Vision Algorithms For Mobile Robotics

GianAndrea Müller

October 26, 2018

Contents

1 Definitions

Definition 1. ***Computer vision*** *is defined by automatic extraction of meaningful information from images and videos of either semantic or geometric nature.*

Definition 2. ***Structure from Motion (SFM)*** *is more general than VO and tackles the problem of 3D reconstruction and 6DOF pose estimation from unordered images sets.*

Definition 3. ***Visual SLAM*** *is simultaneous localization and mapping. It focuses on a globally consistent estimation by performing loop detection and graph optimization in connection with visual odometry. Worse performance, better accuracy then VO.*

Definition 4. ***Visual Odometry (VO)*** *is the process of incrementally estimating the pose of the vehicle by examining the changes that motion induces on the images of its onboard cameras in real time.*

Definition 5. ***Bundle adjustment*** *is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates. Optimal means that the parameter estimates are found by minimizing some cost function that quantifies the model fitting error, and jointly that the solution is simultaneously optimal with respect to both structure and camera variations. The name refers to the 'bundles' of light rays leaving each 3D feature and converging on each camera centre, which are 'adjusted' optimally with respect to both feature and camera positions.*

Definition 6. ***Photogrammetry*** *is the science of making measurements from photographs, especially for recovering the exact positions of surface points.*

Definition 7. ***Pose-graph****: a network of nodes and edges where the nodes are robot poses and edges are constraints between poses.*

Definition 8. ***Loop closure****: a constraint between the a recent robot pose and a past pose when the robot revisits a previously visited location. Loop closure is highly sensitive to the current estimate of where the robot is. If your current estimate is bad you may not realize you are visiting a previously visited location! There are global loop closure approaches which try to match what the robot sees to everything seen in the past in order to find a closure, such approaches may be computationally expensive.*

2 Basics VO

Advantages

- More accurate than wheel odometry.
- Not affected by wheel slippage.
- Can be used complementary to
  - wheel encoders
  - GPS
  - inertial measurement units
  - laser odometry

Assumptions

- sufficient illumination
- dominance of static scene
- enough texture
- sufficient scene overlap

2.1 Working Principle

$$\boxed{T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}} \text{ Homogeneous transformation matrix}$$

Working Principle

1. Compute the relative motion T_k from images I_{k-1} to I_k.
2. Concatenate them to recover the full trajectory:
$$C_n = C_{n-1}T_n$$
3. An optimization over the last m poses can be done to refine the trajectory locally (Pose-graph or Bundle adjustment).

2.2 How to find T_k

In general:

$$\boxed{T_k = \arg\min_{T} \iint_{\bar{B}} \rho \left[I_k\left(\pi(\mathbf{T} \cdot \pi^{-1}(\mathbf{u}, d_{\mathbf{u}}))\right) - I_{k-1}(\mathbf{u})\right] d\mathbf{u}}$$

Direct image alignment

$$\boxed{T_{k,k-1} = \arg\min_{T} \sum_i ||I_k(\mathbf{u}'_i) - I_{k-1}(u_i)||^2_\sigma}$$

minimizes the per-pixel intensity difference.
This can be done at different resolutions: dense, semi-dense or sparse.

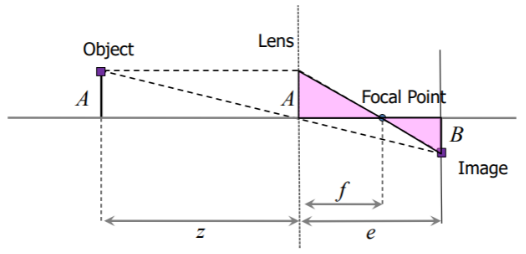
3 Image Formation

3.1 Pinhole Camera

Tradeoff between narrowness of the hole and diffraction issues.

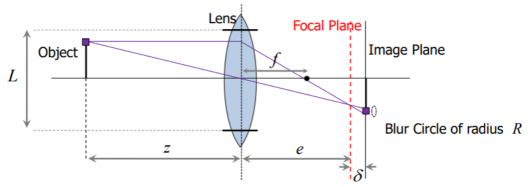
3.2 Converging Lens

- Rays passing through the optical center are not deviated.
- All light rays that are parallel to the optical axis converge in the focal point.



$\boxed{\frac{1}{f} = \frac{1}{z} + \frac{1}{e}}$ Thins lens equation

f	focal length	[m]
z	distance between image and lens	[m]
e	distance between object and lens	[m]



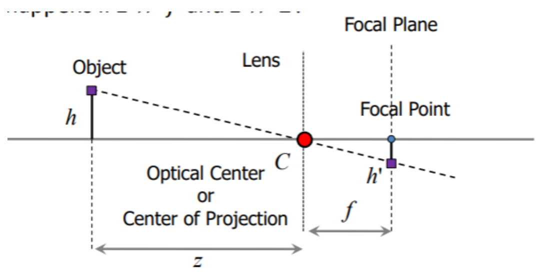
$\boxed{R = \frac{L\delta}{2e}}$ Blur Circle Formula

If the blur circle radius is smaller than a single pixel on the sensor, distance can no longer be calculated!

R	blur radius	[m]
L	aperture	[m]
δ	distance from the film	[m]

3.3 Pin-hole approximation

If $z \gg f$ and $z >> L$:



$\boxed{f \approx e}$

Do not confuse the center of projection and the focal point!

$\boxed{\frac{h'}{h} = \frac{f}{z} \Rightarrow h' = \frac{f}{z}h}$ Relation between image and object

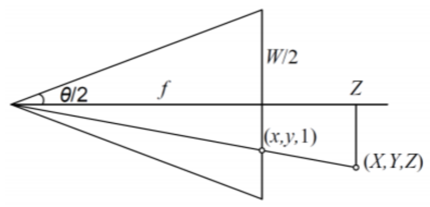
$\boxed{\begin{array}{l} x' = f\frac{x_c}{z_c} \\ y' = f\frac{y_c}{z_c} \end{array}}$ Perspective Camera Equations

3.4 Perspective Projection

Straight lines remain straight, angles are not preserved!

All parallel lines in reality intersect in the vanishing point in the projected picture, except for lines that are parallel to the camera plane, for those the vanishing point is infinitely far. All vanishing points lie on the vanishing lines, parallel planes intersect on the vanishing line.

Definition 9. ***Depth of Field (DOF)*** *is the distance between the nearest and farthest objects in a scene that appear acceptably sharp in the image.*



$\boxed{\tan(\frac{\theta}{2}) = \frac{W}{2f} \text{ or } f = \frac{W}{2}\left[\tan(\frac{\theta}{2})\right]^{-1}}$ Relation between field of view and focal length

$O = (u_0, v_0)$	Optical center	$[\text{m}] \in \mathbb{R}^2$
k_u, k_v	scale factors for pixel dimensions	$\left[\frac{pixel}{\text{m}}\right]$
$\alpha_u = k_u f, \alpha_v = k_v f$	focal length in pixels	$[pixel]$

$$\boxed{\begin{array}{l} u = u_0 + k_u x \Rightarrow u = u_0 + \frac{k_u f X_c}{Z_c} \\ v = v_0 + k_v y \Rightarrow y = v_0 + \frac{k_v f Y_c}{Z_C} \end{array}}$$

Introduction of scale:

$$p = \begin{pmatrix} u \\ v \end{pmatrix} \Rightarrow \tilde{p} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

So the equations above can be expressed in matrix form:

$$\boxed{\begin{bmatrix}\lambda u\\ \lambda v\\ \lambda\end{bmatrix} = \underbrace{\begin{bmatrix}k_u f & 0 & u_0\\ 0 & k_v f & v_0\\ 0 & 0 & 1\end{bmatrix}}_{K} \begin{bmatrix}X_c\\ Y_c\\ Z_c\end{bmatrix} = \begin{bmatrix}\alpha_u & 0 & u_0\\ 0 & \alpha_v & v_0\\ 0 & 0 & 1\end{bmatrix} \begin{bmatrix}X_c\\ Y_c\\ Z_c\end{bmatrix}}$$

The K matrix consists of the location of the intersection of the optical axis with the image plane (u_0, v_0).

3.4.1 Summary: Projection of a Point from world coordinates to Pixels

1. Given a point in world coordinates P_w calculate its location in camera coordinates P_c using a homogeneous transformation $T = [Rt]$:

$$\begin{bmatrix}X_c\\ Y_c\\ Z_c\\ 1\end{bmatrix} = \begin{bmatrix}R & t\end{bmatrix} \begin{bmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{bmatrix}$$

2. Project the point to the 1 meter image plane to get the **calibrated** or **normalized coordinates** p:

$$\begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}\frac{X_c}{Z_c}\\ \frac{Y_c}{Z_c}\end{bmatrix}$$

3. Optionally apply lens distortion to get to **distorted normalized coordinates** p_d:

$$\begin{bmatrix}x'\\ y'\end{bmatrix} = (1 + k_1 r^2 + k_2 r^4) \begin{bmatrix}x\\ y\end{bmatrix}$$

4. Convert the distorted normalized coordinates to ge the discretized pixel coordinates $(u, v)^T$:

$$\lambda \cdot \begin{bmatrix}u\\ v\\ 1\end{bmatrix} = K \begin{bmatrix}x'\\ y'\\ 1\end{bmatrix}$$

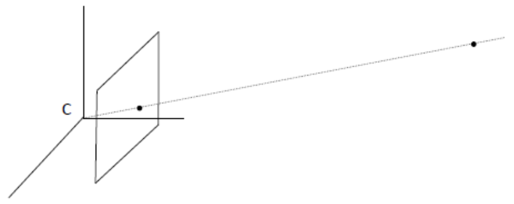
3.4.2 Summary: Undistoring and image

1. Define all pixel coordinates of the destination image (undistorted image).
2. Distort those coordinate location.
3. Measure the image intensity of the source image at the calculated locations.
4. Map the measured intensities back to the destination image.

3.5 Pose determination from n Points (PnP)

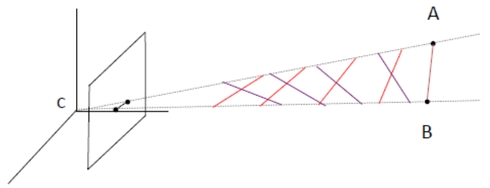
How many points are enough?

1. A single point



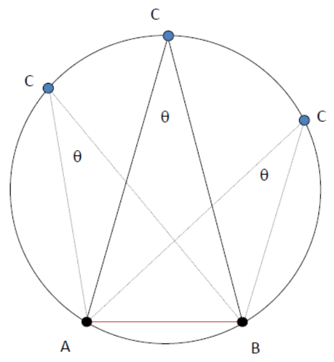
The camera can move along the line of projection. Thus we have infinitely many solutions.

2. Two Points

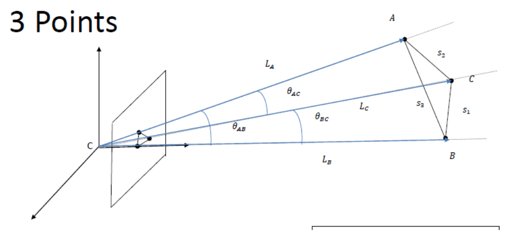


Since we don't know size and orientation of the line between the two points, the camera position is still undetermined.

Differently formulated the knowing of an angle between two points does not fix the location of the camera:



3. Three Points



From Carnot's theorem:

$$s_1^2 = L_B^2 + L_C^2 - 2L_BL_C\cos(\theta_{BC})$$
$$s_2^2 = L_A^2 + L_C^2 - 2L_AL_C\cos(\theta_{AC})$$
$$s_3^2 = L_A^2 + L_B^2 - 2L_AL_B\cos(\theta_{AB})$$

A system of polynomial equations in n unknowns can have no more solutions than the product of their respective degrees. In this case: 8. Since all terms are constant or quadratic half of the solutions are negative and thus invalid. This leaves us with a total of four valid solutions. **A forth point disambiguates these solutions.**

3.6 Camera Calibration: Direct Linear Transform (DLT)

Estimate **intrinsic** and **extrinsic** parameters.
Find the projection matrix m as

$$M = K[R|T]$$

Which can be used as follows:

$$\begin{bmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{bmatrix} = \begin{bmatrix}m_1^T\\ m_2^T\\ m_3^T\end{bmatrix}\begin{bmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{bmatrix}$$

where m_i^T are the rows of the projection matrix M.

Now assume that $P = \begin{bmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{bmatrix}$

And project the whole equation to pixel coordinates using

$$u = \frac{\tilde{u}}{\tilde{w}} = \frac{m_1^T\cdot P}{m_3^T\cdot P}$$
$$v = \frac{\tilde{v}}{\tilde{w}} = \frac{m_2^T cdot P}{m_3^T\cdot P}$$

thus

$$(m_1^T - u_i m_3^T)\cdot P_i = 0$$
$$(m_2^T - v_i m_3^T)\cdot P_i = 0$$

By re-arranging and writing in matrix form:

$$\underbrace{\begin{bmatrix}P_1^T & 0^T & -u_1P_1^T\\ 0^T & P_1^T & -v_1P_1^T\\ \vdots & \dots & \vdots\\ P_n^T & 0^T & -u_nP_n^T\\ 0^T & P_n^T & -v_nP_n^T\end{bmatrix}}_{Q}\underbrace{\begin{bmatrix}m_1\\ m_2\\ m_3\end{bmatrix}}_{M} = \begin{bmatrix}0\\ 0\\ \vdots\\ 0\\ 0\end{bmatrix}$$

where Q can be written out as

$$Q = \begin{bmatrix}X_w^1 & Y_w^1 & Z_w^1 & 1 & 0 & 0 & 0 & 0 & -u_1X_w^1 & -u_1Y_w^1 & -u_1Z_w^1 & -u_1\\ 0 & 0 & 0 & 0 & X_w^1 & Y_w^1 & Z_w^1 & 1 & -v_1X_w^1 & -v_1Y_w^1 & -v_1Z_w^1 & -v_1\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ X_w^n & Y_w^n & Z_w^n & 1 & 0 & 0 & 0 & 0 & -u_nX_w^n & -u_nY_w^n & -u_nZ_w^n & -u_n\\ 0 & 0 & 0 & 0 & X_w^n & Y_w^n & Z_w^n & 1 & -v_nX_w^n & -v_nY_w^n & -v_nZ_w^n & -v_n\end{bmatrix}$$

and M can be written out as

$$M = [m_{11} \quad m_{12} \quad m_{13} \quad m_{14} \quad m_{21} \quad m_{22} \quad m_{23} \quad m_{24} \quad m_{31} \quad m_{32} \quad m_{33} \quad m_{34} \quad m_{41} \quad m_{42} \quad m_{43} \quad m_{44}]^T$$

For a minimal solution, the matrix $Q_{(2nx12)}$ should have at least rank 11 in order to have a unique (up to scale), non-trivial solution M. Since each 3D-to-2D correspondence provides 2 independent equations we need at least $5 + \frac{1}{2}$ point correspondences, thus 6.
For an over-determined solution we can minimize the squarred error QM subject to $||M||^2 = 1$.
For maltab :`[U,S,V] = svd(Q); M = V(:,12);`

Once M is known the intrinsic and extrinsic parameters can be calculated using:

$$M = K(R|T)$$

Degenerate configurations:
There are certain combinations of 3D-2D correspondences which are degenerate and do not deliver additional information:

1. Points lying on a plane and/or along a single line passing through the projection center.
2. Camera and points on a twisted cubic (i.e. smooth curve in 3D space of degree 3)

3.6.1 Camera Calibration from 3D objects

Given the M matrix we can recover the extrinsic and intrinsic parameters based on:

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} = \begin{bmatrix} \alpha_u r_{11} + u_0 r_{31} & \alpha_u r_{12} + u_0 r_{32} & \alpha_u r_{13} + u_0 r_{33} & \alpha_u t_1 + u_0 t_3 \\ \alpha_v r_{21} + v_0 r_{31} & \alpha_v t_{22} + v_0 r_{32} & \alpha_v r_{23} + v_0 r_{33} & \alpha t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

To enforce that $R \cdot R^T = I$ we can use QR factorization of M, which decomposes M into a R(orthogonal), T, and upper triangular matrix i.e. K.
A 3D calibration example is $Tsai's 1987$:

1. Edge detection
2. Straight line fitting to the detected edges
3. Intersection of the edges to find the corners
4. Using more than 6 points, not all in a plane!

The parameters describing the resulting calibration are:

f_y	f_x/f_y	skew	x_0	y_0	residual
1673.3	1.0063	1.39	379.96	305.78	0.365

- The ratio f_x/f_y is not zero since the pixels were not squares.
- The skew indicates that the pixels were parallelograms.
- Today it can be mostly assumed that $\frac{\alpha_u}{\alpha_v} = 1$ and $K_{12} = 0$.
- The residual is the average reprojection error. Today algorithms are expected to deliver errors about around 0.2.

Definition 10. *The **reprojection error** is computed as the distance (in pixels) between the observed pixel point and the camera-reprojected 3D point.*

3.6.2 Camera Calibration from 2D objects

A 2D calibration example, which in contrast requires the points to lie on a plane is Zhang 1999.

1. Neglect radial distortion.
2. All points lie on a plane $\Rightarrow Z_w = 0$

$$\tilde{p} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}_{\text{Homography}} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

3. Conversion to pixel coordinates yields:

$$u = \frac{\tilde{u}}{\tilde{w}} = \frac{h_1^T \cdot P}{h_3^T \cdot P}$$
$$v = \frac{\tilde{v}}{\tilde{v}} = \frac{h_2^T \cdot P}{h_3^T \cdot P}$$

where $P = (X_w, Y_w, 1)^T$, which leads to the two equations of interest:

$$(h_1^T - u_i h_3^T) \cdot P_i = 0$$
$$(h_2^t - v_i h_3^T) \cdot P - i = 0$$

4. By rearranging

$$\underbrace{\begin{bmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \vdots & vdots & \vdots \\ P_n^t & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{bmatrix}}_{Q \text{ is known}} \underbrace{\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}}_{H \text{ is unkown}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

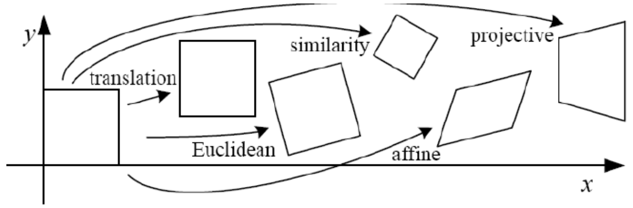
Minimal solution:

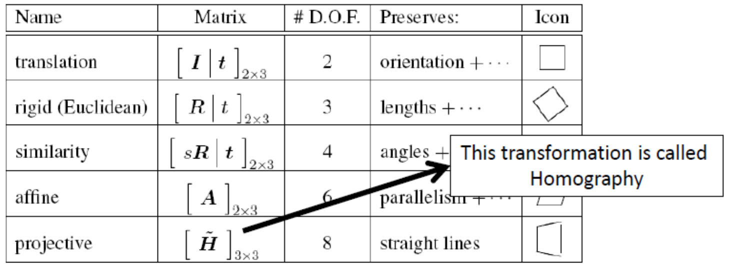
- $Q_{(2n\times 0)}$ should have rank 8 to have a unique (up to scale) non-trivial solution H.
- Each point correspondence provides 2 independent solutions, thus at least 4 **non-collinear points** is required.
- For an overdetermined solution $n > 4$ points SVD can deliver a solution.

Having found H, K and $[Rt]$ can be found making a QR decomposition such that:

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

3.6.3 Types of 2D Transformations





Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	□
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	◇
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles $+$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	▱
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	⏢

This transformation is called Homography

3.6.4 Find the camera pose from 2D-3D correspondences (DLT algorithm)

- Goal: Calculation of $[R|t]$ that satisfy the perspective projection equation:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{W} \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

- This leads to calculating the solution of $Q \cdot M = 0$, as described in section 3.6. The following caveats are to be expected:
  1. If M solves $Q \cdot M = 0$, αM does as well, thus M is recovered up to an unknown scaling factor.
  2. Remember that Q is built using normalized coordinates, not pixel coordinates!
  3. To solve the overdetermined system of equations the least squares approach is implemented using singular value decomposition. It can be shown that the solution of this problem is the eigenvector corresponding to the smallest eigenvalue of Q^TQ, which simply corresponds to the last column of V if S, where $Q = USV^T$, has its diagonal entries sorted in descending order. The `svd` function in matlab provides that.

```
[~,~,V] = svd(Q);
M = reshape(V(:,12),4,3)';
```

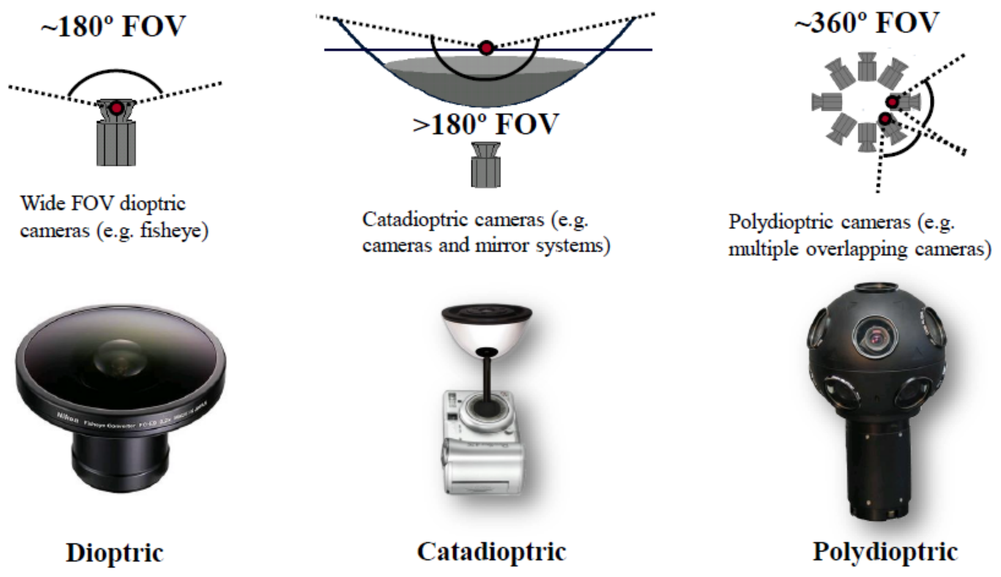
  4. To enforce R actually is a unitary matrix, we have to ensure that the M matrix, being a transformation matrix, actually makes a shift in positive z direction, thus $t_z = M_{34} > 0$.

```
if M(3,4)<0
    M=-M;
end
```

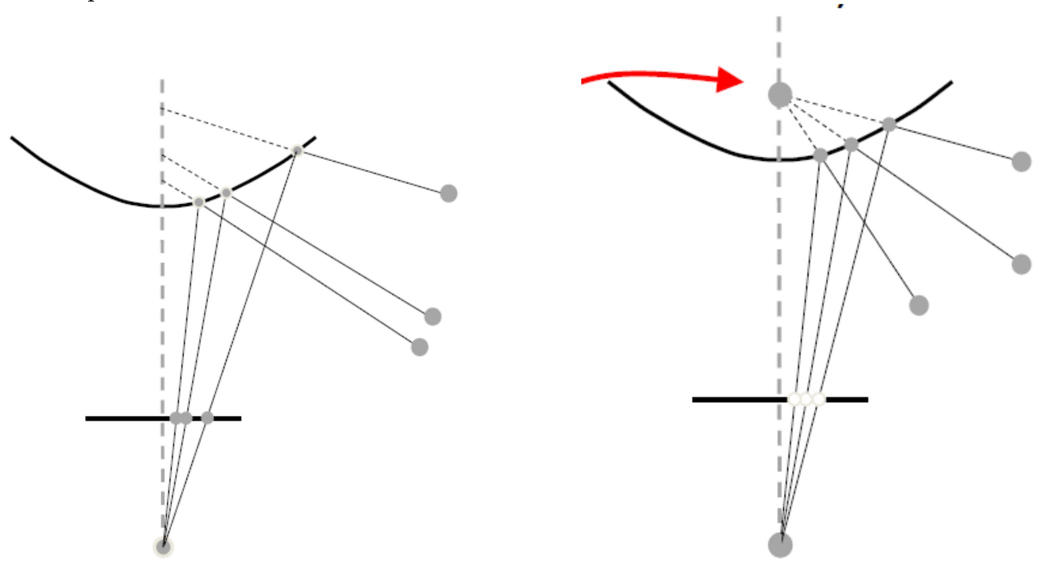
  5. Since the solution of the least squares problem only delivers the approximation of a homogeneous transformation matrix, we cannot be sure that R is actually a rotation matrix in $SO(3)$. To extract the actual rotation matrix from M, that is closest to the original estimated R, singular value decompositions is used. Hereby $R = USV^T$, which can be made unitary by removing the scaling factors in S. Thus $\tilde{R} = UV^T$.
  6. Since our solution M might actually be a scaled version αM. Now, having found the correct rotation matrix $\tilde{R}$, we can compare it to the original R to find the scale between the two. Thus $\alpha = \frac{||\tilde{R}||}{||R||}$. And the correct homogeneous transformation matrix is found as:

$$\tilde{M} = [\tilde{R}|\alpha t]$$

3.7 Omnidirectional Cameras


~180º FOV
>180º FOV
~360º FOV
Wide FOV dioptric cameras (e.g. fisheye)
Catadioptric cameras (e.g. cameras and mirror systems)
Polydioptric cameras (e.g. multiple overlapping cameras)
Dioptric
Catadioptric
Polydioptric

Catadioptric cameras come in different versions:

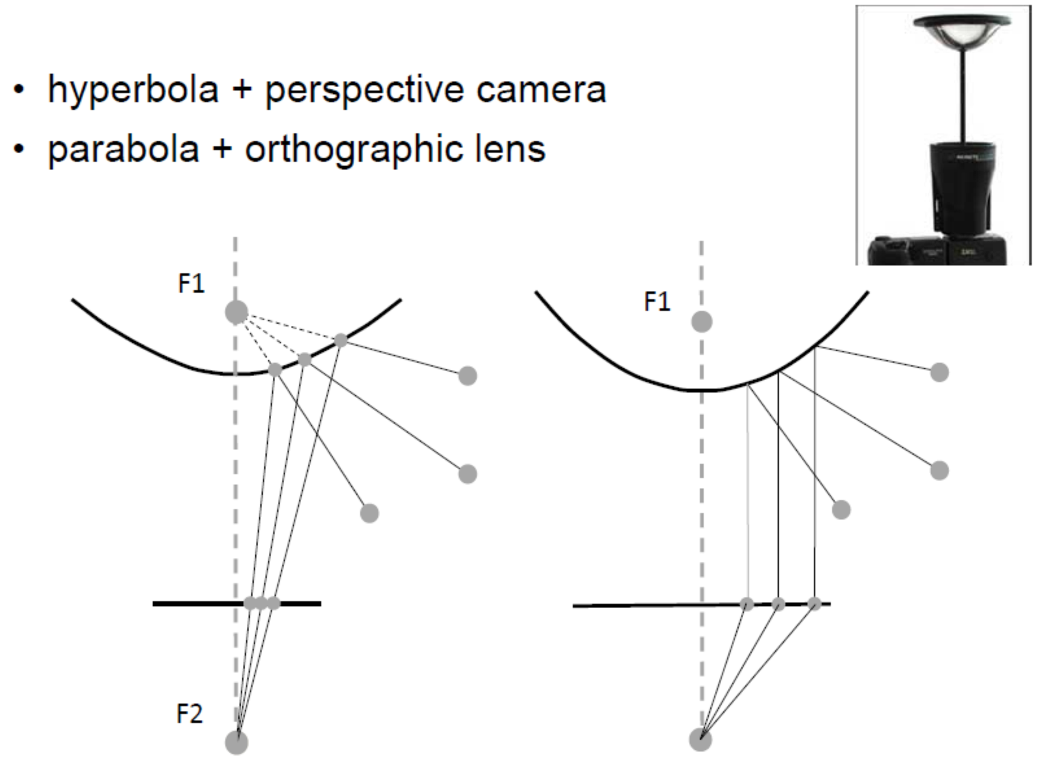


Definition 11. *A camera is a **central catadioptric camera** if the projection is such that there is a single effective viewpoint.*

It is important to have a single view point to enable:

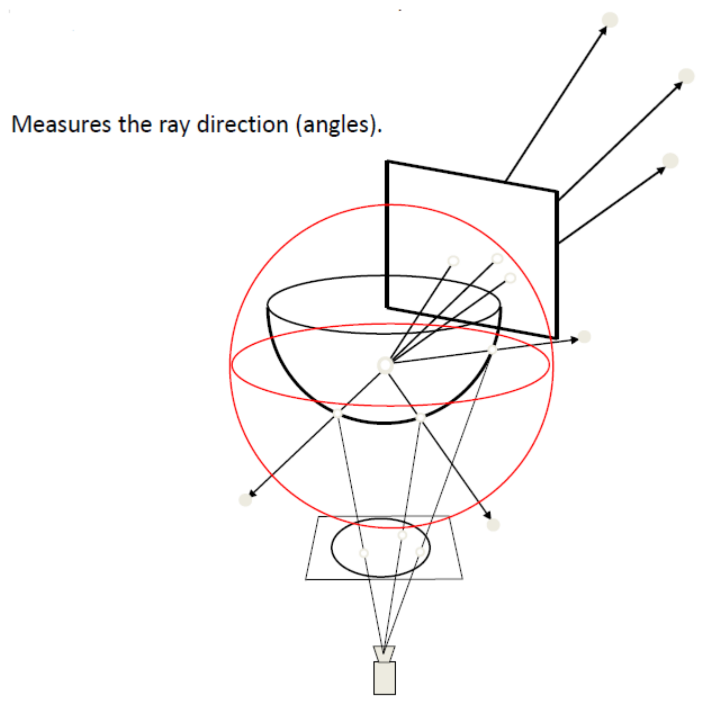
- Unwrapping an omnidirectional image into a perspective image.
- Transforming image points into normalized vectors on the unit sphere.
- Applying standard algorithms valid for perspective geometry.

3.7.1 Example for central catadioptric lenses



- hyperbola + perspective camera
- parabola + orthographic lens

3.7.2 Equivalence between Perspective and Omnidirectional Model



3.8 Digital Images

Definition 12. ***Matlab coordinates:*** $[rows, cols]$
C/C++ coordinates: $[cols, rows]$

4 Filtering

- A smoothing filter has positive values, always sums up to 1 to preserve the overall brightness of the picture and removes high-frequency contents, is thus a low-pass filter.
- A derivative filter has opposite signs, used to get high responses in regions of high contrast, sums to 0 and highlights high frequency components.

4.1 Types of noise

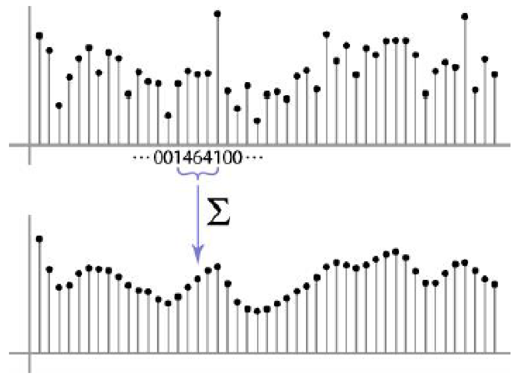
Definition 13. ***Salt and pepper noise****: random occurences of black and white pixels. Resulting from electromagnetive waves.*

Definition 14. ***Impulse noise****: random occurences of white pixels.*

Definition 15. ***Gaussian noise****: variations in intensity drawn from a Gaussian distribution. Very useful model for real world sensor noise.*

4.2 Noise removal

- Moving average filter. Based on the assumption of likeness of close pixels and the assumption of location-independent noise. It is possible to weigh pixels non-uniformly.



Definition 16. ***Convolution*** *defines the operation needed for the implementation of a moving average filter. One of the sequences is flipped and then slid over the other, multiplying each element with each other element and adding them up. A convolution is noted as $a_\star b$.*
Properties: linearity, associativity, commutativity

A convolution in 2D requires flipping the filter in both dimensions, which is equivalent to a 180° rotation. Then the convolution is defined as:

$$G[x,y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[x,y]H[x-u, y-v] \qquad G = F * H$$

Definition 17. *The* ***cross-correlation*** *is almost equivalent to a convolution, but does not include a flipping of the filter.*
Properties: linearity

$$G[x,y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[x,y]H[x+u, y+v]$$

4.3 Box filters

4.3.1 Moving Average Filter

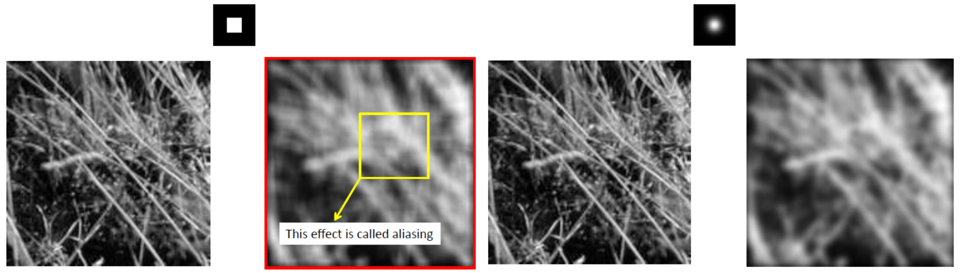
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \frac{1}{9} \quad \text{Unweighted}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \frac{1}{16} \quad \text{Weighted}$$

4.3.2 Gaussian Filter

$$H[u,v] = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

A finely resolving Gaussian filter will not introduce aliasing as a box filter would.



Filter with sharp edges will cause aliasing, since they introduce high frequency contributions. A gaussian filter has the property of being smooth in the image domain as well as in the frequency domain. It does not introduce high frequency contributions.

Important parameters:

- Size of the kernel (How large is the box?)

  Good choice: 3σ, since 90% of the information is contained within that span.

- Variance of the filter (How large is sigma σ) The larger the image the more intense the blur.

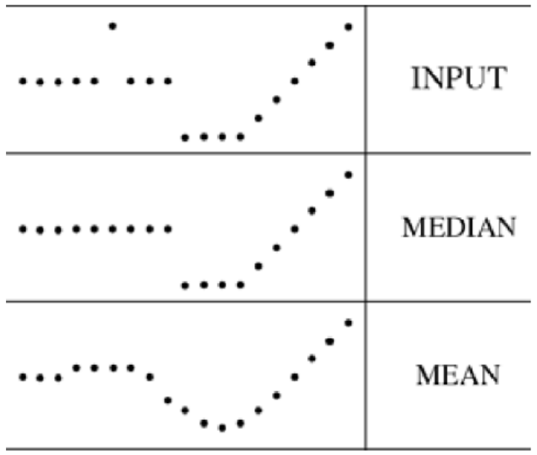
4.4 Boundary issues

Ho to treat boundaries when filtering an image?

- zero padding (black)
- wrap around
- copy edge
- reflect across edge

4.5 Median Filter

A linear smoothing filter will not remove salt and pepper noise, but lead to more corruption of the image. For this reason a median filter is applied.



A median filter preserves sharp transitions but removes small brightness variations.

4.6 High-Pass Filtering

To accomplish edge detection it makes sense to consider the first order derivatives of the image. For a 2D function $F(x,y)$ the partial derivative is:

$$\frac{\partial F(x,y)}{\partial x} = \lim_{\epsilon\to 0} \frac{F(x+\epsilon,y)-F(x,y)}{\epsilon}$$

For discrete data ϵ is set to 1.

$$\frac{\partial F(x,y)}{\partial x} \approx \frac{F(x,+1,y)-F(x,y)}{1}$$

Possibilities of filters implementing that are:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{Prewitt filter}$$

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{Sobel filter}$$

Since both direction are needed to represent all intensity changes within the image the gradient is used:

$$\nabla F = \left[\frac{\partial F}{\partial x}, \quad \frac{\partial F}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\frac{\partial F}{\partial y}}{\frac{\partial F}{\partial x}}\right) \quad \text{Gradient Direction}$$

$$||\nabla F|| = \sqrt{\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2} \quad \text{Edge strength}$$

In order for the high pass filter not to pick up noise instead of the edge to be detected, the image needs smoothing before the edge detection. Convolution allows convolution of the two filters which, if using a gaussian for smoothing, is equivalent to filtering by the derivative of a gaussian filter.

This is implemented by the **Canny edge-detection algorithm (1986)**.

1. Conversion to grayscale.

2. Application of smoothing and gradient filter. (Derivatives of Gaussian)

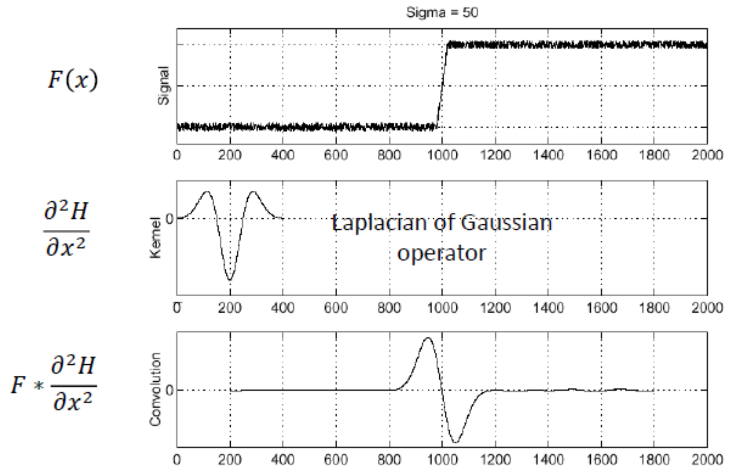
3. Plotting of the edge strength.

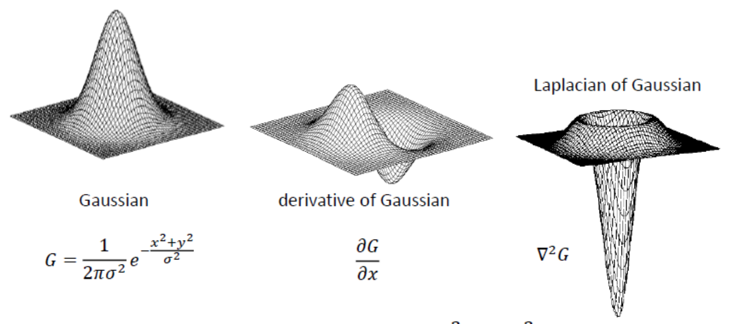
4. Thresholding of the image. Setting all pixels below threshold to zero.

5. Non-maxima suppression (local-maxima detection) along edge direction.

4.7 Laplacian of the Gaussian

Instead of analysing the first derivative (searching for maxima to find edges) we can take the second derivative and detect the same maximum by finding the zero crossing of the second derivative.




Gaussian
derivative of Gaussian
Laplacian of Gaussian
$G = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$
$\frac{\partial G}{\partial x}$
$\nabla^2 G$

- ∇^2 is the Laplacian operator: $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

5 Point Feature Detection and Matching

5.1 Template Matching

The correlation of a filter (a template) and the image can be used to find the location of the template, as the maximum of the filtered image. The matching will only work though if scale, orientation, illumination and the in general the appearance of the object and the template are similar.

5.1.1 Similarity Measures

Definition 18. *The **Sum of Absolute Differences (SAD)** is defined as*

$$SAD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} |H(u,v) - F(u,v)|$$

Definition 19. *The **Sum of Squared Differences (SSD)** is defined as*

$$SSD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} (H(u,v) - F(u,v))^2$$

SSD is computationally expensive.

Definition 20. *The **Normalized Cross Correlation (NCC)** takes values between* -1 *and* $+1$ *where (*$+1$ *is taken if the images are identical). It is defined as*

$$NCC = \frac{\sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)F(u,v)}{\sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)^2}\sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v)^2}}$$

To account for illumination differences the mean of each image is subtracted before calculating similarity. This leads to the definition of

Definition 21. ***Zero-mean SAD, SSD, NCC***

$$\mu_H = \frac{\sum\limits_{u=-k}^{k}\sum\limits_{v=-k}^{k} H(u,v)}{(2N+1)^2}$$

$$\boxed{ZSAD = \sum_{u=-k}^{k}\sum_{v=-k}^{k} |(H(u,v) - \mu_H) - (F(u,v) - \mu_F)|}$$

$$\boxed{ZSSD = \sum_{u=-k}^{k}\sum_{v=-k}^{k} ((H(u,v) - \mu_H) - (F(u,v) - \mu_F))^2}$$

The above are note invariant the affine illumination changes, ZNCC is.

$$\boxed{ZNCC = \frac{\sum\limits_{u=-k}^{k}\sum\limits_{v=-k}^{k}(H(u,v)-\mu_H)(F(u,v)-\mu_F)}{\sqrt{\sum\limits_{u=-k}^{k}\sum\limits_{v=-k}^{k}(H(u,v)-\mu_H)^2}\sqrt{\sum\limits_{u=-k}^{k}\sum\limits_{v=-k}^{k}(F(u,v)-\mu_F)^2}}}$$

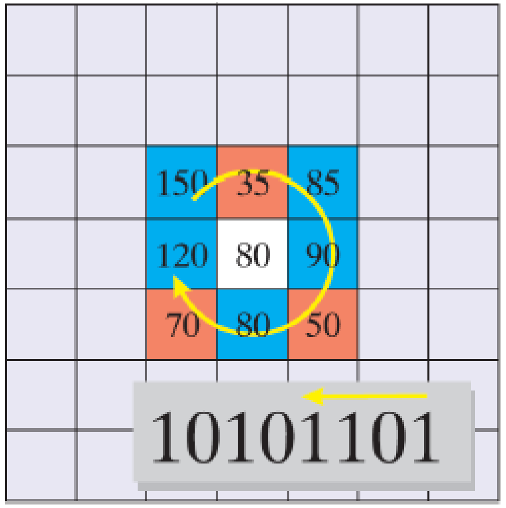
Definition 22. ***Affine intensity changes*** *are defined as*

$$I'(x,y) = \alpha I(x,y) + \beta$$

5.1.2 Census Transform

Definition 23. *The* ***Hamming distance*** *of two strings is the number of bits that are different.*

The census transform maps an image patch to a bit string (value larger than center pixel $\rightarrow$ 1 otherwise $\rightarrow$ 0) and compares strings using the Hamming distance.



Advantages

- More robust to object-background problem (same object, different background yields less similarity).
- No square roots or division required, thus very efficient, especially on FPGA.
- Intensities are considered relative to the center pixel of the patch making it invariant to monotonic intensity changes.

Definition 24. ***FPGA*** *is a field programmable gate array. Thus an integrated circuit designed to be configured by a customer or a designer after manufacturing.*

5.2 Feature Matching

Challenges

- Find distinctive features.
- Account for rotations, translations, distortions, color changes, illumination changes, lens-imperfections.
- Find the same feature in both images (repeatability).
- Match the corresponding points.

Definition 25. *A* ***corner*** *is defined as the intersection of one or more edges.*

+ A corner has high localization accuracy (very good for VO)

- Less distinctive than a blob.

E Harris, Shi-Tomasi, SUSAN, FAST

Definition 26. *A* ***blob*** *is any other image pattern, which is not a corner, that differs significantly form its neighbours in intensity and texture.*

- Has less localization accuracy than a corner.

+ Blob detectors are better for place recognition.

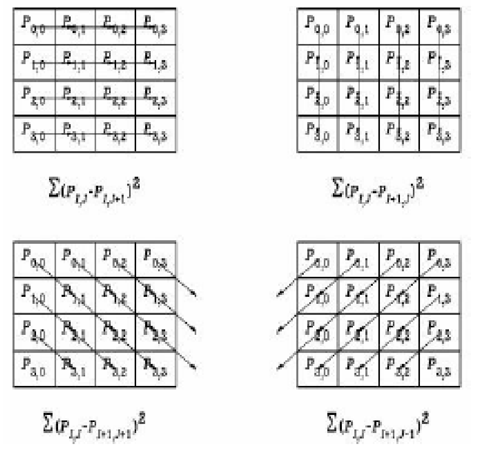
+ More distinctive than a corner.

E MSER, LOG, DOG (SIFt), SURF, CenSurE

5.2.1 The Moravec Corner Detector

1. A corner has a significant change in SSD in at least 2 directions and is thus repeatable and distinctive.

2. Sums of squares of differences of pixels adjacent in each of four directions (horizontal, vertical and two diagonals) over each window are calculated, and the window's interest measure is the minimum of these four sums. This makes sense since a corner is indicated by changes in all directions.


$\Sigma(P_{i,j}-P_{i,j+1})^2$
$\Sigma(P_{i,j}-P_{i+1,j})^2$
$\Sigma(P_{i,j}-P_{i+1,j+1})^2$
$\Sigma(P_{i,j}-P_{i+1,j-1})^2$

5.2.2 Harris Corner Detector

1. consider the reference patch centered at (x,y) and the shifted windows centered at $(x+\Delta x, y+\delta y)$. The patch has size P.

2. Apply SSD

3. $I_x = \frac{\partial I(x,y)}{\partial x}, \quad I_y = \frac{\partial I(x,y)}{\partial y}$, approximated with first order Taylor:
$$I(x+\Delta x, y+\Delta y) \approx I(x,y) + I_x(x,y)\Delta x + I_y(x,y)\Delta y$$

4. Thus SDD can be approximated as
$$SSD(\Delta x, \Delta y) \approx \sum_{x,y\in P} (I_x(x,y)\Delta x + I - y(x,y)\Delta y)^2$$

5. This can be written in matrix form:
$$SSD(\Delta x, \Delta y) \approx \sum_{x,y\in P} \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix} \Rightarrow SSD(\Delta x, \Delta y) \approx \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} M \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$
where $M = \sum_{x,y\in P} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum I_x^2 & \sum I_xI_y \\ \sum I_xI_y & \sum I_y^2 \end{bmatrix}$.

6. For any corner rotated with ϕ: $M = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix}$

   If any λ is close to zero there is no corner.

7. Using the singular value decomposition can be used to find λ_i directly.

8. λ_i can be used to identify a corner. A corner has been found if the minimum of the two eigenvalues is larger than a certain threshold.

9. **Cornerness function** $R = \min(\lambda_1, \lambda, 2)$

10. The corner detector using this criterion is called **Shi-Tomasi** detector.

11. Alternatively, to avoid the calculation of the eigenvalues, another cornerness function can be defined: $R = \lambda_1 - \lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k\text{trace}^2(M)$, where $k \in (0.04, 0.15)$.

5.3 Scale changes

- Scale changes are difficult to detect using a patch of a defined size.

- A possible solution would be rescaling the patch in order to match a feature with itself after a scale change.

- To remedy to computational intensity included in testing all n patches for s different scales a possible solution would be to assign a scale to each feature.

5.4 Automatic Scale Selection

- Idea: Define a function that is invariant to scale changes.

- $f(x, y, patch - size)$ would be the candidate function that has a value for each patch size. Now this function has a maximum at a certain patch size, which allows defining the scale of the detected feature.

- **This has to be done for each feature individually!**

- After finding the scale of a picture it can be normalized, this reduces the computational effort of finding matches of features in different images can be reduced significantly, since the iteration over patch sizes is not needed anymore.

- If there are multiple local maxima, the feature is replicated multiple times at the corresponding scales. The computational effort is still reduced that way.

5.4.1 How to find that function?

- Convolution with a kernel that highlights edges:

$$f = Kernel * Image$$

- The Laplacian of Gaussian kernel is most effective under certain conditions.

$$L \circ G = \nabla G(x, y)$$

- The $L \circ G$ Kernel already includes smoothing.

5.4.2 How to describe features?

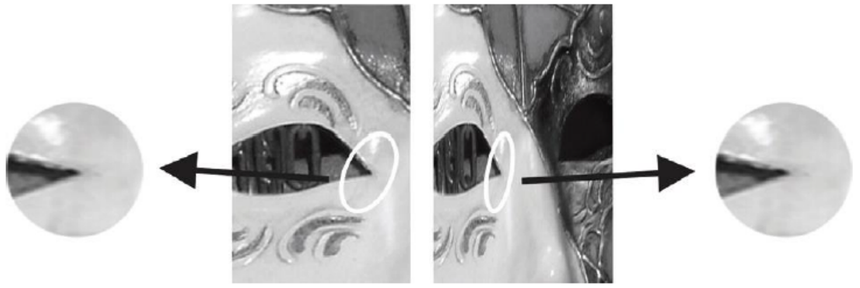
- Idea: Find a feature descriptor that is **invariant** to geometric and photometric changes.

5.4.3 How to achieve invariance with Patch descriptors?

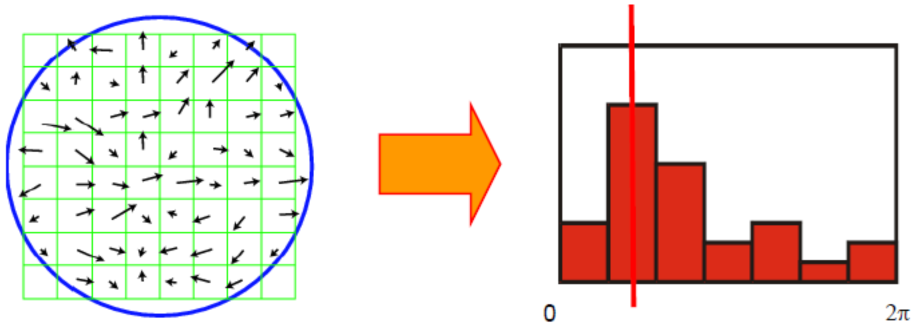
1. Re-scaling and De-rotation:
   a) Find correct scale using LoG operator.
   b) Rescale the patch.
   c) Find local orientation (Dominant direction of gradient (Harris eigenvectors)).
   d) De-rotate patch through patch warping. $\rightarrow$ Canonical orientation.
      - Start with an empty canonical patch.
      - For each pixel in the destination patch find the corresponding location in the source patch as defined by the **warping function**.
      - Interpolate in the source frame to find the intensity in the destination frame.
      - **Roto-Translational Warping:**

$$x' = x\cos\theta - y\sin\theta + a$$
$$y' = x\sin\theta + y\cos\theta + b$$

      - **Affine Warping:** The second moment matrix M can be used to identify the two directions of fastest and slowest change of intensity, which can be used to define an elliptic patch, that is then normalized to a circular patch.



2. **Disadvantages:**
   - If warping not accurate the matching score decreases significantly.
   - Computationally expensive.
3. HOG descriptor (Histogram of Oriented Gradients)
   a) Multiply the patch by a Gaussian kernel.
   b) Compute gradients vectors at each pixe.
   c) Build a histogram of gradient orientations, weighted by the gradient magnitudes.
   d) Extract all local maxima and make a descriptor (HOG) for each.
   e) Apply a circular shift to the descriptor such that the detected maximum corresponds with 0 degrees.



5.5 SIFT Descriptor

1. Multiply the patch by a Gaussian filter
2. Divide the patch into 4x4 sub-patches
3. Compuge HOG (8bins, i.e. 8 directions) for all pixels inside each sub-patch.
4. Concatenate all HOGs into a single 1D vector. $4x4x8 = 128$ values.
5. The descriptor vector v is then normalized such that:

$$\underline{\overline{v}} = \frac{v}{\sqrt{\sum_i^n v_i^2}}$$

This guarantees invariance to linear illumination changes. Overall the SIFT descriptor is invariant to affine illumination changes.

+ Can handle severe viewpoint changes (up to 50°.

+ Can handle even non affine changes in illumination (low to bright scenes).

- Computationally expensive: 10 frames per second on an i7.

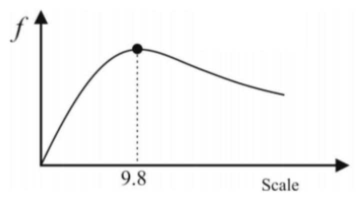
5.6 SIFT detector

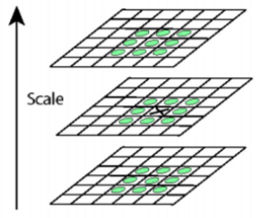
Idea: Detect keypoints as local extrema over the image as well as over the patch scale, using a Difference of Gaussian (DoG) kernel.

1. Incrementally convolve the initial image with Gaussians $G(k^i\sigma)$ to produce blurred images separated by a constant factor k in scale space.
   a) Initial Gaussian: $\sigma = 1.6$.
   b) k is chosen. $k = 2^{1/s}$ where s is the number of intervals into which each octave of scale space is divided.
   c) For efficiency reasons, when k^i equals 2, the images is downsampled by a factor of 2 and the procedure is repeated again up to 5 octaves. Downsampling and reusing the gaussian kernels is equivalent to increasing i further, but much more computationally efficient.

2. Ajdacent blurred images are then subtracted to produce the Difference-of-Gaussian (DoG) images.

3. Local Maxima a found in the resulting scales (scale = 3 DoGs).





An efficient approach to that problem is using an image dilation algorithm for detecting maxima. `imdilate(image,mask)`

For a twodimensional peak-search the algorithm using imdilate would do the following:

a) Define the mask as: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

b) Then the dilation serves to calculate the neighbouring maximum for each pixel.

c) Finally the resulting dilated image is compared to the original image. The local maxima are found as `original>filtered` which in Matlab results in a logical array, where 1s indicate a local maximum. In other words this comparison yields `true` if the pixel is larger than the maximum of its neighbours.

In this case the used mask is the 3D equivalent of the mask used above.

5.6.1 Summary

- Based on the property of the LoG allowing to recognize features of a certain radius, and on the idea that that radius may be changed by changing the standard deviation of the gaussian curve, one can vary the peak-finding function over the scale by varying σ. In addition, instead of increasing the radius of the filter, the image size is decreased for the next octave of Gaussians.

6 Matlab