Image Analysis & Computer Vision

Summary Andreas Biri, D-ITET 19.01.18

1. Introduction

Applications: Surveillance, Virtual/Augmented Reality, Remote Sensing, Robotics, Inspections, Medicine
- easier if controlled conditions & little uncertainty
- increased productivity/flexibility/reliability/attention

Geometric optics: look at perfect, straight rays
Physical optics: include interaction with materials

Properties of light: EM wave with orthogonal E & M
- *wavelength* (400 - 800nm for visible light)
- *direction* (perpendicular to EM fields)
- *amplitude*
- *phase*
- *direction of polarisation* (can be exploited)

Visible spectrum: 380 – 760 nm
- 380 – 450: violet
- 450 – 490: blue
- 490 – 560 : green (555nm most receptive)
- 630 – 760: red
- higher: *near-infrared* (exploit at night)

Solar spectrum has peaks around 500nm, therefore human vision most receptive there (exploit e.g. for NVGs)

Near-infrared: region just above 760nm
- can be used for better reception at night (invisible to human eye, therefore does not blind us but seen by cam)
- useful for detecting vegetation, as strong reflection

Use special light sources to find e.g. different plants simply by exploiting their different spectral reflectance

1.1 Scattering ("Streuung")

Rayleigh:	small particles,	**strong** wavelength dep.
Mie:	comparable size,	**weak** wavelength dep.
Non-selective:	large particles,	**not** wavelength dep.

Long wavelengths observe less scattering (infrared) wherefore we can see through the clouds with radar

Blue scattered the most (therefore blue sky, *Tyndall* effect)
- if sand in the air, more scattering of other wavelengths

1.2 Reflection ("Spiegelung")

Angle of reflection = Angle of incidence

At high angle of incidence (very flat), full reflectance

Brewster angle: for dielectrics (insulators), no reflection of parallel polarized light under correct angle ($\sim 60^{\circ}$)

For conductors, more or less preserving, strong reflection everywhere (can be exploited in metal detection on chips)

Roughness: can influence and lead to *diffuse* reflection
- flat surface: **specular** reflection, everything mirrored
- Rough: **diffuse** reflection, into many directions

Lambertian reflection: observed brightness only depends on direction of incoming light, not on viewing angle
- intensity decreases with viewing angle ($\cos\theta$)
- larger surface area observed if more oblique ($1/\cos\theta$)
- therefore, surface looks equally bright from all directions

Bidirectional Reflection Distribution function (BRDF): 4D function specifying how much light is reflected in direction from light coming from another one

Does not depend on the wavelength

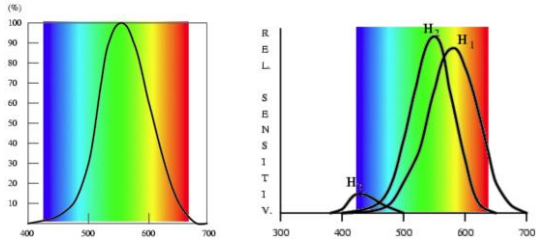
1.3 Refraction ("Brechung")

In contrast to reflection, refraction depends on wavelength
- angle of refraction different for varying wavelengths
- n_i is changing, material not same for all $\lambda's$

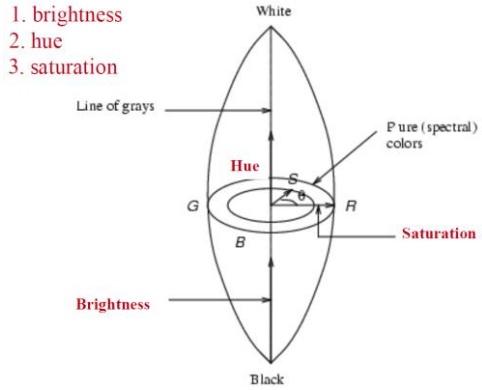
Dispersion: depends on material AND wavelength
- if none, then absorption based on resonance frequencies

1.4 Diffraction ("Beugung"): Also wavelength dependent

1.5 Human Vision System (HVS)



HVS much more sensible to green light (555nm) than rest
- need stronger signal to achieve same brightness
- topic of *Photometry* (subjective impression), whereas *Radiometry* focuses on objective, physical measurements

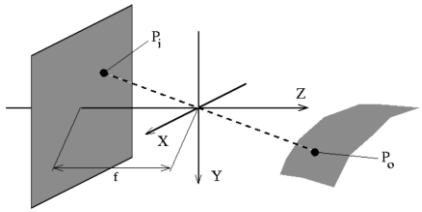


Larger saturation gives more intense / pure colours

Relative luminance: context plays a strong role (contrast)
Spatial frequencies: brain automatically filters high freqs.
Filling in: brain tries to auto-complete, find patterns

2. Acquisition of Images

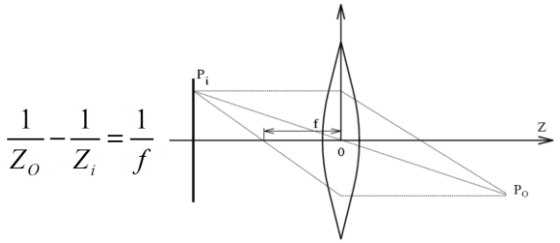
2.1 Cameras



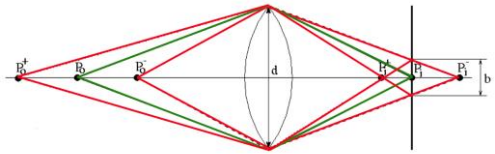
$$\frac{X_i}{X_o} = \frac{Y_i}{Y_o} = \frac{f}{-Z_o} = -m$$

m : linear magnification

Lens: captures light from one point and concentrates it
- sharpness (light from one point focused)
- brightness (large enough hole for sufficient light)
- only works for points *in focus* of the lens
- assume "thin" lens ($thickness \ll radii$), parallel to axis



Depth of field: only in focus in certain range, else blurred
- decreases with d, increases with Z_0
- balance between incoming light (d) & usable depth (ΔZ)
- f: focal lens ; b : acceptable blob size (~ pixel)



$$\Delta Z_0^- = Z_0 - Z_0^- = \frac{Z_0(Z_0 - f)}{Z_0 + f\,d/b - f}$$

Aberrations

Occur if assumptions violated:
- not all points focused into 1 image point
- all image points in a single plane
- magnification constant

Geometrical: forms are distorted
- *Spherical:* parallel rays do not converge
- *Radial:* different magnification for various angles
  (**Barrel:** magnification *de*creases on borders
  Pincushion: magnification *in*creases on borders)

Chromatic: depends on wavelength & materials
- rays of different wavelengths focused in different planes
- use multiple lenses so that all are focused at same point

Camera types

CCD: *Charge-coupled devices*

Use area more efficiently for light reception
- has to read everything one after another (slow)
- high production cost & power consumption
- blooming (charges can spill over to other cells)

CMOS: *Complementary Metal Oxide Semiconductor*

Lots of logic in-between, less light sensitive
- cheaper to produce, as standard CMOS technology
- more noise because of per-pixel amplification
- can increase sensitivity with micro lenses

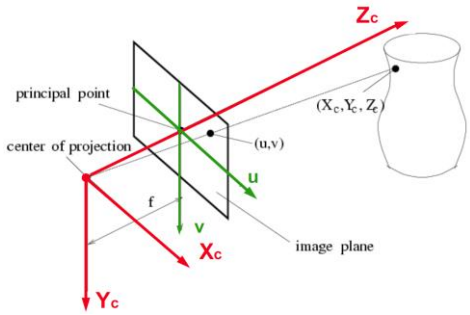
Colour separation

Prism: separate light into 3 beams using prism & 3 sensors
- full use of resolution, but expensive & sensitive

Filter mosaic: Bayer filter, reduce effective resolution

Filter wheel: rotate multiple filter in front of lens
- only suitable for static scenes (take multiple shots)

Perspective projection

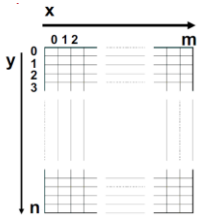


Origin at *center of projection*
- Z axis: optical axis
- X & Y: parallel to image rows / columns

$$u = f\,\frac{X}{Z}, \qquad v = f\,\frac{Y}{Z}$$

For constant Z (far enough away), we can approximate:

$$x = k\,X, \qquad y = k\,Y, \qquad k = \frac{f}{Z}\ (scaling)$$



$$\begin{cases} x = k_x u + s\,v + x_0 \\ y = \quad k_y v + y_0 \end{cases}$$

with :

Internally calibrated: know k_x, k_y, s, x_0, y_0
- project coordinates in image plane to pixel coordinates

Externally calibrated: know C, R
- project world coordinates to coordinates in image plane

Calibration Matrix K: includes all internal parameters

k_i : number of pixels / unit length (k_x horizontally, k_y vertically)

k_y / k_x : *aspect ration*

Homogenous coordinates: additional dimensions
- only defined up to a factor (does not influence coords)
- can be used to create *linear system* (to use matrices)

Photometric camera model: light received at area decreases with angle by $cos^4(\alpha)$

Irradiance: energy received on camera surface
Radiance: energy emitted by light source

We can get the grey levels from the irradiance as

$$f = g\, I^{\gamma} + d\,, \qquad g : Gain \text{ ("Blende")}$$
$$d : dark\ reference$$

2.2 Illumination

Simplify image processing by controlling environment

Back-lighting: place lamps behind object
- high-contrast silhouette images for binary vision

Directional lighting: create sharp shadows / specular reflection to get information about shape (e.g. crack)

Diffuse lighting: illuminate everything uniformly

Polarized lighting: use different materials & filters
- improve contrast btw Lambertian & specular reflections (latter keeps polarization, diffuse reflection polarizes)
- improve contrast btw dielectrics & metals (*Brewster*)

Coloured lighting: highlight region of similar colour using a band-pass filter (monochromatic)

Structured lighting: objects distort projected pattern

Stroboscopic lighting: compensate motion blur
- only illuminate shortly to have short integration time

3. Sampling & Quantisation

3.1 Discretization of continuous signals

Sampling: discretize space to pixels
Quantization: discretize amplitude/signal to levels

Mostly, rectangular shape used, even though hexagon would be optimal (more isotropic, no connectivity issues)

For binary images (structural importance), quantization is only secondary, sampling is however central
- can do non-uniform coverage (fine sampling for details)

Sampling: integrate brightness over cell & read at center
- represent signal through decomposition in orthogonal basis (Dirac eigenfct for each position)

3.2 Spatial & frequency domain

Point spread function: spatial distribution of points

Separate **convolution mask** into smaller ones to reduce complexity of the operator and reduce computations

Sinusoids are *eigenfunctions* of LSI systems and can therefore describe all signals by linear combination
- LSI: *linear shift-invariant* (LTI for discrete case)

Fourier transform: project function onto base

$$F(u,v) = \iint_{-\infty}^{\infty} f(x,y)\, e^{-i2\pi(ux+vy)}\, dxdy$$

Inverse Fourier transform:

$$f(x,y) = \iint_{-\infty}^{\infty} F(u,v)\, e^{i2\pi(ux+vy)}\, dudv$$

Repetitive patterns show peaks in spatial frequencies (as periodic structure); mostly, DC components very strong
- can be used to only show repetitions / discard structure

Phase still central: tells “how to combine the amplitudes”

$$f(x,y) \leftrightarrow F_R(u,v) + iF_I(u,v)$$

spatial domain	frequency domain
real	real part even imaginary part odd
real,even	real, even
real, odd	imaginary, odd

Rotations: Fourier and spatial domain both rotate the same way (same direction)

Scaling: zoom *out* in spatial (increase frequency) ↔ zoom *in* in frequencies (inverse proportionality)

Modulation transfer fct: Fourier transform of *point spread*
- can be easier to analyse in freq. domain, as convolution becomes multiplication of functions

Sampling in spatial domain (convolution with a window) is multiplication with a **2D sinc** in frequency domain (lowpass), as we integrate the intensity over the pixel area and thereby **suppress high frequencies when integrating**

3.3 Discretization in space & frequency

We need to discretize both in spatial & frequency domain

Spatial domain: multiplication with 2D pulse train (pixels) → convolution with Dirac train results in periodic repetition of the original signal which might overlap

Aliasing: overlap of periodic repetitions of the same signal
- if sample fast enough, periodic repetitions are separated far enough in frequency space (if bandlimited!)

Sampling frequency $\geq 2 *$ maximal signal frequency

Use Gaussian as a good approximation for sinc (as sinc requires infinite support) for interpolation (recreating spatial function from frequency components) as we want to multiply the frequency spectrum with a box filter to only get a single repetition of the Fourier spectrum of the signal
- higher order kernel: decreased aliasing / better lowpass

Limiting the spatial extend: As pictures are limited in space, the recovered Fourier transform is affected

Leakage: mixing up frequencies over the whole spectrum
- can be compensated by subsequent sampling in freq.

Frequency domain: when sampling in frequency spectrum, the spatial signal is repeated periodically

To summarize: in order to get perfect representation:

1. Signal needs to be band-limited
2. Sampled at or above Nyquist rate

Those two steps prevent *Aliasing*

3. Sampling compatible with signal period

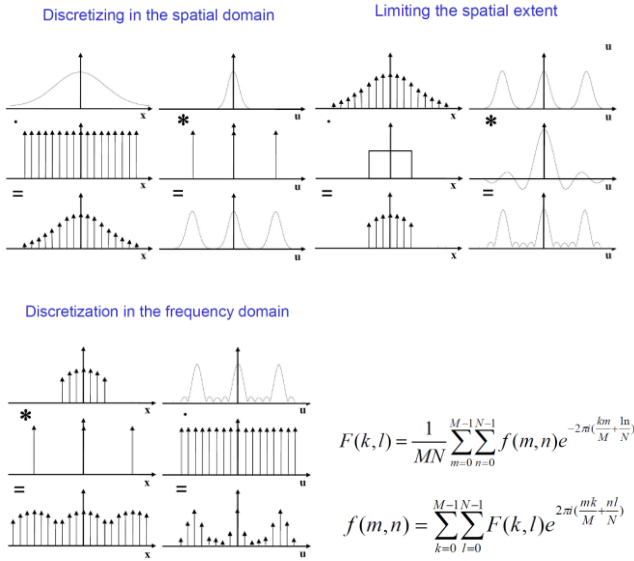
This limits the *Leakage* problem from the limited space

4. Signal is periodic

This allows us to have *discrete Fourier spectrum* as well

Discrete Fourier Transform *assumes* periodicity in spatial & frequency domain

(might introduce false high frequencies, as "periodic")



$$F(k,l)=\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)e^{-2\pi i\left(\frac{km}{M}+\frac{ln}{N}\right)}$$

$$f(m,n)=\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}F(k,l)e^{2\pi i\left(\frac{mk}{M}+\frac{nl}{N}\right)}$$

4. Image Enhancement

Normal image: high DC component, as large homogenous areas with power situated primarily around center
Noise: white noise has frequency components everywhere

→ High SNR in center of image, less at high frequencies
- noise primarily at high frequencies (*Salt-and-pepper*)
- edge information however also lost if dropped

4.1 Noise suppression

Convolution linear filters

Use low-pass filter to mask high-freq. components

1. 2D sinc (mask in frequ. Domain)
- rippling effects due to ripples of spatial filter & convolution in spatial domain

2. Use convolution filters without ripples (approximate)
- *Averaging filters*: no ripples in spatial, but blurry image & ripples in frequency domain (high order > smooth > better low-pass)
- *Box filters:* in general, all have ripples in frequency (periodic, as box filters discrete in spatial domain)
- *Binomial filters* (Gaussian): no ripples & always positive, separable (efficient), gives good low-pass

All linear filters: remove low-pass gives blurred results

Non-linear filters

Edge-preserving to fight blurring

Median filter: rank-order neighbours, take median value (*odd-man-out*, no new grey values emerge)

→ Get rid of outliers (*Salt-and-pepper* noise), but no smoothing as with averaging filters and preserves discontinuities (but some details better with Gaussian)

Anisotropic diffusion: adapt size of Gaussian "at runtime"
- **width/diffusion as a function of the image gradient**
- restrain so that not totally washed out (restraining force)

1. Gaussian smoothing across homogenous area (large kernel)
2. No smoothing across edges (small kernel)

4.2 Image de-blurring

(if a lot of noise: first smooth with a Gaussian)

Unsharp masking

Interpret blurred image as diffusion process & invert it

Approximate Laplacian with *Difference-of-Gaussians* (DoG)
- sharper impression due to "overshooting" of flanks

Inverse filtering

Estimate blurring filter and invert it
- for $B(u,v)=0$, info is irreparably lost
- for high frequencies, the inverted filter can amplify noise

Wiener filter

Optimal filter: no noise amplification, discard if low SNR
- need to know H very precisely & SNR, which I dont

$$Wf(H)=H'(u,v)=\frac{H(u,v)}{H^*(u,v)H(u,v)+1/\mathrm{SNR}}$$

4.3 Contrast enhancement

- compensate under- , overexposure
- spending intensity range on interesting part of image

Histogram equalisation: want to have spread intensities
- use intensities more evenly but still keep relative order

Use *Cumulative intensity probability* (Cumulative distribution function, CDF) as a mapping function
- intervals with many intensities are expanded
- not entirely flat due to discrete nature of histogram

5. Feature Detection

5.1 Edge Detection

Locating high Intensity Gradient magnitudes

Locate edges at slopes of point spread function

Can calculate gradient as convolution (as linear, shift-inv.)
- prone to noise (need something to smooth first)

Sobel mask: discrete approximation
- separable: *Derivative* + *Binomial* filter
- smooth in one direction, derive in other
- gaps & several pixels wide lines, strongly varying

$\frac{\partial}{\partial x}$

-1	0	1
-2	0	2
-1	0	1

$\frac{\partial}{\partial y}$

-1	-2	-1
0	0	0
1	2	1

Locating inflection points

Search zero-crossings of Laplacians, as edges lie at the intensity inflection points; again, use masks
- sensitive no noise, therefore combine with smoothing
- find **one-pixel thick edges & closed contours**

Difference of Gaussians (DoG): "Mexican hat" filter
- smooth, then find zero-crossing; approximate derivative

Canny edge detector

Use *matched filter* for optimal detection (low SNR, correct)
- want to maximize difference between zero-crossings
- looks like derivative of Gaussians (good approximation)
- as we also want to have smoothing along the curve

→ directional derivatives of Gaussian, pick largest one
- **Non-maximum suppression:** only take gradient which is not smaller than any of the two neighbours
- **Hysteresis threshold:** implement lower & upper limit

→ One-line thickness, can have gaps, very efficient

5.2 Corner Detection

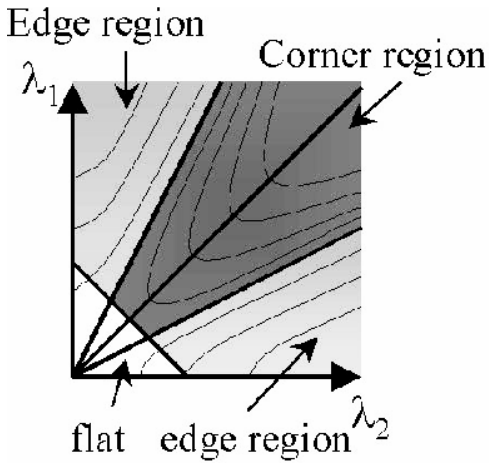
Harris Corner Detector

Distinguish: homogenous areas, edges & corners
- look at derivatives in 2 directions and decide on them

Find the directions of minimal & maximal change
- use the structure tensor / second order moments
- look for eigenvectors & eigenvalues

Two small eigenvalues: Plane (no variation)
One large, one small: Edge
Two large eigenvalues: Corner (multiple directions)

$$Determinant - k * (Trace)^2 = f_{xx}f_{yy} - f_x^2 f_y^2 - k * (f_{xx} + f_{yy})^2$$



6. Image decomposition

Scale space: scenes contain information at different levels
- develop hierarchical descriptions
- increase efficiency by working on lower resolutions

1. Smooth with Gaussian filter (filter high freq.)
2. Take difference image (→ DoG / Laplacian)
3. Reduce size of smoothed image (downsample)

Zero-crossings of Laplacian yield edge information

For discrete filter c_{-1}, c_0, c_{+1}: $c_0 \geq 4\, c_{-1} c_{+1}$

Unitary image transformations

Image decomposition into family of orthogonal basis images as a linear decomposition
- concentrate energy in a few components (compress)
- separable basis images for all dimensions
- image independent basis is suboptimal (e.g. Dirac/Fourier)

$$\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} B_i(x,y)\; B_j^*(x,y) = \delta_{ij}$$

Find weights by projecting image onto base using the orthogonality of the basis vectors (larger error with others)

For well-known bases (*sin, cos, Hadamard*), only need to send weights, otherwise also require sending basis

Discrete Cosine Transform: best decorrelation, efficient
- do not have "unnatural" frequencies due to periods by eliminating boundary discontinuities

Hadamard: very efficient, as only "+1"s and "-1"s

Principal Component Analysis (PCA)

Use image-dependent basis (KLT)

Reduce dimensionality of data while retaining data
- transform to new, uncorrelated variables
- **collect maximal variance in uncorrelated components**

Rotate into direction of max. correlation & redistribute variance / energy mostly there
- find direction with most variation, then second most etc.

Covariance matrix: find vector s.t. variance maximized
- eigenvectors are exactly the vectors we search for
- higher eigenvalue equals to stronger variation / difference in direction corresponding to eigenvector
- all eigenvectors should be uncorrelated ("unique"), which can be shown to require orthogonality

Inspection: find outliers as variations in small eigenvalues, as we can observe the largest abnormalities there (usually none in that direction, so it must be abnormal)

→ Helps with *Compression, Inspection, Classification*
- neighbouring pixels are usually strongly correlated
- minimal least-square error for truncated approximations

Problems: very effective, but computationally expensive
- need to send *eigenimages* to other party for compression
- need to compute very large covariance matrices ($N^2 \; x \; N^2$)
- need to specify image statistics

Independent Component Analysis

Try to find the most different parts & separate them into different images with minimal mutual information

While PCA minimizes correlation/redundancy (second order moment), ICA minimizes the dependence in-between parts (also higher moments)

7. Segmentation

Grouping pixels into segments to understand scenes

7.1 Thresholding

Separate object and background based on intensity
- take minimum between two histogram peaks
- ignores neighbouring pixel, but much smaller size

Otsu: minimize within-group variance ($p_1\sigma_1^2 + p_2\sigma_2^2$)
- can also be used locally for individual patches

Mathematical Morphology

reduce noise influence by correcting thresholding
- only operates on binary images, non-linear

Dilation: "if one neighbour is white, become white"

Erosion: "if not entirely white neighbours, become black"

Remove noise in background (*open*): 1. Erode, 2. Dilate

Remove noise in object (*close*): 1. Dilate, 2. Erode

Can also do the same with rank-ordered neighbourhood:
- $i_t = i_1$: Erosion
- $i_t = i_N$: Dilation
- $i_t = i_{N/2}$: Median filtering (edge-preserved smoothing)

Connected component labelling

Points are connected if can be reached through the same component; component should be homogenous

Region growing: to through all pixels which are connected and determine whether they belong to the same region
- determine whether two labels exist for the same component so you can combine both to one new

For two points: $P(i,j)\ and\ Q(k,l)$

$$D_4(P,Q) = |i-k| + |j-l|$$

$$D_8(P,Q) = max(|i-k|, |j-l|)$$

7.2 Edge based – Hough transformation

For predefined shapes & parametric shape models
- use parameters to display in lower dimensions

Straight line: defined by position & orientation
- better: angle & distance to origin (no infinity problem)
- find points which contribute to find finite lines

Hough transform: use known parameters to find form

1. Inspect all points of interest (from edge detection)
2. For each point, draw the parameter space
3. Peaks in the counters correspond to forms

Robust to noise, but requires good peak detection
- use weighting contributions (e.g. gradient magnitude) to improve robustness of counters to detect correct lines

7.3 Region based

Detect homogenous regions & use them as seeds

Grow regions as long as homogeneity criterion satisfied
- merge close areas if still inside bounds
- use "watersheds" to separate areas cleverly

7.4 Statistical Pattern Recognition

Unsupervised learning

Distribute measurements to classes
- homogeneity within classes & small variance

K-means: chose K classes and use K centers / means
- minimizes in-class variance

1. For each measurement, find "nearest" class
2. Recalculate class mean and continue

Maximize the multiplication of the probabilities that each measurement can be found in its selected class

Supervised Generative models

Use examples to learn distributions from measurements
- need to learn correct distributions, features are central

Segment after *maximum a-posteriori (MAP)*, i.e. maximize segmentation given the new measurements using previous knowledge (know distribution of classes & examples)
- "Which class would result in the highest likelihood?"

$$p(c_i|m) = \frac{p(m, c_i)}{p(m)} = \frac{p(m|c_i)\, p(c_i)}{p(m)}$$

Can also do inverse: choose class so that probability of measurements are the highest given the class, i.e.

$$\arg_i p(m|c_i) \quad \text{(Data Likelihood)}$$

Don't need class priors for this, as need to estimate

Markov random fields: probability of class of pixel $P(c_j)$ only depends on direct neighbours, not on rest

Joint distribution: $p(c, m) = \prod_{j=1}^{M} p(m_j|c_j)\, P(c_j)$

Solve this by defining a *Gibbs* energy and minimizing it
- computationally expensive to do in practice
- alternative: approximate by guessing label & only work on neighbourhoods when sampling individual labels

Supervised Discriminative models

Only want function to map from features to classes, "learn from examples and apply" ground-truth

Learning: estimate parameters of the model
- *optimization problem*: minimize difference to ground truth

K-nearest neighbours (KNN): find K nearest neighbours in training set and use their labels
- high dependency on K and the used training set
- require lots of training samples for accuracy

Others: *Random forests, Neutral networks*

8. Surface: colour & texture

Luminance: "How much power at $\boldsymbol{\lambda}$?"
Chrominance: "$Which\ \lambda$?"

8.1 Colour

CIE primaries: $\lambda_1 = 700\ nm, \lambda_2 = 546\ nm, \lambda_3 = 435\ nm$

$$R_i(i) = \int H_i(\lambda)\, C(\lambda)\, d\lambda = \sum_{j=1}^{3} m_j \int H_i(\lambda)\, P_j(\lambda)\, d\lambda$$

H_i : sensitivity of a cone to a specific wavelength
C_i : received signal strength per wavelength

Project light source onto 3 numbers → lose information
- multiple (different) sources can give the same impression
- use 3 primaries to create the same impression as original

Tristimulus values: $T_j = m_j/w_j$ (white is reference)

Spectral matching curves: tell you strength of all primaries to get vision of intended monochromatic wavelength
- negative values cannot be produced (are simply dropped)

Chromatic coordinates: normalise so sum of all = 1
- does not include brightness information anymore
- can be displayed with two coordinates (no brightness)
- Y represents luminance
- can produce all colours inside the triangles / shape
- can be transformed to faithfully represented perceptual distance between colours everywhere

Colour constancy

Environment strongly influences the system & colour
- perception depends on surroundings

Patches *keep* colour even if they reflect differently (spectral light source can change but colour remains)

Patches *change* colour if surrounding patches change

8.2 Texture

Characteristics
- *oriented* vs *isotropic* (no dominant orientation)
- *regular* vs *stochastic* (no rules to build from smaller parts)
- *coarse* vs *fine*

Fourier features

Peaks show periodicity (dominant frequencies)
- low freq. → coarse, high freq. → high

Coarseness: high or low (look at ring in Fourier space)

Orientation: directionality in power spectrum

However, only collects features over entire picture

Cooccurrence matrix

Histograms do not capture spatial relationships

Cooccurence: probability distribution for intensity pairs
- contain info about intensity at head and tail of vector
- periodic: everything on main diagonal ($i = j$)
- can also get energy / entropy / contrast / max. probability

Filter banks

Laws filters: fixed convolution filters, very effective

Gabor filters: Gaussian envelope multiplied with cosine
- good localization in both domains (actually optimal)
- can cover entire Fourier domain to get direction/scale

Eigenfilters: filters adapted to the optimal texture
- sparse to increase efficiency of large patterns
- use small Eigenfilters for inspection (see error)

Stochastic models

1. **Neighbourhood system:** compare pixel pairs (Cliques)

2. **Statistical parameter set:** histogram for each clique type
- synthesize texture based on model corresponding to type

9. Deformable contour models

Similar to region growing & watershed
- here clearly defined shapes, continuity of objects
- can use fitting techniques to add missing edges
- use voting methods like Hough to get parameters

Snakes

Given an initial contour near the desired object, it should evolve to fit the object boundaries exactly
- enforce continuity of curves

Elastic band adjusts so that:
- near image positions with high gradients (edges)
- satisfy shape "preferences" and priors
- can be used for continuous deformations (*tracking*)

Energy minimization: minimize energy function so that
- wraps around given object (needs to be outside)
- stretches & fits to object (needs to be inside)

$$E_{tot} = E_{internal} + E_{external}$$

Internal: encourage prior shape: smoothness, elasticity, known shape prior (only based on snake itself)
- deformation energy (stretching & bending increases it)
- helps for missing data (e.g. holes)
- can be extended to allow for some (known) size to prevent everything from shrinking to nothing

$$E_{internal} = \sum_{i=0}^{n-1} \underbrace{\alpha \left\| v_{i+1} - v_i \right\|^2}_{\text{Elasticity, Tension}} + \underbrace{\beta \left\| v_{i+1} - 2v_i + v_{i-1} \right\|^2}_{\text{Stiffness, Curvature}}$$

External: encourage contour to fit to image structure
- attract curve towards interesting features (e.g. curves)
- larger directional derivative gives less energy (negative)

$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

Shape prior: if known, can penalize if we deviate from a known form using an additional energy term

Energy minimization

Can also add other energy terms at will:
- Pressure (Balloons) to push outside / normal
- Gravity for constant force in preferred direction
- Interactive force by user for input
- Distance-based energy (to the nearest edge) gives gradient everywhere and prevents short-sightedness

Partial Differential Equations (PDE)

Get Euler-Lagrange differential equation & approximate it in the discrete case to get a linear equation system
- to invert matrix, use boundary conditions (closed contour)

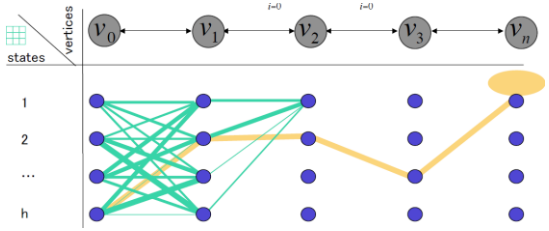
Extension: take temporal development into account
- *kinetic energy:* includes the "mass" of the snake
- *friction:* use damping coefficient to prevent jumps

Greedy search: local solution

For each point, search window around it and move where energy function is minimal
- does not guarantee convergence nor optimality
- requires decent initialization & correct parameters

Dynamic programming: Viterbi algorithm

A single point only affects its neighbours (local effect)
- again, search locally around initialized snake for optima
- iterate until each point is in the center of its box, i.e. the snake is optimal in the local (constrained) search space



For each position, find optimal (least costly) position at next point → minimize over entire chain (globally)
- for circular snakes, simply fix a single node

10. Motion Extraction

Motion might be only cue for segmentation

10.1 Optical Flow

For all points, try to find where they will move to
→ dense motion reconstruction (2D motion vector)

Try to find projection of 3D vectors onto image
- apparent motion of brightness patterns
- wrong if patterns not existent / illumination changes

Assumption: pixels keep their intensities across frames
- usually, temporal change is rather slow

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

Measureable: $I_x = \frac{dI}{dx}, I_y = \frac{dI}{dy}, I_t = \frac{dI}{dt}$

Unknown: $u = \frac{dx}{dt}, v = \frac{dy}{dt}$

1 equation for two unknowns is underdetermined
- could be solved using higher derivatives of intensity
- planar intensity profiles can never be resolved (common)

Aperture problem: only component along gradient can be retrieved, the other one is unknown

Horn & Schunck algorithm

Add additional smoothness constraint (small changes)
- reduce error of smoothness and equation above
- also reduces the influence of noise (damped)
- *Regularization:* add extra constraints for bad problems

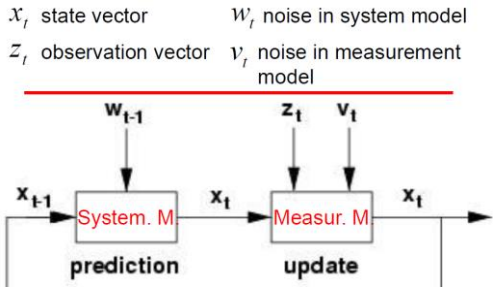
Calculus of variations: assume you know solution, we can then reformulate the optimization to one over scalar

Euler-Lagrange equations for each function (u, v)
- use iterative methods for PDE & multiple frames

$$\Delta u = \lambda(I_x u + I_y v + I_t) I_x$$
$$\Delta v = \lambda(I_x u + I_y v + I_t) I_y$$

10.2 Condensation filter

Track object which we want to follow
- shift focus from single pixels to entire object



1. **Prediction** based on *system* model

$$p(x_t|z_{t-1}) = \int p(x_t|x_{t-1})\, p(x_{t-1}|\, z_{t-1})\, dx_{t-1}$$

2. **Update** based on *measurement* model

$$p(x_t|z_t) = {p(z_t|x_t)\, p(x_t|z_{t-1})}\Big/{p(z_t|z_{t-1})}$$

Recursive Bayesian filter: track multiple possible states
- object not a single state but probability distribution
- "predict and then look if actually there"

Use *Kalman filters*: assume linear model, Gaussians
- only way to analytically solve recursive Bayesian filters
- exact model, but only very restricted in applications

CONDENSATION: Conditional **Den**sity Propag**ation**

- discrete samples for performance (can quickly reconstruct Gaussian from a few samples)
- can directly test multiple hypotheses at once
- requires post-processing of distribution (e.g. avg)

1. Predict new sample set based on system model
2. Sample from predicted set with prob. based on measurement model (improbables fade out)
   - weights for sampling based on measurements
   - samples can be drawn multiple times, but noise in system model will create different ones

Other approaches: model-based (shape) & Feature tracking

11. Feature Extraction

Feature: property characteristic to pattern / object

Features deal with large variations in:
- Viewpoint
- Illumination
- Background
- Occlusions
- Shape (deformations)

Global features: can only detect entire object
Local: can extract part of object (with occlusions)

11.1 Interest points

Want to find same points with high precision
- uniqueness of the patch is important to track

Sum of Square Differences (SSD): calculate difference in intensities between mask can original

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

Study eigenvalues & eigenvectors of H:
- largest eigenvalue corresponds to direction of big change
- smallest increase can be seen at small eigenvector

Shift difference (error) should be large in all directions
- therefore, minimum of E should be large for all vectors
- achieved by the smallest eigenvalue of H

→ **take point where even small eigenvalue is still large**

Harris Corner Detector: similar, there just look at both eigenvalues instead of simply the smaller one

11.2 Invariance under geometric change

Want to discriminate between different interest points and search a **descriptor** for **matching** it over multiple frames
- local (planar) patch should be invariant under transformation (geometric / photometric change)

Perpendicular direction

Only see limited deformations: **Similarity**
- rotation
- translation
- amplification (scale changes)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathrm{s} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Any direction, but far enough distance

Still assume that object (more or less) in one plane
- Z distance to object is constant

2D affine transformation: **Affinity**

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Any direction, any distance

All distances possible: **Projectivity**

Invariances

More difficult to get invariances for all projectivities than similarities: $Similarities \subset Affinities \subset Projectivities$

Parallel lines: invariant for sim. & affine, but not project.
Orthogonality: invariant to sim, but not affine & project.
Parallelograms: use intensity extrema as edges (same area)

Can also use invariances to show *collinearity, tangency, inflections, parallelism* (not P)

Mostly, can assume only similarities (also comp. simple)

Photometric: use *gradient* (directions), as often invariant

11.3 Examples

Integrals: Evaluate relative affine invariant parameter along two edges by integrating the area along a closed curve using the derivative / curvature

Parallelograms: define it using invariant extrema of a function to select edge points, then create a parallelogram whose area is constant across images

Maximally Stable Extremal Regions (MSER)

Start with intensity extremum, then move intensity threshold and watch region grow
- take regions at threshold where growth is slowest, as they indicate strong edges
- can even order region according to growth speed

Scale Invariant Feature Transformation (SIFT)

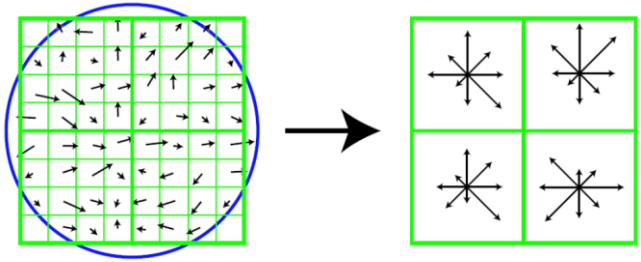
Dominant orientation selection using Gaussians

Point of interest: blob-detection over various scales
- if point is maximum of 27 points (over 3 scales)

1. Compute image gradients
2. Build orientation histogram
3. Find maximum match

As maximum oriented the same way, metric is rotation-invariant and therefore does not require corrections

Match interest points based on descriptors:
- threshold image gradients over a grid on multiple scales
- create histogram within block & normalize with Gaussian



12. 3D acquisition

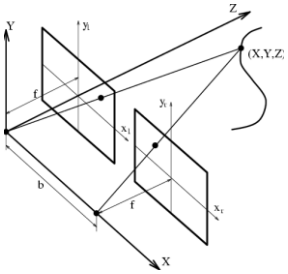
Passive: uni-directional (using textures & shading
multi-directional (using stereo cameras)

Active: uni-directional (using time-of-flight)
multi-directional (using line scanning, light)

12.1 Passive

Stereo

Uses triangulation of two cameras to get 3D position



- Identical cameras
- separated by *base line* b

- coplanar image planes
- world coords in origin

$$\rho\, p = K\, R^t\, (P - C)$$

Increasing b & f increases the depth resolution (more sampling), but is less receptive to near objects

Disparity: horizontal shift between image x coords

Tilted cameras: allow cameras to be nearer, but still sensitive enough in depth in this area

Epipolar line: projection of viewing ray from 1. Camera onto image plane of 2. Camera: point must be on it
- *epipole:* projection of 1. Camera's center onto 2. Camera
- *fundamental matrix F*: used to calculate line for other

Using epipolar lines reduces the search for matching pixels, as it has to lie on this plane (search only best there)
- much quicker and better matching (only 1D now)
- can actually calculate F if I have at least 8 corresponding points (e.g. using RANSAC) as homogenous system

Point matching: find pair of points on epipolar lines
- correlations (deformations, match intensity)
- feature-based (edges, corners)
- due to noise, might have to approximate rays

Structures

Use homogenous texture to detect how surface is tilted so that projection onto image plane looks like it does
- can do same things with contours (e.g. ellipses/circles)

Silhouettes

Use multiple pictures of rotating object to get silhouette and combine it to a complete shape

12.2 Active

Line scanning: use active triangulation of a laser
- shine laser on object and get reflection on camera
- if know orientation of laser & cam, can get point
- send "plane of light" for guaranteed intersection

Structured light: project patterns of special shape onto scene and observe deformation of patterns
- *Serial binary pattern*: get exact position in 2^n lines
- *Coloured patterns:* yields 3^n lines
- distinguishing lines is equal to multiple simultaneous planes projected onto the object → 3D point cloud

Time-of-Flight: send modulated light signal and measure time it requires to travel before returning to the sensor
- send pulsed: for longer distances
- measure phase difference: determine small differences
- still very good over range (compared to stereo / light)
- requires returned signal (difficult on dark surfaces)

Photometric stereo: project different light sources
- light shining from different angles reflects differently depending on the source
- use intensity to determine possible surface orientations

13. Tracking

Feature-based tracking: generic (corners, blobs, regions)
Model-based tracking: application specific (face, torso)

Tracking-by-detecting: detect independently in each frame
Temporal filtering: use model to predict next position

Particle filters: predict where its going to be & use measurements to improve estimation in next frame

13.1 Feature-based

Region tracking

Use constant (known) background to calculate difference
- doesn't work if background changes
- cannot detect if object has same colour as background
- use snakes around figure to get good shapes

Mean-shift tracking: track region with colour distribution, always going where the center of mass is

Point tracking

Look for changes in intensity (*optical-flow equation*)
- need to capture fast enough, else local minima
- need to have a non-zero gradient (**aperture problem**)

Use *spatial coherence constraint*: pixel's neighbours have the same movement (solves aperture problem)
- solve least squares problem (similar to Harris Corner)

Temporal tracking

Requires initial template to know how object looks
- SIFT: use known descriptors and find them again

Lucas-Kanade Template Tracker: use iterations to optimize warp parameters until we have the best deformation

13.2 Model-based

Tracking-by-Detection (in 3D)

Use reference images of object to detect it

1. Detect Keypoints (strongest feature points)
2. Build feature descriptors
3. Match Keypoint descriptors to database

Use neural networks for class detection (learn descriptors)

Space Time analysis: collect detections in space and see in future and past which path would give best correlation

Body Articulation

Pictorial structures: model consists of parts & structure
- *parts:* 2D image fragments, single components
- *structure:* tells how components are connected
- minimize energy function to fit everything ("Does it make sense to have this configuration?") → combine parts

Use training data to learn normal walking behaviour and match it temporally (have periodicity & variations)

Articulated tracking: use low-dimensional representation

On-line learning

If we have good tracking, we can update the template during measurements & adapt to gradual changes
- update criteria to adapt to environment / illumination

Self-learning: can adapt to wrong things, as teaches model itself and can therefore strongly deviate from original
- can use original model as "*anchor*" to limit drift

Context

Can use supporters to improve tracking of objects
- occlusions can be found if look at neighbouring objects
- use scene to infer on past / future

14. Specific Object Recognition

13.1 Model-based

Try to compare image features with features of objects stored in database, figuring out type & pose
- very slow *hypothesise-and-verify* approach
- much more difficult in 3D, require projecting onto 2D
- very compact, can deal with clutter (know model)
- need to create sample set for each object

Invariant-based recognition: instead of trying to figure out pose, find features which are invariant to it
- use integral of curvature & compare "histogram"

13.2 Image-based (appearance)

Template matching: shift model over image & compare
- simple change in viewpoint requires lots of templates

Principal Component Analysis: represent data in lower-dimensional space keeping most of the variance
- take lots of pictures from different angles and store dominant PCs to compare them to new image
- can estimate pose based on nearest PC representation
- large models which cannot easily handle clutter
- very efficient & easy to produce for lots of models

13.3 Hybrid techniques

Euclidean invariant features

1. **Detection:** get points of interest (e.g. corners)
2. **Description:** Take circular region around it & *locally* calculate region invariants from different viewpoints (limited search, as largely invariant)
3. **Matching:** Compare image with database & chose best one (also see that structure of corners same), easily handle cluttering due to *local* character
   - difficult if uniform texture, as pose not clear

Invariance: viewpoint, illumination, background, occlusion
- start using local invariant features to decrease templates

Local: no dependence on occlusion & background
Invariance: robust to change in viewpoint & illumination

Repeatability: Want to represent them under affine & linear photometric changes
Distinctiveness: should uniquely identify the object

To be able to compare them locally, the areas should look the same (without having to compare images first):

1. Use Canny edge detector to get shapes
2. On contours, find extrema of average value
3. Use them to select parallelograms based on the invariant function → same areas

Using those *affine invariant neighbourhoods*, we can then extract descriptors from them which we will match

Improvements – Visual words

Visual words: quantize / cluster descriptors & match them hierarchically in a *vocabulary tree* for speed-up

1. Create descriptor for many points of interest
2. Group points of interest clusters to get the word
3. Use multiple such layers as a hierarchy (faster)

Inverted file index: get images which correspond to words
- works very fast for very large databases & vocabularies
- weight all found references & choose one with the most

"Find all images in which a visual word occurred"

K-means clustering: randomly initialize K cluster centers, then assign each feature & iteratively improve centers

RANSAC

Test configuration of the descriptor matches to verify
- remove all outliers by testing that fundamental matrix exists which can generate the given point distribution
- test epipolar geometry (rigidity) & projectivities (planar)

15. Object Category Recognition

Additional complexity of intra-class variations

Classification: binary answer "Is it class X?"
Detection: needs localization "Where is X?"

15.1 Classification

Local features cover the same portion of the object in any view (only based on local characteristics, no comparison)

Visual word: a part of an image with similar image content
- local characteristic: can be a single component of a body

Specific, textured objects: sparsely sample interest points
- use a large vocabulary to get best results

Object categorization: dense sampling over entire image
- better recognition with smaller vocabulary (general characteristics, not too specialized)

Bag-of-words: use visual words in image for "histogram"
- good descriptors to tell apart / compare and get features
- summarize entire image with content distribution
- can describe image with single vector, good for learning
- *orderless representation*: discards spatial relations

Classifier: based on a bag-of-words, categorize image into given classes based on decision rules
- performs better than part-based models, as not as relying on sample image, much better adaptability to unseen
- Nearest neighbour, neural networks, support vector

Lack of spatial relations (order) can be good and bad
- no reliance on fixed model, more adaptable to changes
- does not realise when things "don't make sense"
- use **visual phrases** of frequently co-occurring words
- keep spatial relation as feature or use grid-based

Spatial Pyramid Representation: multiple grid layers
- works well in classifying scenes (but confused when e.g. camera turned, as wrong grid classifications)

15.2 Detection

Use sliding window to pass entire image & search for features which match previous training samples

Greyscale: sensitive to illumination variations
Vector of pixel intensities: sensitive to even small shifts

Gradient-based: summarize local distribution of gradients in histogram (split image into grids to compare parts)
- *locally orderless*: invariance to small shifts & rotations
- *contrast-normalization:* allows variable illuminations

Histogram of Oriented Gradients (HOG)

Represent each image patch with histogram of gradient
- local normalization allows for comparable results
- decide in subwindows based on nearest neighbour, CNN

Integral image: value at (x, y) is sum of all pixels above and to the left; compute any sum in constant time
- allow extremely fast computation of rectangular filters, as can only take edge points of rectangle & sum / subtract

AdaBoost: strong classifier by combining many weak ones
- subsequent learning process: iteratively improve (*boost*)
- try to minimize sum of weights of all classifications
- misclassified ones have high weight (expensive)
- very fast after being trained & chosen good filters
- only works well on good 2D shapes which remain rigid

Use Integral image representation & apply multiple filters:

1. Evaluate each filter on each example
2. Sort examples by filter values
3. Find best threshold for each filter (min error)
4. Re-weight based on errors, then re-do

Cascade filters & try to filter out bad image as fast as possible to increase efficiency, keeping complex filters last
- keep false positives rate low, need to drop out fast

15.3 Part-based models

Deformable Part Model (DPM)

Represent object by its parts
- parts are detected locally
- encode spatial structure of the parts as components
- positions can change (deformable) compared to root

Sliding window approach using HOG & latent SVM
- SVM: *Support Vector Machine*
- parts are learned automatically by SVM

"Find best constellation of parts maximizing criteria"

Reduce if deviate from "ideal" locations of parts

Star model

Check on position of parts relative to a central part

1. Find root of object
2. Find rest around it

Implicit Shape Model (ISM)

Learn an appearance codebook & star-topology structure
- uses probabilistic Generalized Hough Transform
- good localization properties, flexible geometric model
- requires supervised training data to learn code-book
- no discriminative learning (can have too many parts)

Predict where the center of the object is based on the different parts of the object & a trained code-book:

1. Find the visual words contained in the image, each containing *displacement vectors* to center
2. Find the center of the object based on votes

Convolutional Neural Network (CNN)

Each layer influences next layer of cells
- layers are hidden, training influences connections