.g.: $I : \{1, \dots, X\} \times \{1, \dots, Y\} \to S$
- For greyscale CCD images, $n = 2$, $S = \mathbb{R}^+$

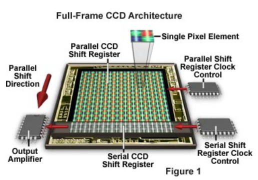
What is a pixel?

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

1.2 Image sources

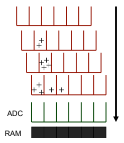
digital camera (CCD)





analog to digital Conversion

- 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, high 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 $= 2^{24} = 16.7$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

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

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" 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 masks

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

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.)

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

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. (on 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:

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

```
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);
```

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

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

$\spadesuit = i, j \in$ 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 $I \oplus S = \bigcup_{b \in S} I_b$.

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".

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 structuring 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 provides a stick-figure representing the rogion shap. Used in object recognition, in particular, character recognition. *Problems:* Definition of a maximal disc is poorly defined on a digital grid and is sensitive 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 $(2M+1) \times (2N+1)$,

$$I'(m,n) = f * (g * I(N(m,n))),$$

$$I''(m,n) = \sum_{j=-N}^{N} g(j)I(m,n-j),$$

$$I'(m,n) = \sum_{i=-M}^{M} f(i)I''(m-i,n),$$

$\to (2M+1) + (2N+1)$ 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:

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

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. → Neighbor's influence decreases monotonically. *(3)* Still one lobe in frequency domain. → 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

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon}\right)$$

$\frac{\partial f}{\partial x}$ which is obviously a convolution $(-1\ 1)$

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

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

$$A = \mathcal{I}(1),$$

$$A + C = \mathcal{I}(3),$$

$$A + B = \mathcal{I}(2),$$

$$A + B + C + D = \mathcal{I}(4).$$

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

Also possible along diagonal.

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

4 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 $E(p,q)$

$$\begin{aligned} &= \sum_{x,y=-\infty}^{\infty}[s(x,y) - t(x-p,y-q)]^2 \\ &= \sum_{x,y=-\infty}^{\infty}|s(x,y)|^2 + |t(x,y)|^2 \\ &\quad - 2\cdot\sum_{x,y=-\infty}^{\infty} s(x,y)\cdot t(x-p,y-q) \end{aligned}$$

Equivalently, maximize *area correlation*

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

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

where in the last step the Cauchy-Schwarz inequality was used. Equality ⇔

$s(x,y) = \alpha \cdot t(x-p,y-q)$ with

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

Diagram of template matcher

$s(x,y)$ search peak(s) → $t(-x,-y)$ object location(s) p, q → $r(x,y)$ 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 3×3 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. → do it in one operator Laplacian of Gaussian (LoG)

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

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 → 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 → 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

$$S(\Delta x,\Delta y) = \sum_{x,y\in\text{window}}\left[f(x,y) - f(x+\Delta x, y+\Delta y)\right]^2$$

Linear approximation for small Δx, Δy

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

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

Feature point extraction

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

Find points for which the following is large

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

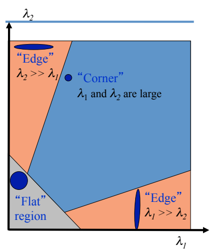
for $\|\Delta\| = 1$. i.e. maximize eigenvalues of M.

Keypoint detection Often based on eigenvalues λ_1, λ_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: $f(x,y) \to f(x,y) + c$ *(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,scale,orientation)

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

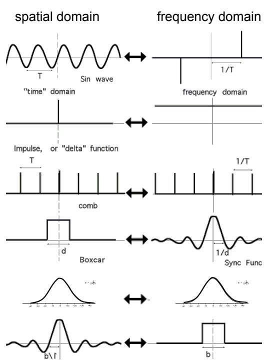
5.2 Definition

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

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

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

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

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 → Delta function

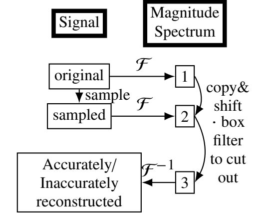
$S_{2D}\left(f(x,y)\right)$

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

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

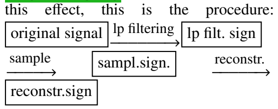
with S = Sample operator.

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:



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

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

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

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

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

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

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

Singularities are avoided by the rugelarization ε.

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 • Simplified 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:

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

or as a vector

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

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 MN. *(2)* Compute transform coefficients $\mathbf{c} = A\mathbf{f}$ 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 $\|\mathbf{c}\|^2 = \mathbf{c}^*\mathbf{c} = \mathbf{f}^*A^*A\mathbf{f} = \|\mathbf{f}\|^2$

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:

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

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

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

Eigenmatrix of autocorrelation matrix

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

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

where $\Lambda = \text{diag}(\lambda_0,\ldots,\lambda_{MN-1})$. *(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 ϕ are ordered according to decreasing eigenvalues. *(2)* Transform coefficients are pairwise uncorrelated

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

(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 $\mathbf{b} = I_J\mathbf{c}$, where I_J contain ones on the first J diagonal positions, else zeros. *(2)* Energy in first J coefficients for arbitrary transform A

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

$$= \text{tr}(I_J AR_{ff}A^* I_J) = \sum_{k=0}^{J-1} a_k^T R_{ff}\bar a_k$$

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

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

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

Differentiating L with respect to a_j yields necessary condition

$$R_{ff}\bar a_j = \lambda_j\bar a_j, \qquad \forall j < J$$

6.2 Basis images and eigenimages (EI)

For a unitary transform, the inverse transform $\mathbf{f} = A^*\mathbf{c}$ 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_{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 recognition 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 computatonal burden for many recognition techniques. Transform $\mathbf{c} = W\mathbf{f}$ 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

$$\underset{i}{\text{argmin}}\, D_i = \|I_i - I\|$$

Computionally 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

$$\hat I_i \mapsto p_i, \quad E^*\hat I_i = p_i.$$

$$\rightsquigarrow \hat I_i \approx Ep_i,$$

$$\rightsquigarrow I_i - I = \hat I_i - \hat I \approx E(p_i - p)$$

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

with closest rank-k approximation property of SVD. Approximate

$$\underset{i}{\text{argmin}}\, D_i = \|I_i - I\| \approx \|p_i - p\|.$$

6.3 Eigenfaces (EF)

Concatenate face pixels into "observation vector", $\mathbf{x}$.

EF 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 identification.

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 individual 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 at maximized.

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

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

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

$$F_{\text{opt}} = \underset{F}{\text{argmax}}\left(\frac{\det\left(FR_BW^*\right)}{\det(FR_WF^*)}\right),$$

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

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

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

$$R_B\mathbf{w}_i = \lambda_i R_W\mathbf{w}_i,\ i = 1,\ldots,k$$

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.

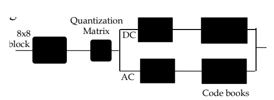
6.5 JPEG image compression

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

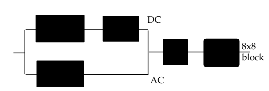
Concept Block-based discrete cosine transform (DCT)

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 fraction
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: 1×1, *Level 1:* 2×2, *Level 2:* 4×4, 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

In Visual Computing, people seem like to use

$$I_\spadesuit = \frac{\partial I}{\partial\spadesuit}, \quad u = \frac{\partial x}{\partial t}, \quad v = \frac{\partial y}{\partial t},$$

where the subindex means a derivative if and only if we are talking about I.

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:

$$I\left(\frac{\mathrm{d}x}{\mathrm{d}t}\delta t,\, y + \frac{\mathrm{d}y}{\mathrm{d}t}\delta t,\, t + \delta t\right)$$

$$= I(x,y,t)$$

Optical flow constraint equation:

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{\partial I}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial I}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{\partial I}{\partial t} = 0$$

8.4 The aperture problem

The motion of an edge seen through an aperture is (in some cases) inherently ambiguous. E.g. the edge is physically moving upwards, but the edge motion alone is consistent with many other possible motions, and in this case the edge e.g. appears to move diagonally.

8.5 Optical Flow meaning

Estimate of observed projected motion field. Not always well defined! Compare: 1 Motion Field (or Scene Flow), projection of 3-D motion field 2 Normal Flow: observed tangent motion 3 Optic Flow: Apparent motion of the brightness pattern: Apparent motion of the brightness pattern (hopefully equal to motion field) 4 Consider barber pole illusion

Planar motion Ideal motions of a plane, X, Y being the horizontal and vertical direction and Z normal to the image plane: 1 translation in X 2 translation in Z 3 rotation around Z 4 rotation around Y

8.6 Regularization: Horn & Schunck algorithm

The Horn-Schunck algorithm assumes smoothness in the flow over the whole image. thus, it tries to minimize distortions in flow and prefers solutions which show more smoothness. The flow is formulated as a global energy functional which is the sought to be minimized. This function is given for two-dimensional image streams as Eq. A.2: The associated ELE are

$$\frac{\partial L}{\partial\dot x} - \frac{\partial}{\partial x}\frac{\partial L}{\partial\frac{\partial\dot x}{\partial x}} - \frac{\partial}{\partial y}\frac{\partial L}{\partial\frac{\partial\dot x}{\partial y}} = 0,$$

$$\frac{\partial L}{\partial\dot y} - \frac{\partial}{\partial x}\frac{\partial L}{\partial\frac{\partial\dot y}{\partial x}} - \frac{\partial}{\partial y}\frac{\partial L}{\partial\frac{\partial\dot y}{\partial y}} = 0.$$

this gives

$$\frac{\partial I}{\partial x}\left(\frac{\partial I}{\partial x}\dot x + \frac{\partial I}{\partial y}\dot y + \frac{\partial I}{\partial t}\right) - \alpha^2\Delta\dot x$$

$$= 0,$$

$$\frac{\partial I}{\partial y}\left(\frac{\partial I}{\partial x}\dot x + \frac{\partial I}{\partial y}\dot y + \frac{\partial I}{\partial t}\right) - \alpha^2\Delta\dot y$$

$$= 0.$$

with $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$.

Remarks ① Coupled PDE solved using iterative methods and finite differences

$$\ddot{x} = \Delta \dot{x} - \lambda \left(\frac{\partial I}{\partial x}\dot{x} + \frac{\partial I}{\partial y}\dot{y} + \dot{I} \right) \frac{\partial I}{\partial x},$$

$$\ddot{y} = \Delta \dot{y} - \lambda \left(\frac{\partial I}{\partial x}\dot{x} + \frac{\partial I}{\partial y}\dot{y} + \dot{I} \right) \frac{\partial I}{\partial y}.$$

② More than two frames allow a better estimation of $\dot{I}$. ③ Information spreads from corner-type patterns. ④ Errors at boundaries ⑤ Example of *regularisiation*: selection principle for the solution of illposed problems.

8.7 Lucas-Kanade: Integrate over a Patch

The Lucas-Kanade method assumes that the displacement of the image contents between two nearby instants (frames) is small and approximately constant within a neighborhood of the point p under consideration. thus the optical flow equation can be assumed to hold for all pixels within a window centered at p. Namely, the local image flow (velocity) vector $(\dot{x}, \dot{y})$ must satisfy

$$\frac{\partial I(q_k)}{\partial x}\dot{x} + \frac{\partial I(q_k)}{\partial y}\dot{y} = -\frac{\partial I(q_k)}{\partial t}$$

for $k = 1, \ldots, n$ and q_k the pixels inside the window. These equations can be written in matrix form

$$A\mathbf{v} = \mathbf{b} \quad (8.1)$$

where $\mathbf{x} = (x\ y)^T$, $\quad \mathbf{v} = (\dot{x}\ \dot{y})^T$ and

$$A_{ij} = \frac{\partial I(q_i)}{\partial x_j}, \qquad \mathbf{b}_i = -\frac{\partial I(q_i)}{\partial t}.$$

Eq. 8.1 is overdetermined, so do compromise solution by the least squares principle Eq. A.3

8.8 Gradient-Based Estimation

Assume *brightness constancy*. Let $f_1(x)$ and $f_2(x)$ be 1D signals (images) at two time instants. Let $f_2 = f_1(x - \delta)$, where δ denotes translation.

$$\rightsquigarrow f_1(x) - f_2(x)$$

$$= \delta f_1'(x) + O\left(\delta^2\right)$$

$$\rightsquigarrow \tilde{\delta} = \frac{f_1(x) - f_2(x)}{f_1'(x)} \approx \delta \quad (8.2)$$

Assume displaced image well approximated by first-order Taylor series

$$I(\mathbf{x} + \mathbf{u}, t + 1) \quad (8.3)$$

$$\approx I(\mathbf{x}, t) + \mathbf{u} \cdot \nabla I(\mathbf{x}, t) + I_t(\mathbf{x}, t)$$

Insert Eq. 8.2 in Eq. 8.3 to get

$$\nabla I(\mathbf{x}, t) \cdot \mathbf{u} + I_t(\mathbf{x}, t) = 0. \quad (8.4)$$

This i s called the *gradient constraint equation*.

Intensity Conservation Tracking points of constant brightness can also be viewed as the estimation of 2D paths $\mathbf{x}(t)$ along which intensity is conserved:

$$I(\mathbf{x}(t), t) = c,$$

$$\rightsquigarrow \frac{\mathrm{d}}{\mathrm{d}t} I(\mathbf{x}(t), t) = 0$$

$$\rightsquigarrow \nabla I \cdot \mathbf{u} + I_t = 0. \quad (8.5)$$

$\rightsquigarrow$ gradient-constraint equation Eq. 8.4.

Least-Squares estimation One cannot recover $\mathbf{u}$ from one gradient constraint since Eq. 8.4 is one equation with two unknowns, u_1 and u_2. The intensity gradient constrains the flow to a one parameter family of volecities along a line in *velocity space*. One can see from Eq. 8.4 that this line is perpendicular to ∇I and its perpendicular distance from the origin is $|I_t| / \|\nabla I\|$.

One common way to further constrain $\mathbf{u}$ is to use gradient constraints from nearby pixels, assuming they share the same 2D velocity. With many constraints there may be no velocity that simultaneously satisfies them all, so instead we find the velocity that minimizes the constraint errors. The least-squares (LS) estimator minimizes the squared errors:

$$E(\mathbf{u}) = \sum_{\mathbf{x}} g(\mathbf{x})[\mathbf{u} \cdot \nabla I(\mathbf{x}, t) + I_t(\mathbf{x}, t)]^2, \quad (8.6)$$

where $g(\mathbf{x})$ is a weighting function that determines the *support* of the estimator (the region within which we combine constraints). It is common to let $g(\mathbf{x})$ be Gaussian in order to weight constraints in the center of the neighborhood more highly, giving them more influence. The 2D velocity $\hat{u}$ that minimizes $E(\mathbf{u})$ is the least squares flow estimate.

The minimum of $E(\mathbf{u})$ can be found from its critical points, where its derivatives with respect to $\mathbf{u}$ are zero; i.e.,

$$\frac{\partial E(u_1, u_2)}{\partial u_1}$$

$$= \sum_{\mathbf{x}} g(\mathbf{x}) \left[u_1 I_x^2 + u_2 I_x I_y + I_x I_t \right]$$

$$= 0$$

$$\frac{\partial E(u_1, u_2)}{\partial u_2}$$

$$= \sum_{\mathbf{x}} g(\mathbf{x}) \left[u_2 I_y^2 + u_1 I_x I_y + I_y I_t \right]$$

$$= 0$$

These equations may be rewritten in matrix form:

$$M\mathbf{u} = \mathbf{b} \quad (8.7)$$

$$M = \begin{pmatrix} \sum g I_x^2 & \sum g I_x I_y \\ \sum g I_x I_y & \sum g I_y^2 \end{pmatrix},$$

$$\mathbf{b} = -\begin{pmatrix} \sum g I_x I_t \\ \sum g I_y I_t \end{pmatrix}.$$

When M has rank 2, then the LS estimate is $\hat{u} = M^{-1}\mathbf{b}$.

Implementation Issues Usually we wish to estimate optical flow at every pixel, so we should express M and $\mathbf{b}$ as functions of position $\mathbf{x}$, i.e., $M(\mathbf{x})\mathbf{u}(\mathbf{x})$. Note that the elements of M and $\mathbf{b}$ are local sums of products of image derivatives. An effective way to estimate the flow field is to first compute derivative images through convolution with suitable filters. Then, compute their products (I_x^2, $I_x I_y$, I_y^2, $I_x I_t$ and $I_y I_t$), as required by Eq. 8.7. These quadratic images are then convolved with $g(\mathbf{x})$, to obtain the elemnts of $M(\mathbf{x})$ and $\mathbf{b}(\mathbf{x})$.

In practice, the image derivatives will be approximated using numerical differentiation. It is important to use a consistent approximation scheme for all three directions.

8.9 Aperture Problem

When $\mathbf{M}$ in Eq. 8.7 is rank deficient one cannot solve for $\mathbf{u}$. This is often called the aperture problem as it invariably occurs when support $g(x)$ is sufficiently local. However, the important issue is not the width of the image structure. However, the important issue is not the width of support, but rather the dimensionality of the image structure. Even for large regions, if the image is one-dimensional then M will be singular. When each image gradient within a region has the same spatial direction, it is easy to see that rank $M = 1$. Moreover, note that a single gradient constraint only provides the normal component of $\mathbf{u}$,

$$\hat{\mathbf{u}} = \frac{-I_t}{\|\nabla I\|} \frac{\nabla I}{\|\nabla I\|}$$

8.10 Iterative Optical Flow Estimation

Equation 8.7 provides an optimal solution, but not to our original problem. Higher-order terms ignored!

$$|\tilde{\delta} - \delta| = \frac{\delta^2}{\left|f_1'(x)\right|^2} \left|f_1''(x)\right| + O\left(\delta^3\right)$$

For a sufficiently small displacement, and bounded $|f_1''/f_1'|$, we expect reasonably accurate estimates. This suggests a form of Gauss-Newton optimization in which we use the current estimate to *undo* the motion, and then we reapply the estimator to the *warped* signals to find the residual motion. This continues until the residual motion is sufficiently small.

In 2D, given an estimate of the optical flow field $\mathbf{u}^0$, we create a *warped* image sequence $I^0(\mathbf{x}, t)$:

$$I^0(\mathbf{x}, t + \delta t) = I\left(\mathbf{x} + \mathbf{u}^0 \delta t, t + \delta t\right), \quad (8.8)$$

where δt is the time between consecutive frames. Assuming that $\mathbf{u} = \mathbf{u}^0 + \delta\mathbf{u}$, from brightness constancy and Eq. 8.8 we get

$$I^0(\mathbf{x}, t) = I^0(\mathbf{x} + \delta\mathbf{u}, t + 1)$$

If $\delta\mathbf{u} = 0$, then clearly I^0 would be constant through time (assuming brightness constancy). Otherwise, we can estimate the residual flow using

$$\delta\hat{u} = M^{-1}\mathbf{b} \quad (8.9)$$

where M and $\mathbf{b}$ are computed by taking spatial and temporal derivatives (differences) of I^0. The refined optical flow estimate then becomes

$$\mathbf{u}^1 = \mathbf{u}^0 + \delta\hat{u}.$$

In an iterative manner, this new flow estimate is then used to rewarp the original sequence, and another residual flow can be estimated.

This iteration yields a sequence of approximate objective functions that converge to the desired objective function. At iteration j, given the estimate $\mathbf{u}^j$ and the warped sequence I^j, our desired objective function is

$$E(\delta\mathbf{u}) = \sum_{\mathbf{x}} g(\mathbf{x}) \left[I(\mathbf{x}, t) - I\left(\mathbf{x} + \mathbf{u}^j + \delta\mathbf{u}, t + 1\right) \right]^2$$

$$= \sum_{\mathbf{x}} g(\mathbf{x}) \left[I^j(\mathbf{x}, t) - I^j\left(\mathbf{x} + \delta\mathbf{u}^j, t + 1\right) \right]^2$$

$$=: \tilde{E}(\delta\mathbf{u}). \quad (8.10)$$

The gradient approximation to the difference to $E(\delta\mathbf{u})$ gives the approximate objective function $\tilde{E}$. From $\left|\tilde{d} - \delta\right|$ one can show that $\tilde{E}$ approximates E to the second-order in the magnitude of the residual flow, $\delta\mathbf{u}$. The approximation error vanishes as $\delta\mathbf{u}$ is reduced to zero. The iterative refinement with rewarping reduces the residual motion at each iteration so that the approximate objective function converges to the desired objective function, and hence the motion estimate converges to the optimal LS estimate $E(\delta\mathbf{u})$.

The most expensive step at each iteration is the computation of image gradients and the matrix inverse in 8.9. One can, however, formulatio the problem so that the spatial image derivatives used to form M are taken at time t, and as such, do not depend on the current flow estimate $\mathbf{u}^j$. To see this, note that the spatial derivatives are computed at time t which leads to $I(\mathbf{x}, t) = I^j(\mathbf{x}, t)$. Of course $\mathbf{b}$ in 8.9 will always depend on the warped image sequence and must be recomputed at each iteration. In practice, when M is not recomputed from the warped sequence then the spatial and temporal derivatives will not be centered at the same location in (x, y, t) and hence more iterations may be needed. =======

$$\nabla I(\mathbf{x}, t) \cdot \mathbf{u} + I_t(\mathbf{x}, t) = 0.$$

This i s called the *gradient constraint equation*.

8.11 Pyramid/Coarse-to-fine

Limits of the (local) gradient method: ① Fails when intensity structure within window is poor ② Fails when displacement is large (typical operating range is motion of 1 pixel per iteration!). Linearization of brightness is suitable only for small displacements.
③ Brightness is no strictly constant in images. Actually less problematic than it appears, since we can pre-filter images to make them look similar.

In practice, our images have temporal sampling rates lower than required by the sampling theorem to uniquely reconstruct the continuous signal. AS a consequence, temporal aliasing is a common problem in motion estimation.

The spectrum of a translating signal is confined to a plane through the origin in the frequency domain. That is, if we construct a space-time signal $f(\mathbf{x}, t)$ by translating a 2D signal $f_0(\mathbf{x})$ with velocity $\mathbf{u}$, i.e., $f(\mathbf{x}, t) = f_0(\mathbf{x} - \mathbf{u}t)$, one can show that the space-time Fourier transform of $f(\mathbf{x}, t)$ is given by

$$F(\omega_x, \omega_y, \omega_t)$$

$$= F_0(\omega_x, \omega_y)$$

$$\cdot \delta\left(u_1\omega_x + u_2\omega_y + \omega_t\right), \quad (8.11)$$

where F_0 is the 2D Fourier transform of f_0. Eq. 8.11 shows that the spectrum is nonzero only on a plane, the orientation of which gives the velocity. When the continuous signal is sampled in time, replicas of the spectrum are introduced at intervals of $2\pi/T$ radians, where T is the time between frames. It is easy to see how this causes problems; i.e., the derivative filterrsl may be more sensitive to the spectral replicas at high spatial frequencies than to the original spectrum on the plane through the origin.

Optical flow can be estimated at the coarsest scale of a Gaussian pyramid, where the image is significantly blured, and the velocity is much slower (due to subsampling). The coarse-scale estimate can be used to warp the next (finer) pyramid level to *stabilize* its motion. Since the velocities after warping are slower, wider low-pass frequency band will be free of aliasing. One can therefore use derivatives at the finer scale to estimate the residual motion. This coarse-to-fine estimation continues until the finest level of the pyramid (the original image) is reached. Mathematically, this is identical to iterative refinement except that each scal's estimate must be up-sampled and interpolated before warping the next finer scale.

While widely used, coarse-to-fine methods have their drawbacks, usually stemming from the fact that fine-scale estimates can only be as reliable as their coarse-scale precursors; a poor estimate at one scale provides a poorr initial guess at the next finer scale, and so on. That said, when aliasing does occur, one must use some mechanism such as coarse-to-fine estimation to avoid local minima in the optimization.

8.12 Robust Motion Estimation

The LS estimator is optimal when the gradient constraint errors, i.e.,

$$e(\mathbf{x}) := \mathbf{u} \cdot \nabla I(\mathbf{x}, t) + I_t(\mathbf{x}, t)$$

are mean-zero Gaussian, and the errors in different constraints are independent and identically distributied (IID.). Not surprisingly, this is a fragile assumption. for example, brightness constancy is often violated due to changing surface orientation, specularities reflections, or time-varying shadows. When there is significant depth variation in the scene , the constant motion model will be extremely poor, especially at occlusion boundaries.

LS estimators are not suitable when the distribution of gradient constraint errors is heavy-tails, as they are sensitive to small numbers of measurement outliers. It is therefore often crucial that the quadratic estimation in Eq. 8.6 be repalced by a robust estimator, $\rho(\cdot)$, which limits the influence of constraints with larger errors:

$$E(\mathbf{u}) = \sum_{\mathbf{x}} g(\mathbf{x}) \rho(e(\mathbf{x}), \sigma).$$

For example the redescending Geman-McClure estimator

$$\rho(e, \sigma) = e^2 / \left(e^2 + \sigma^2\right),$$

where σ^2 determines the range of constraint errors for which influence is reduced.

8.13 Motion Models

Thus far we have assumed that the 2D velocity is constant in local neighbourhoods. Nevertheless, even for small regions this is often a poor assumption. We now consider generalizations to more interesting motion models.

Affine Model General first-order affine motion is usually a better model of local motion than a tranllational model. An affine velocity field centered at location $\mathbf{x}_0$ can be expressed in matrix form as

$$\mathbf{u}(\mathbf{x}; \mathbf{x}_0) = A(\mathbf{x}; \mathbf{x}_0)\, \mathbf{c}, \quad (8.12)$$

where$(c_1\ c_2\ c_3\ c_4\ c_5\ c_6)^T$ are the motion model parameters, and

$$A(\mathbf{x}; \mathbf{x}_0)$$

$$= \begin{pmatrix} 1 & 0 & x - x_0 & y - y_0 & 0 & 0 \\ 0 & 1 & 0 & 0 & x - x_0 & y - y_0 \end{pmatrix}.$$

From Eq. and we get the gradient constraint equation

$$\nabla I(\mathbf{x}, t) A(\mathbf{x}; \mathbf{x}_0)\mathbf{c} + I_t(\mathbf{x}, t) = 0,$$

for which the LS estimate for the neighbourhood has the form

$$\hat{c} = M^{-1}\mathbf{b} \quad (8.13)$$

where now M and $\mathbf{b}$ are given by

$$M = \sum_{\mathbf{x}} g A^T \nabla I^T \nabla I A,$$

$$\mathbf{b} = -\sum_{\mathbf{x}} g A^T \nabla I^T I_t.$$

When M is rank deficient thecre is insufficient image structure to estimate the six unknowns. Affine models often require larger support than constant models, and one may need a robust estimator instead of the LS estimator.

Iterative refinement is also straightforward with affine motion models. Let the optimal affine motion be $\mathbf{u} = A\mathbf{c}$, and let affine estimate at iteration j be $\mathbf{u}^j = A\mathbf{c}^j$. Because the flow is linear in the motion parameters, it follows that $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}^j$ and $\delta\mathbf{c} = \mathbf{c} - \mathbf{c}^j$ satisfy

$$\delta\mathbf{u} = A\delta\mathbf{c}.$$

Accordingly, defining $I^j(\mathbf{x}, t)$ to be the original sequence $I(\mathbf{x}, t)$ warped by $\mathbf{u}^j$ as in Eq. 8.8 we use the same LS estimator as in Eq. 8.13, but with I and $\hat{c}$ replaced by I^j and '$\delta\hat{c}$.

8.14 Low-order Parametric Deformations

There ary many other polynomial and rational deformations that make useful motion models. *Similarity deformations*, comprising translation (d_1, d_2), 2D rotation θ, and uniform scaling by s are a special case of the affine model, but still very useful in practice. In a neighborhood centred at $\mathbf{x}_0$ it has the form as Eq. 8.13, but with $\mathbf{c} = (d_1, d_2, s\cos\theta, s\sin\theta)^T$ and

$$A(\mathbf{x}; \mathbf{x}_0) = \begin{pmatrix} 1 & 0 & x - x_0 & -y + y_0 \\ 0 & 1 & y - y_0 & x - x_0 \end{pmatrix}.$$

8.15 Global Smoothing

While area-based regression is commonly used, some of the earliest formulations of optiacl flow estimation assumed smoothness through nonparametric motion models,rather than an explicit parametric model in each local neighbourhood. One such energy functional was proposed by Horn and Schunck in Eq. A.2. A key advantage of global smoothing is that it enable propagation of information over large distances in the image. In image regions of nearly uniform intensity, such as a blank wall or tabletop, local methods will often yield singular (or poorly conditioned) systems of euations. Global methods can *finn in* the optical flow from nearby gradient constrains.

The equation above can be minimized directly with discrete approximations to the integral and the derivatives. This yields a large system of linear equations. The main disadvantage of global methods is computational efficiency. Another problem is in the setting of the *regularization parameter* λ that determines the amount of desired smoothing.

8.16 Probabilistic Formulations

One problem with the above estimators is that, althought they provide useful estimates of optical flow, they do not provide confidence bounds. Nor do they show how to incorporate any prior information one might have bout motion to further constrain the estimates. As a result, one may not be able to propagate flow estimates from one time to the next, nor know how to weight them when combining flow estimates from different information sources. THese issues can be adressed with a probabilistic formulation.

The cost function 8.10 has a simple probabilistic interpretation. Up to normalization constants, it corresponds to the log likelihood of a velocity under the assumption that intensity is conserved up to gaussian noise.

$$I(\mathbf{x}, t) = I(\mathbf{x} + \mathbf{u}, t + 1) + \eta.$$

If we assume that the same velocity $\mathbf{u}$ is shared by all pixels within a neighbourhood, that η is white Gaussian noise with standard deviation σ, and uncorrelated at different pixels, we obtain the conditional density

$$p(I \mid \mathbf{u}) = \propto e^{E(\mathbf{u})/2\sigma^2}.$$

=======

8.17 Parametric motion models

Global miton models offer:
① More constrained solutions than smoothness (Horn-Schunck)
② Integration over a large area than a translation-only model can accomodate (Lucas-Kanade)

9 Video Compression

9.1 Perception of motion

Perception of motion: Human visual system is specifically sensitive to motion. Eyes follow motion automatically. Some distortions are not as percoivable as in image coding (would be if we froze frame). No good psycho-visual model avaivable. Vusal perception is limited to $<$ 24 Hz. Asuccession of images will be perceived as continuous if frequency is sufficiely high. Cinema 2424 Hz, TV 25 Hz or 50 Hz. We still nee to avoid aliasing (wheel effect). High-rendering framerates desired in computer games (needed due to absence of motion blur). Flicker can be perceived up to $>$ 60 Hz in particular in periphery. Issue addressed by 100 Hz TV.

9.2 Interlaced video format

Two temporarlly shifted half images, increase of frequency 25 Hz $\to$ 50 Hz. Reduction of spatial resolution. Full image representation: progressive.

9.3 Why compress video?

Raw HD TV signal $720p@50$ Hz:

$$1280 \cdot 720 \cdot 50 \cdot 24 \text{ bits/s}$$

$$= 1\,105\,920\,000 \text{ bits/s} > 1 \text{ Gb/s}$$

Only 20 Mb/s HDTV channel bandwidth requires compression of factor of 60 (0.4 bits/pixel on average)

9.4 Lossy video compression

Take advantage of redundancy. Spatial correlation between neighboring pixels. Temporal correlation between frames. Drop perceptuall unimportant details.

Temporal Redundancy Take advantage of similarity between successive frames

Temporal processing
Usually high frame rate: Significant temporal redundancy. Possible representations along temporal dimension:
① Transform/subband methods: Good for textbook case of constant velocity uniform global motion. Inefficient for nonuniform motion, i.e. real-world motion. Requires large number of frame stores which leads to delay. (Memory cost may alse be an issue.) Is ineffective for many scene changes or high motion.
② Predictive methods: Good performance using only 2 frame stores. However, simple frame differencing is not enough…

Goal Exploit the temporal redundancy

Predict current frame based on previously coded frames

Types of coded frames:
① I-frame: Intra-coded frame, coded independently of all other frames.
② P-frame: Predictively coded frame, coded based on previously coded frame I or P. Can send motion vector plus changes.
③ B-frame: Bi-directionally predicted frame, coded based on both previous and future coded frames I and P. In case something is uncovered.

Motion-compensated prediction
Simple frame differencing *fails* when there is motion. Must account for motion.
$\to$ Motion-compensated (MC) prediction. MC-prediction generally provides significant improvements. Questions: How can we estimate motion? How can we form MC-prediction?

Ideal situation
① Partition video into moving objects
② describe object motion $\to$ Generally very difficult

Practical approach Block-Matching Motion Estimation:
① Partition each frame into blocks, e.g. 16×16 pixels
② Describe motion of each block
$\to$ No object identification required and good, robust performance.

9.5 Block-matching motion estimation

Assumptions: ① Translational motion within block:

$$f(n_1, n_2, k_{\text{cur}})$$

$$= f(n_1 - mv_1, n_2 - mv_2, k_{\text{ref}}).$$

ME Algorithm ① Divide current frame into non-overlapping $N_1 \times N_2$ blocks.
② For each block, find the best matching block in reference frame.

9.5.1 Determining the best matching block

For each block in the current frame, search for best matching block in the reference frame.

Metrics for determining "best match":

MSE

$$= \sum_{n_1, n_2 \in \text{Block}} [f(n_1, n_2, k_{\text{cur}}) - f(n_1 - mv_1, n_2 - mv_2, k_{\text{ref}})]^2.$$

MAE

$$= \sum_{n_1, n_2 \in \text{Block}} |f(n_1, n_2, k_{\text{cur}}) - f(n_1 - mv_1, n_2 - mv_2, k_{\text{ref}})|.$$

Candidate blocks All blocks in, e.g. $(\pm 32, \pm 32)$ pixel area

Strategies for searching candidate blocks for best match.
① Full search: Examine all candidate blocks
② Partial (fast) search: Examine a carefully selected subset.

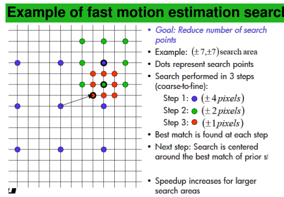
Motion vector Estimate of motion for best matching block.

9.6 Motion vector and motion vector field

Motion vector Expresses the *relative horizontal and vertical offsets* (mv_1, mv_2), or motion, of a given block from one frame to another.

Motion vector field Collection of motion vectors for all the blocks in a frame.

Example of fast motion estimation search



Motion Vector Presision
• Motivation: Motion is not limited to integer-pixel offsets. However, video is only known at discrete pixel locations. To estimate sub-pixel motion, frames must be spatially interpolated.
• Fractional MVs are used to represent the sub-pixel motion.
• Improved performance (extra complexity is worthwhile)
• Half-pixel ME used in most standards: MPEG-1/2/4
• Why are half-pixel motion vectors better? They can capture half-pixel motion. Averaging effect (from spatial interpolation) reduces prediction error $\to$ Improved prediction. For noisy sequences, averaging effect reduces noise $\to$ Improved compression.

9.6.1 Practical Half-Pixel Motion Estimation Algorithm

Half-pixel ME (coarse-fine) algorithm:
① Coarse step: Perform integer motion estimation on blocks; find best integer-pixel MV
② Fine step: Refine estimate to find best half-pixel MV
ⓐ Spatially interpolate the selected region in reference frame
ⓑ Compare current block to interpolated reference frame block.
ⓒ Choose the integer or half-pixel offset that provides best match Typically, bilinear interpolation is used for spatial interpolation

9.7 Block Matching Algorithm

Issues Block size, search range, motion vector accuracy

Estimate Done typically only from luminance

Advantages ① Good, robust performance for compression.
② Resulting motion vector field is easy to represent (one MV per block) and useful for compression.
③ Simple, periodic structure, easy VLSI implementations

Disadvantages
① Assumes translational motion model $\to$ Breaks down for more complex motion.
② Often produces blocking artifacts (OK for coding with Block DCT)

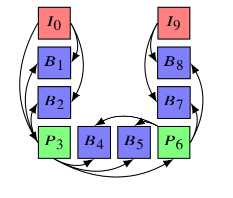
Bidirectional MC prediction is uset to estimate a block in the current frame from a block in:
① Previous frame
② Future frame
③ Average of ablock from the previous frame and a block from the future frame
④ Neither, i.e. code current block without prediction

Example: Prediction with P- and B-frames
① Motion compensated prediction: Predict the current frame based on reference frame(s) while compensating for the motion.
② Examples of block-based motion-compensated prediction (P-frame) and bi-directional prediction (B-frame).

9.8 Frame types

Main addition over image compression: Exploit the temporal redundancy. Predict current frame based on previously coded frames. Three types of coded frames:
① *I-frame*: Intra-coded frame, coded independently of all other frames
② *P-frame*: Predictively coded frame, coded based on previously coded frame
③ *B-frame*: Bi-directionally predicted frame, coded based on both previous and future coded frames.

MPEG Group of Pictures (GOP)



Starts with an I-frame, ends with frame right before next I-frame. "Open" ends in B-frame, "closed" in P-frame. MPEG Encoding a parameter, but "typical":

$$IBBPBBPBBI,$$

$$IBBPBBPBBPBBI.$$

Periodic I-frames enable random access into the coded bitstream. Parameters:
① Spacing between I frames
② Number of B frames between I and P frames

Example compression performance
I: $\frac{1}{7}$, P: $\frac{1}{20}$, B: $\frac{1}{50}$, Average: $\frac{1}{27}$.

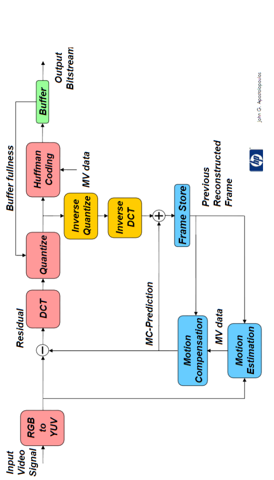
9.9 Summary of Temporal Processing

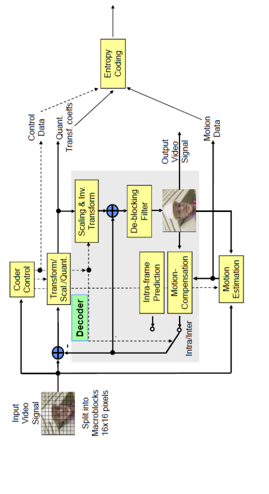
① Use MC-prediction (P and P frames) to reduce temporal redundancy.
② MC-prediction usually performs well; In compression have a second change to recover when it performs badly.
③ MC-prediction yields
ⓐ Motion vectors
ⓑ MC-prediction error or residual $\to$ Code error with conventional image coder
④ Sometimes MC-prediction may *perform badly*
ⓐ Examples: complex motion, new imagery (occlusions)
ⓑ Approach: *1.* Identify frame or individual blocks where prediction fails
2. Code without prediction

9.10 Basic Video Compression Architecture

① Exploiting the reduncancies:
ⓐ Temporal: MC-prediction (P and B frames)
ⓑ Spatial: Block DCT
ⓒ Color: color space conversion

2 Scalar quantization of DCT coefficients
3 Zigzag scanning, runlength and Huffman coding of the nonzero quantized DCT coefficients



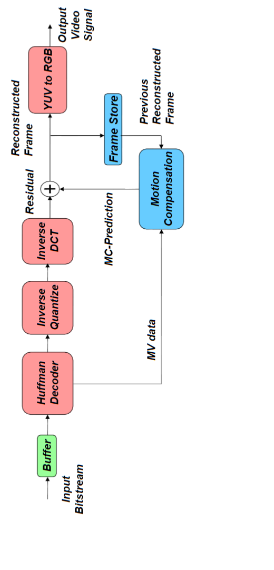


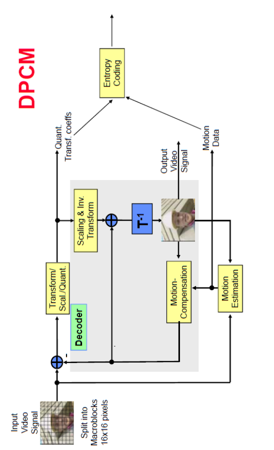
Transimission chain Content Creation → Transmission → Consumption. ⇒ Variable needs!

9.11 Scalable Video Coding

Produces *different layers with prioritized importance*. *Prioritized importance is key* for a variety of applications:
1 *Adapting* to different bandwidths, or client resources such as spatial or temporal resolution or computational power.
2 *Facililitates error-resilience* by explicitly identifying most important and less important bits.
Procedure:
1 Decompose video into *multiple layers of prioritized importance*
2 Code layers into *base and enhancement* bitstreams
3 Progressively combine *one or more bitstreams* to produce *different levels of video quality*.
Example of scalable coding with base and two enhancement layers: Can produce three different qualities: 1 Base layer 2 Base + Enh1 layers 3 Base + Enh1 + Enh2 layers Scalability with respect to: Spatial or temporal resolution, bit rate, computation, memory.

Example • Encode image/video into three layers: Base, Enh1, Enh2 • Low-bandwidth receiver: Send only Base layer. • Medium-bandwidth receiver: Send Base & Enh1 layers • High-bandwidth receiver: Send all three layers Base, Enh1, Enh2. • Can adapt to different clients and network situations • Three basic types of scalability (refine video quality along three different dimensions):
a Temporal scalability → Temporal resolution
b Spatial scalability → Spatial resolution
c SNR (quality) scalability → Amplitude resolution
• Each type of scalable coding provides scalability of one dimension of the video signal. Can cambine multiple types of scalability to provide scalability along multiple dimensions





Temporal Scalability based on the use of *B-frames* to refine the *temporal resolution* B-frames are dependent on other frames. Howere, no other frame depends on a B-frame. Each B-frame may be discarded without affecting other frames.

Spatial scalability Based on refining the *spatial resolution Base layer* is *low resolution* version of video. *Enh1* contains coded *difference* between upsampled base layer and original video. Also called pyramid coding.

SNR scalability Based on refining the *amplitude resolution Base layer* uses a *coarse quantizer*. *Enh1* applies a *finer quantizer* to the difference between the original DCT coefficients and the coarsely quantized base layer coefficients.

9.12 Standards

Goal *Ensuring interoperability*: Enabling communication between devices made by different manufacturers. Promoting a technology or industry. Reducing costs.

Scope of standardization Not the encoder, not the decoder. Just the *bitstream syntax* and the *decoding process* (e.g. use IDCT but not how to implement the IDCT) This enables improved encoding and decoding strategies to be employed in a standard-compatible manner.

9.13 Quality measure

objective: PSNR
• Error for one pixel, difference between original and decoded value
$e(v,h) = \tilde{x}(v,h) - x(v,h)$
• Mean-squared-Error, MSE e.g. over an image

$$e_{\mathrm{MSE}} = \sqrt{\frac{1}{N\cdot M}\sum_{v,h=1}^{v=N,h=M} e^2(v,h)}$$

• Peak-Signal-to-Noise-Ratio
PSNR $= [\max x]^2 / e^2_{\mathrm{MSE}}$
E.g. $x = 2^K$ or 255. One can use a log-scale like dB.

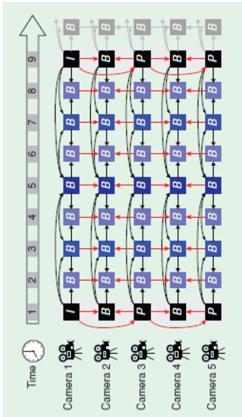
MPEG Structure MPEG codes video in *a hierarchy of layers*. The sequence layer is not shown.

MPEG-2 *Goal:* To enable more efficient implementations for different applications (interoperability points)
• Profile: Subset of the tools applicable for a family of applications
• Level: bounds on the complexity for any profile

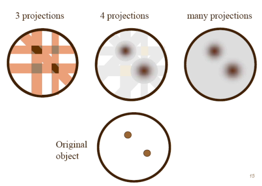
MPEG-4 Natural Video Coding Extension of MPEG-1/2-type algorithms to code arbitrarily shaped objects.

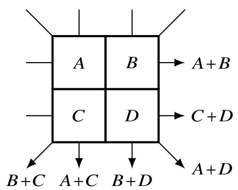
Sprite Coding Background prediction. Hypothesis: Same background exists for many frames, changes resulting from camera motion and occlusions. One possible coding strategy: 1 Code & transmit entire sprite *once*
2 Only transmit camera motion parameters for each subsequent frame Significant coding gain for some scenes.

3D TV Requeires the transmission of multiple views (two or more). Opportunity to do better than simulateously broadcast multiple views. Besides intra-frame and temporal redundancy, also *ex ploit inter-view redundancy*.



10 Radon Transform





$$\begin{bmatrix}1&1&0&0\\1&0&1&0\\1&0&0&1\\0&1&1&0\\0&1&0&1\\0&0&1&1\\1&1&0&0\end{bmatrix}\begin{bmatrix}A\\B\\C\\D\end{bmatrix} = \begin{bmatrix}A+B\\A+C\\A+D\\B+C\\B+D\\C+D\end{bmatrix}$$

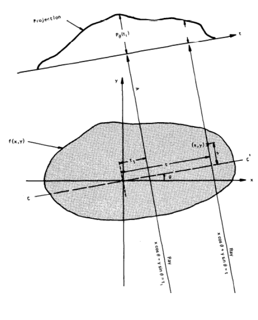
$$\Leftrightarrow K\mathbf{x} = \mathbf{b}$$

$$\rightsquigarrow K^T K\mathbf{x} = K^T\mathbf{b}$$

$$\rightsquigarrow \left[K^T K\right]^{-1} K^T\mathbf{b} = \mathbf{x}$$

Over-determined non-square matrix K. Larger problems must be solved iteratively using standard methods for solving large matrix operation problems.

10.1 Definition



We will use teh coordinate system defined in the figure above to describe line integrals and projections. In this example the object is represented by a two-dimensional function $f(x,y)$ and each line integral by the (θ,t) parameters. The equation of line AB in the figure above is

$$x\cos\theta + y\sin\theta = t$$

and we will use this relationship to define line integral P_θ as

$$P_\theta(t) = \int_{(\theta,t)\text{line}} \mathrm{d}s\, f(x,y)$$

Using a delta function, this can be rewritten as

$$P_\theta(t) = \int_{\mathbb{R}^2} \mathrm{d}x\mathrm{d}y\, f(x,y) \cdot \delta(x\cos\theta + y\sin\theta - t)$$

10.2 Fourier Slice Theorem

Two-dimensional Fourier transform of the object function

$$F(u,v) = \int_{\mathbb{R}^2} \mathrm{d}x\mathrm{d}y\, f(x,y) \cdot e^{-j2\pi(ux+vy)}$$

Likewise, define a projection at an angle θ, $P_\theta(t)$, and its fourier transform by

$$S_\theta(w) = \int_{\mathbb{R}^2} \mathrm{d}t\, P_\theta(t) e^{-j2\pi wt}$$

The simplest example of the Fourier Slice Theorem is given for a projection at $\theta = 0$. First, consider the Fourier transform of the object along the line in the frequency domain given by $v = 0$. The Fourier transform integral now simplifies to $F(u,0)$

$$= \int_{\mathbb{R}^2} \mathrm{d}x\mathrm{d}y\, f(x,y) e^{-j2\pi ux}$$

$$= \int_{-\infty}^{\infty} \mathrm{d}x\, P_{\theta=0} e^{-j2\pi ux}.$$

This represents the 1D FT of the projection $P_{\theta=0}$; thus we have the following relationship between the vertical projection and the 2D transform of the object function:

$$F(u,0) = S_{\theta=0}(u)$$

This is the simplest form of the Fourier Slice Theorem.

10.3 Reconstruction

General The inverse of the Radon transform can be used to reconstruct the original density from tme scattering data, and thu it forms the mathematical underpinning for tomographic reconstruction, also knows asimage reconstruction.

Sinogram The Radon transform data is often called a *sinogram* because the Radon transform of a Dirac delta function is a distribution supported on the graph of a sine wave. Consequently the Radon transform of a number of small objects appears graphically as a number of blurred sine waves with different amplitudes and phases.

Filtered backprojection algorithm
1 Sum for each of the K angles, θ, between 0° and 180°.
2 Measure the projection, $P_\theta(t)$.
3 Multiply it by the weighting function $2\pi|w|/K$.
4 Sum over the image plane the inverse Fourier transforms of the filtered porjections (the backprojection process). The weighting function weights the values outside such that the final weight of the whole body is uniform.

This kind of filter removes the noise around the reconstructed object. In some application the scanner can't cover the full angle 0° to 180° so the reconstruction has artefacts like lines in different angles.

11 Image denoising

$$\mathbf{y} = \mathbf{x}_{\mathrm{orig}} + \mathbf{w}$$

$$\text{measurements} = \text{original image} + \text{noise}$$

11.1 Sparse representations for image restoration

MAP estimation Energy minization problem

$$E(\mathbf{x}) = \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 + \Pr(\mathbf{x})$$

$$= \text{relation to measurements} + \text{image model } (-\log \text{prior})$$

Some *classical priors*:
1 Smoothness $\lambda\|\mathcal{L}x\|_2^2$
2 Total variation $\lambda\|\nabla\mathbf{x}\|_1^2$
3 MRF priors etc.

11.2 Sparse linear model

Signal $\mathbf{x} \in \mathbb{R}^m$

Dictionary $D = [\mathbf{d}_1, \dots, \mathbf{d}_p] \in \mathbb{R}^{m\times p}$ be a set of normalized "basis vectors".

sparse vector & code D is "adapted" to $\mathbf{x}$ if it can represent it with a few basis vectors - that is, there exists a *sparse vector* $\boldsymbol{\alpha}$ in $\mathbb{R}^p$ such that

$$\mathbf{x} \approx D\boldsymbol{\alpha}.$$

We call $\boldsymbol{\alpha}$ the *sparse code*.

$$\begin{pmatrix}x_1\\ \vdots\\ x_m\end{pmatrix} \approx \left(\mathbf{d}_1 \middle| \cdots \middle| \mathbf{d}_p\right)\begin{pmatrix}\alpha_1\\ \vdots\\ \alpha_p\end{pmatrix}$$

2D example Complete basis has 2 elements. Overcomplete sparse dictionary with 4 elements allows sparse representation.

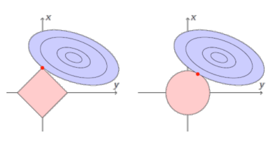
Idea A dictionary can be good for representing a class of signals, but not for representing white Gaussian noise.

Sparse Decomposition problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \underbrace{\frac{1}{2}\|\mathbf{x}-D\boldsymbol{\alpha}\|_2^2}_{\text{data fitting term}} + \underbrace{\lambda\psi(\boldsymbol{\alpha})}_{\text{sparsity-inducing regularization}}$$

ψ induces sparsity in $\boldsymbol{\alpha}$. It can be
1 The L^0 "pseudo-norm". $\|\boldsymbol{\alpha}\|_0 := \#\{i \mid \alpha_i \neq 0\}$ (NP-hard),
2 The L^1 norm (convex) $\|\boldsymbol{\alpha}\|_1 := \sum_{i=1}^{p}|\alpha_i|$,
3 The L^q norm $\|\boldsymbol{\alpha}\|_q := \left(\sum_{i=1}^{p}|\alpha_i|^q\right)^{1/q}$
This is a *selection* problem. When ψ is the L^1-norm, the problem is called Lasso or basis pursuit.

Incucing sparsity
Left: $\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}|\mathbf{x}-D\boldsymbol{\alpha}|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1$
Right: $\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}|\mathbf{x}-D\boldsymbol{\alpha}|_2^2$ s.t. $\|\boldsymbol{\alpha}\|_1 \leq T$.



Designed dictionaries Wavelets, Curvelets, Wedgelets, Bandlets,…lets

Learned dictionaries of patches

$$\min_{\boldsymbol{\alpha}_i, D\in C} \sum_i \frac{1}{2}\|\mathbf{x}_i - D\boldsymbol{\alpha}_i\|_2^2 + \lambda\psi(\boldsymbol{\alpha}_i)$$

reconstruction + sparsity

solve denoising problem
1 Extract all overlapping 8 × 8 patches $\mathbf{x}_i$
2 Solve a matrix factorization problem (Eq. above) with $n > 100000$. Average thereconstruction of each patch.

Inpainting, Demosaicking

$$\min_{D\in C,\boldsymbol{\alpha}} \sum_i \left\|\boldsymbol{\beta}_i \otimes (\mathbf{x}_i - D\boldsymbol{\alpha}_i)\right\|_2^2 + \lambda_i\psi(\boldsymbol{\alpha}_i)$$

RAW Image Processing GRBG-mosaic → white balance, black substraction → denoising → demosaicking → conversion to sRGB, gamma correction → image.

12 Texture

Key issue Representing texture
1 Texture analysis/segmentation → representing texture
2 Texture synthesis: Useful, also gives some insight into quality of representation
3 Shape from texture Computer graphics: texture mapping

General • Textures are made up of quite stylised subelements, repeated in meaningful ways.
• Representation: Find the subelements, and represent their statistics.
• Subelements: Normalized correlation, apply filters.
• Filters: Spots and oriented bars at a variety of different scales (by experience), details probably don't matter
• Statistics: Within reason, the more the merrier. At least, mean and standard deviation. Better, various conditional histograms.

Oriented pyramids Laplacian pyramid is orientation independent. Apply an oriented filter to determine orientations at each leayer. By clever filter design, we can simplify synthesis. This represents image information at a particular scale and orientation.

Final texture representation
1 Form an oriented pyramid (or equivalent set of responses to filters at different scales and orientations)
2 Square the output
3 Take statistics of responses
a e.g. mean of each filter output (are there lots of spots)
b std of each filter output
c mean of one scale conditioned on other scale having a particular range of values (e.g. are the spots in straight rows?)

Texture synthesis
1 Use image as a source of probability model
2 Choose pixel values by matching neighbourhood, then filling in
3 Matching process: Look at pixel differences, count only sythesized pixels

12.1 Histogram

Principle
1 Intensity probability distribution
2 Captures global brightness information in a compact, but incomplete way
3 Doesn't capture spatial relationships

Co-occurrence matrix Is a matrix of distribution that is defined over an image to be the distribution of co-occurring values at a given offset. Mathematically, a co-occurence matrix C is defined over an $n \times m$ image I, parametrized by an offset $(\Delta x, \Delta y)$ as:

$$C_{\Delta x,\Delta y}(i,j) = \sum_{p=1}^{n}\sum_{q=1}^{m}\begin{cases}1 & \text{if } \spadesuit,\\ 0 & \text{else.}\end{cases}$$

$\spadesuit$: $I(p,q) = i, I(p+\Delta x, q+\Delta y) = j$.
For an image with grey tones

$$\begin{pmatrix}0&0&1&1\\0&0&1&1\\0&2&2&2\\2&2&3&3\end{pmatrix}$$

Compute

	0	1	2	3
0	#(0,0)	#(0,1)	#(0,2)	#(0,3)
1	#(1,0)	#(1,1)	#(1,2)	#(1,3)
2	#(2,0)	#(2,1)	#(2,2)	#(2,3)
3	#(3,0)	#(3,1)	#(3,2)	#(3,3)

$$0^\circ: \quad P_{\text{horizontal}} = \begin{pmatrix}4&2&1&0\\2&4&0&0\\1&0&6&1\\0&0&1&2\end{pmatrix}$$

$$45^\circ: \quad P_{\text{right diag.}} = \begin{pmatrix}4&1&0&0\\1&2&2&0\\0&2&4&1\\0&0&1&0\end{pmatrix}$$

12.2 Texture synthesis

Goal Texture synthesis algorithms are intended to create an *output image* that meets the following requirements:
1 The output should have the size given by the user.
2 The output should be as similar as possible to the sample.
3 The output should not have visible artifacts such as seams, blocks and misfitting edges.
4 The output should not repeat, i. e. the same structures in the output image should not appear multiple places.
Like most algorithms, texture synthesis should be efficient in computation time and in memory use.

12.2.1 Methods

View-dependent texture synthesis One can create textures viewed from different angles.

Clique Type Selection
step 1 Collect the complete 2nd-order statistics for the example texture, i.e., the intensity difference distributions of all clique types up to a maximum length. After this step the example texture is no longer meeded.
step 2 Generate an image filled with independent noise with values uniformly distributed in the range of the example texture. This noise image serves as the initial synthesized texture, to be refined in subsequent steps.
step 3 Collect the pairwise statistics of all clique types (up to the same maximal length) for the current synthesized image (initially noise).
step 4 For each clique type, compare the difference distributions of the example texture and the synthesized texture by calculating the Euclidean distance.
step 5 Select the clique type with the maximal distance. If this distance is less than some threshold go to step 8 - the end of the algorithm. Otherwise, add the clique type to the (initially empty) neighborhood system and its difference distribution to the (initially empty) parameter set.
step 6 Synthesize a new texture using the updated neighborhood system and parameter set. The texture should have the prescribed statistics for all clique types in the neighborhood system.
step 7 Go to step 3.
step 8 End of the algorithm.
The distribution distances that are compared between clique types in step 4 are weighted with the number of cliques. This should prevent unstable statistical behavior when there are only few cliques (typical for lang clique types.)

Tiling The simplest way to generate a large image from a sample image is to tile it. This means multiple copies of the sample are simply copied and pasted side by side. The result is rarely satisfactory. Except in rare cases, there will be the seams in between the tiles and the image will be highly repetitive.

Stochastic texture synthesis Stochastic texture synthesis methods produce an image by randomly choosing colour values for each pixel, only influenced by basic parameters like minimum brightness, average colour or maximum contrast. These algorithms perform well with stochastic textures only, otherwise they produce completely unsatisfactory results as they ignore any kind of structure within the sample image.

Single purpose structured
texture synthesis Algorithms of that family use a fixed procedure to create an output image, i. e. they are limited to a single kind of structured texture. Thus, these algorithms can both only be applied to structured textures and only to textures with a very similar structure. For example, a single purpose algorithm could produce high quality texture images of stonewalls; yet, it is very unlikely that the algorithm will produce any viable output if given a sample image that shows pebbles.

Chaos mosaic This method, proposed by the Microsoft group for internet graphics, is a refined version of tiling and performs the following three steps:
1 The output image is filled completely by tiling. The result is a repetitive image with visible seams.
2 Randomly selected parts of random size of the sample are copied and pasted randomly onto the output image. The result is a rather non-repetitive image with visible seams.
3 The output image is filtered to smooth edges.
The result is an acceptable texture image, which is not too repetitive and does not contain too many artifacts. Still, this method is unsatisfactory because the smoothing in step 3 makes the output image look blurred.

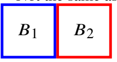
Pixel-based texture synthesis These methods, such as "Texture synthesis via a noncausal nonparametric multiscale Markov random field." Paget and Longstaff, IEEE Trans. on Image Processing, 1998, "Texture Synthesis by Non-parametric Sampling." Efros and Leung, ICCV, 1999, • Assuming Markov property, compute $P(p|N(p))$. Building explicit probability tables infeasible. Instead, let's search the input image for all similar neighborhoods - that's our histogram for p. to synthesize p, just pick one match at random. "Fast Texture Synthesis using Tree-structured Vector Quantization" Wei and Levoy SIGGRAPH 2000 and "Image Analogies" Hertzmann et al. SIGGRAPH 2001. are some of the simplest and most successful general texture synthesis algorithms. They typically synthesize a texture in scan-line order by finding and copying pixels with the most similar local neighborhood as the synthetic texture. These methods are very useful for image completion. They can be constrained, as in image analogies, to perform many interesting tasks. They are typically accelerated with some form of Approximate Nearest Neighbor method since the exhaustive search for the best pixel is somewhat slow. The synthesis can also be performed in multiresolution, such as "Texture synthesis via a noncausal nonparametric multiscale Markov random field." Paget and Longstaff, IEEE Trans. on Image Processing, 1998.

Patch-based texture synthesis Patch-based texture synthesis creates a new texture by copying and stitching together textures at various offsets, similar to the use of the clone tool to manually synthesize a texture. "Image Quilting." Efros and Freeman. SIGGRAPH 2001 and "Graphcut Textures: Image and Video Synthesis Using Graph Cuts." Kwatra et al. SIGGRAPH 2003 are the best known patch-based texture synthesis algorithms. These algorithms tend to be more effective and faster than pixel-based texture synthesis methods.

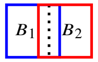
Chemistry based Realistic textures can be generated by simulations of complex chemical reactions within fluids, namely

Reaction-diffusion systems. It is believed that these systems show behaviors which are qualitatively equivalent to real processes (Morphogenesis) found in the nature, such as animal markings (shells, fish, wild cats...).

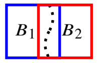
Efros & Leung '99 extended
Observation: neighbor pixels are highly correlated. Idea: Unit of synthesis is block • Exactly the same but now we want $P(B|N(B))$.
• Much faster: Synthesize all pixes in a block at once
• Not the same as multi-scale



Random placement of blocks



Neighobring blocks constrained by overlap



Minimal error boundary cut

Algorithm 1 Pick size of block and size of overlap
2 Synthesize blocks in raster order
3 Search input texture for bolck that satisfies overlap constraints (above and left)
4 Paste new block into resulting texture. Use dynamic programming to compute minimal error boundary cut

12.3 Texture Transfer

Take the texture from one object and "paint" it onto another object. This requires spearating texture and shape. That's hard, but we can cheat. Assume we can capture shape by baundary and rough shading. Then, just add another constraint when sampling: similarity to underlying image at that spot.

Shape from textures Possibility to recover normals both for deterministic and probabilistic textures.

13 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
- impulse response $t(-x, -y)$
- Canny nonmaxima suppression
- Entropy Coding (Huffman code)
- Aperture problem: normal flow
- Lucas-Kanade: Iterative refinement/local gradient method
- Coarse-to-fine-estimation
- SNR scalability EI, EP frame
- MPEG Structure
- Radon transform: Just FT with first Fourier coefficient?
- Sparse representations for image restoration: Inpainting, demosaickiking

A Big equations

$$\mathcal{F}[h](u, v) = \frac{1}{2\ell} \int_{-\ell}^{\ell} \mathrm{d}x_1 \, \exp(-i2\pi u x_1) \cdot \underbrace{\int_{-\infty}^{\infty} \mathrm{d}x_2 \, \delta(x_2) \exp(-i2\pi v x_2)}_{=1} = \operatorname{sinc}(2\pi u \ell) \tag{A.1}$$

$$E = \iint \mathrm{d}x\mathrm{d}y \left[\left(\frac{\partial I}{\partial x} \frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial I}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} \right)^2 + \alpha^2 (\|\nabla \dot{x}\| + \|\nabla \dot{y}\|)^2 \right] \tag{A.2}$$

$$\mathbf{v} = \begin{pmatrix} \sum_i w_i I_x(q_i)^2 & \sum_i w_i I_x(q_i) I_y(q_i) \\ \sum_i w_i I_x(q_i) I_y(q_i) & \sum_i w_i I_y(q_i)^2 \end{pmatrix}^{-1} \cdot \begin{pmatrix} -\sum_i w_i I_x(q_i) I_t(q_i) \\ -\sum_i w_i I_y(q_i) I_t(q_i) \end{pmatrix} \tag{A.3}$$