1 Introduction

2 Image Preprocessing

2.1 Convolution

For two continuous functions f and g, the convolution is defined as
$\boxed{(f * g)(x) = \int_{-\infty}^{\infty} g(\tau) f(x - \tau) d\tau}$ being commutative, associative and linear.

2.2 Fourier Transform

The Fourier transform of f is given by:
$\boxed{\hat{f}(k) := \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx}$ With the inverese Fourier transform:
$\boxed{f(x) := \int_{-\infty}^{\infty} \hat{f}(k) e^{2\pi i k x} dk}$

2.3 Impulse Functions

Dirac impulse: $\boxed{\delta(x) = \begin{cases} 0 & if x \neq 0 \\ \infty & if x = 0 \end{cases}}$ with $\boxed{\int_{-\infty}^{\infty} \delta(x) dx = 1}$

And a series of Dirac impulses: $\boxed{\sum_{n=-\infty}^{\infty} \delta(x - nT)}$

2.4 Nyquist Sampling Theorem

If f is band bounded with cutoff frequency k_0 $\hat{f}(k) = 0$ for all k with $|k| \geq k_0$
then it is completely determined by giving its ordinates at a series of points spaced at most $\frac{1}{2k_0}$
meaning the sample frequency must be larger than $2k_0$

2.5 Quantisation

Characteristic with equidistant steps of size Δ

$$\boxed{g(x) = max\{0, min\{g_{max}, \left\lfloor \frac{I(x)}{\Delta} + \frac{1}{2} \right\rfloor\}\}}$$

$$\boxed{h(x) = \Delta g(x) - \frac{\Delta}{2}}$$

yielding an error of non-overdriven quantiser:

$$\boxed{I(x) - h(x) \in [-\frac{\Delta}{2}, \frac{\Delta}{2}]}$$

2.6 Gamma Correction

$\boxed{g_{out} = g_{max} \left(\frac{g_{in}}{g_{max}} \right)^{\gamma}}$ which keeps black and white, is a nonlinear transformation

2.7 Models of Blur

Blur can be modeled with convolution

$\boxed{g_{blurred} = g_{sharp} * p}$ with p modeling the blur

$\boxed{p_{motion}(x) = \begin{cases} \frac{1}{n} & if - n < x \leq 0 \\ 0 & otherwise \end{cases}}$ along x-axis by n pixels

$$\boxed{p_{Gauss}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}}}$$

2.8 Wiener Deconvoution

Restore a sharp image from a blurred image with a Wiener filter: $\boxed{g_{blurred} = g_{sharp} * p + v}$ with p being a point-spread function (blur), v being pixel noise and assuming g_{sharp} and v being independent.

With $\boxed{g_{restored} = f * g_{blurred}}$, find optimal f that minimizes

$\boxed{e(k) = E\left[|\hat{g}_{sharp}(k) - \hat{g}_{restored}(k)|^2 \right]}$ with E being the expectation value.

2.9 Models of Noise

Statistical noise: $\boxed{g_{noisy}(x, y) = g_{sharp}(x, y) + v(x, y)}$

Malfunctioning sensors: $\boxed{g_{noisy}(x, y) = \begin{cases} g_{sharp}(x, y) & with\ probability\ p \\ arbitrary & otherwise \end{cases}}$

3 Edge-Detection

3.1 Finding Edges

Approximating a derivative by difference: $\boxed{\frac{\partial g}{\partial x} \approx \frac{g(x+1) - g(x-1)}{2}}$

3.2 Laplace Operator

The 2D analogon to the second order derivatie is the Laplace operator:
$\boxed{\bigtriangledown^2 g = \frac{\partial^2 g}{(\partial x)^2} + \frac{\partial^2 g}{(\partial y)^2}}$ which can be appriximated with
$\boxed{\bigtriangledown^2 g \approx g(x+1,y) + g(x-1,y) + g(x,y+1) + g(x,y-1) - 4g(x,y)}$
Because the second order derivative is very noisy, so combine Laplacian with Gaussian smoothing:
$\boxed{\bigtriangledown^2 (G * g) = (\bigtriangledown^2 G) * g}$ which yields
$\boxed{\bigtriangledown^2 G = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} G(x,y)}$ called Laplacian of Gaussian (LoG) or Mexican Hat
The Laplacian of Gaussian can be approximated by calculating the Difference of Gaussian (DoG)
$\boxed{DoG(x,y) = G_{\sigma_1}(x,y) - G_{\sigma_2}(x,y)}$

3.3 Corner Detection

Using dissimilarity measure to find patches of maximal dissimilarity for local moves:

$$\boxed{d := \sum_{(u,v) \in rectangle} (g(u+\Delta u, v+\Delta v) - g(u,v))^2 \approx \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix}^T \begin{pmatrix} \sum(\frac{\partial g}{u})^2 & \sum \frac{\partial g}{u}\frac{\partial g}{v} \\ \sum \frac{\partial g}{u}\frac{\partial g}{v} & \sum(\frac{\partial g}{v})^2 \end{pmatrix} \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix}}$$

For special choice of coordinate system it becomes a diagonal matrix:

$$\boxed{d := \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix}^T \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} = \lambda_1 (\Delta u)^2 + \lambda_2 (\Delta v)^2 \;\; w.l.o.g. \;\; \lambda_1 \geq \lambda_2 \geq 0}$$

4 Curve Fitting

4.1 2D Geometry of Lines

Line in normal form: $\boxed{0 = \langle \vec{n}, \vec{x} \rangle + c}$ so the distance of a point $\vec{r}$ from the line:
$\boxed{d = \|\vec{l} - \vec{r}\| = |\langle \vec{n}, \vec{r} \rangle + c|}$

4.2 Hough Transform

Every line in 2D can be represented as: $\boxed{x\ cos\phi + y\ sin\phi + c = 0}$ so there is a 2D space of all line represented by (ϕ, c) called Hough Space.

4.3 Total Least Squares

To estimate a line through a set of points we need to search the line parameters that minimise d_i
$\boxed{minimise \sum_{i=1}^{N} d_i^2\ subject\ to\ \langle \vec{n}, \vec{n} \rangle = 1}$ Using the Lagrange function:
$\boxed{L(\vec{n}, c, \lambda) = \sum_{i=1}^{n} d_i^2 - \lambda(\langle \vec{n}, \vec{n} \rangle - 1)}$ then zeroing the partian derivative with respect to c and then to n, we can arrive at
$\boxed{\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix} \vec{n} = \lambda \vec{n}}$ hence λ is Eigenvalue and $\vec{n}$ is Eigenvector. We get two solutions of Eigenvalue Problem: $\boxed{\lambda_1 \geq \lambda_2 \geq 0}$ where λ_2 minimises distances and λ_1 maximises distances

4.4 Weighted Least Squares

Because of outliers introduce weights $w_i \geq 0$, one for each edge point, then solve:
$\boxed{minimise \sum_{i=1}^{N} w_i d_i^2\ subject\ to\ \langle \vec{n}, \vec{n} \rangle = 1}$. But how to choose w_i? Replace d_i by a term that grows more slowly!

4.5 M-Estimators

M-Estimator means $\boxed{minimise \sum_{i=1}^{N} \rho(d_i) \; subject \; to \; \langle \vec{n}, \vec{n} \rangle = 1}$ with a suitable function ρ

The derivatives of the Lagrange function of M-Estimators approach and Weighted Least Squares approach are equal if $\boxed{w_i = \frac{\frac{\partial \rho(d_i)}{\partial d_i}}{2d_i}}$. So running Weighted Least Squares with appropriate weights implements M-Estimators. E.g.:

$\boxed{\rho_{Cauchy} : d \mapsto c^2 log(1 + \frac{d^2}{c^2})}$

$\boxed{\rho_{Huber} : d \mapsto \begin{cases} d^2 & if|d| \leq k \\ 2k|d| - k^2 & otherwise \end{cases}}$

4.6 Least Trimmed Sum of Squares Estimator (LTS)

$\boxed{minimise \sum_{i=1}^{p} d_{i:N} \; subject \; to \; \langle \vec{n}, \vec{n} \rangle = 1}$ with $p < N$ and $d_{i:N}$ the i-th element in the ordered list of point-distances. p is typically given as a percentage of N.

4.7 RANSAC

Search a line that passes nearby as many points as possible.

$\boxed{minimise \sum_{i=1}^{N} \sigma(d_i) \; with \; \sigma(d_i) = \begin{cases} 0 & if|d_i| \leq \theta \\ 1 & if|d_i| > \theta \end{cases}}$

4.8 Estimating Circles

Parametric representation of circles:

$\boxed{(x - m_1)^2 + (y - m_2)^2 - r^2 = 0}$

Euclidean distance of point (x,y) from the circle:

$\boxed{d_E = \left| \sqrt{(x - m_1)^2 + (y - m_2)^2} - r \right|}$

Algebraic distance:

$\boxed{d_A = \left| (x - m_1)^2 + (y - m_2)^2 - r^2 \right|}$ Because minimising the Euclidean distance cannot be solved analytically, minimalize the sum of algebraic distances. Use a weighted approach, chose w_i so that we implement Euclidien by using weighted Algebraic.

5 Color

6 Segmentation

6.1 Level Set Evolution

Two class segmentation can be represented by indicatior function:
$\boxed{\phi(\vec{x}) \begin{cases} < 0 & if\ pixel\ \vec{x}\ belongs\ to\ segment \\ > 0 & if\ pixel\ \vec{x}\ belongs\ to\ background \end{cases}}$ with $\boxed{|\phi(\vec{x})| = distance\ of\ \vec{x}\ from\ contour}$
Modeling temporal evolution of signed distance function with $\boxed{\phi(\vec{x}, t)}$. When tracking a point $\boxed{\vec{x}(t)}$ on the boundary over time, it obviously follows $\boxed{\phi(\vec{x}(t), t) = 0\ for\ all\ t}$
$\boxed{0 = \frac{d\phi(\vec{x}(t), t)}{dt} = \nabla\phi\frac{\partial\vec{x}}{\partial t} + \frac{\partial\phi}{\partial t}}$. Solving this for $\frac{\partial\phi}{\partial t}$ yields:
$\boxed{\frac{\partial\phi}{\partial t} = -\nabla\phi\frac{\partial\vec{x}}{\partial t}}$

6.2 Mumford-Shah

Shrink contour if $\boxed{(I - C_{in})^2 > (I - C_{out})^2}$
Expand contour if $\boxed{(I - C_{in})^2 < (I - C_{out})^2}$
Mumford-Shah based segmentation:
$\boxed{\frac{\partial\vec{x}}{\partial t} = \frac{\nabla\phi}{\|\nabla\phi\|}(C + P - \lambda_1(I - C_{in})^2 + \lambda_2(I - C_{out})^2)}$

7 Camera Optics

7.1 World To Image Mapping

Point (x, y, z) is projected onto (x', y') via intercept theorem:

$$\boxed{z\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} f & 0 \\ 0 & f \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}}$$

Mapping from camera to image frame:

$$\boxed{\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \alpha & -\beta cot\theta \\ 0 & \frac{\beta}{sin\theta} \end{pmatrix}\begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}}$$

Mapping from world to camera frame:

$$\boxed{\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R\begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix} + \vec{t}}$$

Rewriting all those, given (ξ, η, ζ) we can calculate (u, v) by

$$\boxed{\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = A\left(R \mid \vec{t}\right)\begin{pmatrix} \xi \\ \eta \\ \zeta \\ 1 \end{pmatrix}} \text{ and } \boxed{\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{\tilde{z}}\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}}$$

The other way round, we can obtain (ξ, η, ζ) given (u, v) by

$$\boxed{\begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix} = zR^T A^{-1}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} - R^T\vec{t} \quad with\ z \leq 0}$$

7.2 Snell´s law

$\boxed{n_e sin\theta_e = n_t sin\theta_t}$ with $\boxed{n_{medium} = \frac{v_{vacuum}}{v_{medium}}}$

7.3 Thin lenses

Via Intercept theorem, we get the lens equation:

$$\boxed{\frac{1}{f} = \frac{1}{b} + \frac{1}{g}}$$

7.4 Depth of Field

Via intercept theorem with:

$\boxed{\triangledown g = g_{far} - g_{near} = 2\frac{g d_h (g-f)}{d_h^2 - (g-f)^2}}$ with $\boxed{d_h = \frac{Df}{\epsilon}}$ as hyperfocal distance.

7.5 Radial Distortion

Mathematical modeling with even polynominals:

$\boxed{\begin{pmatrix} x_d \\ y_d \end{pmatrix} = (1 + k_1 r^2 + k_2 r^4) \begin{pmatrix} x' \\ y' \end{pmatrix}}$ with $r^2 = (x')^2 + (y')^2$

7.6 Camera Calibration

Tsai´s approach:
From world-to-image-mapping take $M := A \left(R \mid \vec{t}\right)$ in which M is a 3x4 matrix.
Determine camera parameters by minimizing sum of squares, which means zeroing partial derivaties.

7.7 Telecentric Lenses

Maginification of telecentric lens: $\boxed{B = \frac{b-f}{f} G}$

Depth of field: $\boxed{\triangledown g = 2\frac{\epsilon}{D}(g-f)}$

8 Illumination

Irradiance of a surface point, meaning amount of incident light:
$\boxed{I(\theta_i, \phi_i)}$
Radiance of a surface point, meaning amount of emitted light:
$\boxed{I(\theta_e, \phi_e)}$
Total irradiance of a surface patch (in sperical coordinates):

$$\boxed{I_0 = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} I(\theta_i, \phi_i) sin\theta_i cos\theta_i d\theta_i d\phi_i}$$

Reflectance of a surface patch can be modeled by the bidirectional reflectance distribution function (BRDF):
$\boxed{f(\theta_e, \phi_e \mid \theta_i, \phi_i)}$ describing how much light is radiated in direction (θ_e, ϕ_e) if one unit of light is irradiated from direction (θ_i, ϕ_i)

8.1 Reflectance

So total amount of radiated light:

$$\boxed{I_0 = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} f(\theta_e, \phi_e \mid \theta_i, \phi_i) I(\theta_i, \phi_i) sin\theta_i cos\theta_i d\theta_i d\phi_i}$$

Lambertian Reflectance:
$\boxed{f(\theta_e, \phi_e \mid \theta_i, \phi_i) = \frac{1}{\pi}}$ hence: $\boxed{L(\theta_e, \phi_e) = \frac{1}{\pi} L_0}$
Specular Reflectance, mirroring effect:
$\boxed{f(\theta_e, \phi_e \mid \theta_i, \phi_i) = \frac{\delta(\theta_e - \theta_i)\delta(\phi_e - \phi_i - \pi)}{sin\theta_i cos\theta_i}}$ hence: $\boxed{L(\theta_e, \phi_e) = I(\theta_e, \phi_e - \pi)}$ Combinations of Reflectance:

$$\boxed{f(\theta_e, \phi_e \mid \theta_i, \phi_i) = \frac{\eta}{\pi} + (1 - \eta)\frac{\delta(\theta_e - \theta_i)\delta(\phi_e - \phi_i - \pi)}{sin\theta_i cos\theta_i}}$$

9 3D Reconstruction

9.1 Stereovision by Triangulation

For camera 1:
$\boxed{\begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix} = zR^T A^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} - R^T \vec{t}}$ with unknown depth z
Same for camera 2 and equaling:
$\boxed{z\vec{p} - \vec{q} = z'\vec{p'} - \vec{q'}}$ is overdetermined, lines of sight might be skewed. Minimize square of norm instead!

9.2 Phase Shift

Measure gray value of pixel 4 times with phase shift multiples of $\frac{\pi}{2}$:
$\boxed{g_i = A\ sin\left(\frac{u}{w}2\pi + \cdots\right) + B}$ with wavelength w. Solve for A and B.
$\boxed{u = \frac{atan2(g_1 - g_3, g_2 - g_4)}{2\pi} w + kw}$ only if $A \neq 0$

10 Pattern Recognition

10.1 Bayesian Classification

assign new object to the class with the largest posteriori probability:
$\boxed{\hat{c} = arg\,max\,P(c \mid x_{new})}$ and $\boxed{P(c \mid x) \propto P(x \mid c)P(c)}$

10.2 Linear Classification

A linear classifier is a function that implements a function of the kind:
$\boxed{\vec{x} \mapsto \begin{cases} +1 & if \langle \vec{x}, \vec{w} \rangle + b \geq 0 \\ -1 & otherwise \end{cases}}$ with $\vec{w}$ the weight vector and b the bias weight.
Margin, meaning the minimal distance between a hyperplane and the convex hull of the training patterns:

$$\boxed{\rho = min\left(d^{(i)} \frac{\langle \vec{x}^{(i)}, \vec{w} \rangle + b}{\|\vec{w}\|} \right)}$$

10.3 Support Vector Machine

SVM is a linear classifier that maximizes the margin ρ. Training a SVM means solving:

$$\boxed{maximize \quad \rho^2 \quad subject\ to \quad d^{(i)} \frac{\langle \vec{x}^{(i)}, \vec{w} \rangle + b}{\|\vec{w}\|} \geq \rho \qquad for\ all\ i}$$

Where $\|\vec{w}\|$ is a degree of freedom, so set $\|\vec{w}\| = \frac{1}{\rho}$ for convenience. That leaves us to:

$$\boxed{minimize \quad \frac{1}{2}\|\vec{w}\|^2 \quad subject\ to \quad d^{(i)}(\langle \vec{x}^{(i)}, \vec{w} \rangle + b) \geq 1 \qquad for\ all\ i}$$

10.4 Fault-tolerant SVM

Soft margin SVM:

$$\boxed{minimize \quad \frac{1}{2}\|\vec{w}\|^2 + C\sum_i \xi_i \quad subject\ to \quad d^{(i)}(\langle \vec{x}^{(i)}, \vec{w} \rangle + b) \geq 1 - \xi_i \qquad for\ all\ i}$$

10.5 Nonlinear SVM

Assume nonlinear transformation:

$$\boxed{\theta : \begin{cases} R^n \to R^m \\ \vec{x} \mapsto \theta(\vec{x}) = \vec{X} \end{cases}}$$

Useful kerner-functions are:

$\boxed{K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle}$ dot product

$\boxed{K(\vec{x}, \vec{y}) = (\langle \vec{x}, \vec{y} \rangle)^d \quad or \quad (\langle \vec{x}, \vec{y} \rangle + 1)^d}$ polynominal kernels

$\boxed{K(\vec{x}, \vec{y}) = e^{-\frac{\|\vec{x}-\vec{y}\|^2}{2\sigma^2}}}$

10.6 Validation Process

Expected risk of misclassification: risk of false negative + risk of false positiv

$$\boxed{E = \int_{A-} P(+)p_+(x)dx + \int_{A+} P(-)p_-(x)dx}$$

Which is unknown but can be approximated by

$\boxed{E \approx \frac{n_{fn} + n_{fp}}{n}}$, where n is the number of elements in the sample set, n_{fp} is the number of false positives in the sample set, n_{fn} is the number of false negatives in the sample set.

10.7 Haar Features

Calculating Haar features the naive way:

$$\boxed{s = \sum_{u=u_0}^{u_0+w-1} \sum_{v=v_0}^{v_0+h-1} g(u, v)}$$

The integral image: $\boxed{I(x, y) := \sum_{u=0}^{x} \sum_{v_0}^{y} g(u, v)}$ makes calculating s simply a 4-way lookup on I:

$$\boxed{s = I(u_0 + w - 1, v_0 + h - 1) - I(u_0 - 1, v_0 + h - 1) + I(u_0 - 1, v_0 - 1) - I(u_0 + w - 1, v_0 - 1)}$$

10.8 Ensembles

Sum up all predictions and compare with zero:

$$\boxed{ensemble(\vec{x}) = sign\left(\sum_{j=1}^{k} c_i(\vec{x})\right)}$$

10.9 Boosting

Introduce weight γ_i for each training pattern, applied to e.g. soft margin SVM this yields:

$$\boxed{minimize \quad \frac{1}{2}\|\vec{w}\|^2 + C\sum_i \gamma_i \xi_i \quad subject\ to \quad d^{(i)}(\langle \vec{x}^{(i)}, \vec{w}\rangle + b) \geq 1 - \xi_i \qquad for\ all\ i}$$

10.10 AdaBoost

Is an implementation of Boosting. With the training error bounded to

$$\boxed{\Pi_{t=1}^{T}\left(2\sqrt{\epsilon_t(1-\epsilon_t)}\right) \leq exp\left\{-2\sum_{t=1}^{T}(\frac{1}{2}-\epsilon_t)^2\right\}}$$

If all $\epsilon_t < \frac{1}{2}$ and $T \to \infty$ AdaBoost yields a perfect classifier.