Theory Questions

Lecture 01

1. (a) *Definition of VO*

   Answer. 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.

   (b) *VO vs VSLAM vs SFM.*

   Answer.

   $$\text{SFM} > \text{VSLAM} > \text{VO}.$$

   Structure From Motion (SFM): more general than VO, tackles the problem of 3D reconstruction and 6DOF pose estimation from **unordered image sets** (e.g. reconstruction through flickr).
   $\rightarrow$ VO focuses on estimating 3D motion **sequentially** and in **real time**.
   Two images can be taken from the same camera but at different positions and at different times, where the **pose** is not **unknown**

   Visual Simultaneous Localization And Mapping (VSLAM):

   - VO focuses on incremental estimation and **local consistency**.
   - focus on **globally consistent** estimation.
   - VSLAM = VO + loop detection + graph optimization.
   - Tradeoff between performance and consistency, simplificity of implementation.
   - VO doesn't need to keep track of all previous history of the camera. Good for real-time.

   (c) *Assumptions*

   Answer.

   - **Sufficient illumination**,
   - **dominance of static scene** over moving objects,
   - **enough texture** to allow apparent motion,
   - **sufficient scene overlap** between consecutive frames.

   (d) *Working Principle*

   Answer.

   I) Compute the relative motion T_k from images I_{k-1} to image I_k

   $$T_k = \begin{pmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{pmatrix}. \tag{1}$$

   II) Concatenate them to recover full trajectory

   $$C_n = C_{n-1} \cdot T_n \tag{2}$$

III) An optimization over the last m poses can be done to refine locally the trajectory (Pose-Graph or Bundle Adjustment).

(e) *Building Blocks*

Answer. VO computes the camera path incrementally, pose per pose

i. Image sequence,
ii. Feature detection (**front end**),
iii. Feature matching (tracking) (**front end**).
iv. Motion estimation (2D-2D,3D-3D,3D,2D) (**front end**),
v. Local optimization (**back end**).

(f) *Dense vs. Semi-Dense vs. Sparse*

Answer. Dense and Semi-dense behave similarly in direct methods. Dense is only useful if one has motion blur and defocus. Sparse methods behave equally well for image overlaps up to 30% but less robust than the other two when the distance between frames increases.

(g) *Bundle Adjustment vs. Pose Graph Optimization (formulas and explanation)*

Answer.

Pose Graph Optimization

So far we assumed that the transformations are between consecutive frames, but transformation can be computed also between non adjacent frames T_{ij} (e.g. when features from previous keyframes are still observed). They can be used as additional constraints to improve cameras poses by minimizing the following:

$$C_k = \text{argmin}_{c_k} \sum_i \sum_j ||C_i - C_j \cdot T_{ij}||^2 \tag{3}$$

- For efficiency, only the last m keyframes are used.
- Gauss-Newton or Levenber-Marquadt are typically used to minimize it. For large graphs, there are open source things.

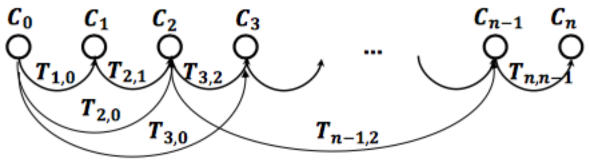


Figure 1: Pose graph optimization.

Bundle Adjustment (BA)

This incorporates the knowledge of landmarks (3D points).

$$X^i, C_k = \text{argmin}_{X^i, C_k} \sum_i \sum_k \rho\left(p_k^i - \pi(X^i, C_k)\right). \tag{4}$$

Outliers are a problem, how can we penalize them? In order to penalize wrong matches, we can juse the Huber or Turkey cost.

$$\textbf{Huber} \qquad \rho(x) = \begin{cases} x^2, & if\ |x| \leq k \\ k \cdot (2|x| - k) & if\ |x| \geq k \text{ linear} \end{cases}$$
$$\textbf{Tukey} \qquad \rho(x) = \begin{cases} \alpha^2 & if\ |x| \geq \alpha \\ \alpha^2 \cdot \left(1 - (1 - (\frac{x}{\alpha})^2)^3\right) & if\ |x| \leq \alpha. \end{cases} \tag{5}$$

Bundle Adjustment vs Pose-graph Optimization

- BA is more precise than pose-graph optimization because it adds additional constraints (landmark constraints).
- But **more costly**: $O((qM + lN)^3)$ with M and N being the number of points and camera poses and q and l the number of parameters for points and camera poses. The Jacobian is cubic in q and l. Workarounds are
  - A small window size limits the number of parameters for the optimization and thus makes real-time bundle adjustment possible.
  - It is possible to reduce the computational complexity by just optimizing the camera parameters and keeping the 3D landmarks fixed, e.g. **freeze the 3D points and adjust the poses**

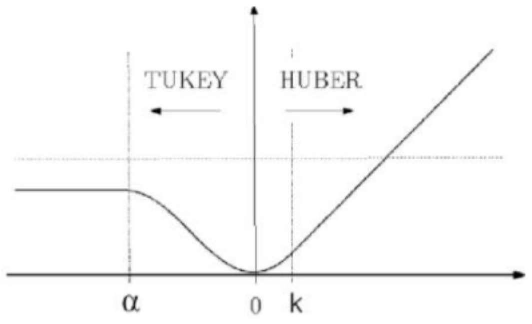


Figure 2: Tukey vs. Huber norm.

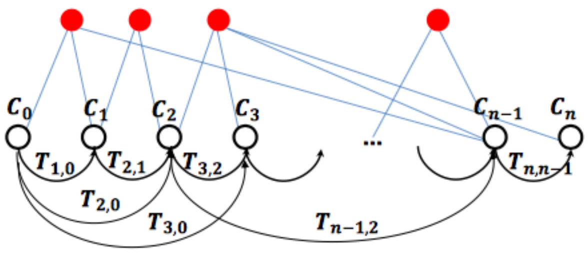


Figure 3: Bundle Adjustment.

Lecture 02/03

1. *Blur Circle*

   Answer. In optics, a circle of confusion (or blur circle) is an optical spot caused by a cone of light rays from a lens, not coming to a perfect focus when imaging a point source.

   - There is a specific distance from the lens at which world points are in focus in the image.
   - Other points project to a **blur circle** in the image with radius
     $$R = \frac{L\delta}{2e} \tag{6}$$
     $\rightarrow$ a minimal pinhole gives minimal R and
     $\rightarrow$ R should remain smaller than image resolution.

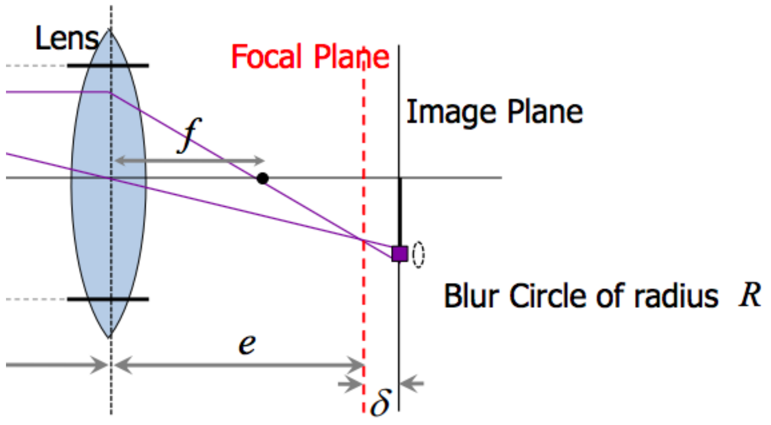


Figure 4: Blur Circle

2. *Thin Lens Equation and Pinhole Approximation*

   Answer. Starting from Figure 5, one can use similar triangles and write
   $$\begin{aligned} \frac{B}{A} &= \frac{e}{z} \text{ and} \\ \frac{B}{A} &= \frac{e-f}{f} = \frac{e}{f} - 1. \end{aligned} \tag{7}$$
   Toghether, we get the **thin lens equation**.
   $$\frac{e}{f} - 1 = \frac{e}{z} \tag{8}$$
   By dividing the thin lens equation by parameter e and considering $z >> f$ we get
   $$\begin{aligned} \frac{1}{z} &= \frac{1}{f} - \frac{1}{e} \\ 0 &= \frac{1}{f} - \frac{1}{e} \\ e &\approx f \end{aligned} \tag{9}$$

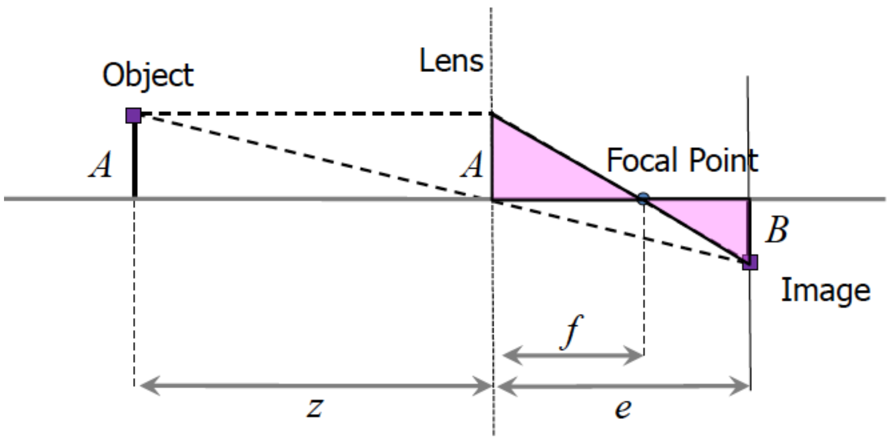


Figure 5: Thin lens

3. (a) *Definition of vanishing points and lines*

   Answer.

   - Straight lines are still straight.
   - Length and angles are lost.
   - Parallel lines in the world intersect in the image as a **vanishing point**.
   - Parallel planes in the world intersect in the image at a **vanishing line**.

   (b) *Prove that the parallel lines intersect at vanishing points and show how to compute it mathematically.*

   Answer.

   Claim 1. World's parallel lines intersect at a vanishing point in the camera image.

   Proof. Let's define two parallel 3D lines:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} + s \cdot \begin{pmatrix} l \\ m \\ n \end{pmatrix}, \qquad \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \end{pmatrix} + s \cdot \begin{pmatrix} l \\ m \\ n \end{pmatrix}. \tag{10}$$

   The perspective projection equation in calibrated coordinates are

$$x = \frac{X}{Z} \qquad y = \frac{Y}{Z}. \tag{11}$$

   Plugging the line equations into the perspective projection equation and letting $s \to \infty$ results in

$$\lim_{s\to\infty} \frac{X_i + sl}{Z_i + sn} = \frac{l}{n} = x_{VP} \qquad \lim_{s\to\infty} \frac{Y_i + sm}{Z_i + sn} = \frac{m}{n} = y_{VP}. \tag{12}$$

This result depends only on the direction vector of the line. □

In Figure 6 another proof is displayed, the lines intersect to a point at infinity and the projection of such point in the image returns the vanishing point **v**. **v** is related to **d** via Eq. 24, where K is the camera matrix. Eq. 25 guarantees that d has unit norm.

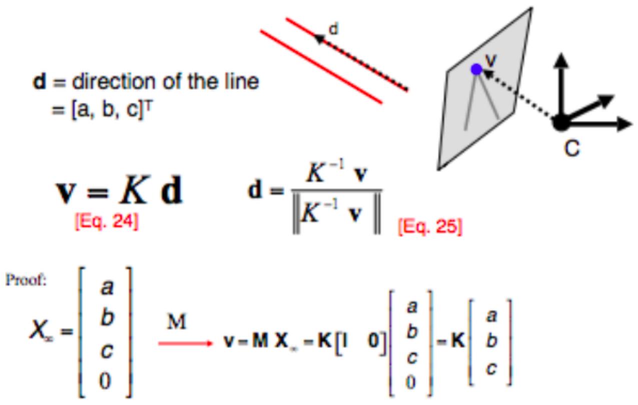

d = direction of the line
= [a, b, c]^T
d
v
C
$\mathbf{v} = K\,\mathbf{d}$
[Eq. 24]
$\mathbf{d} = \frac{K^{-1}\mathbf{v}}{\|K^{-1}\mathbf{v}\|}$
[Eq. 25]
Proof:
$X_\infty = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \xrightarrow{M} \mathbf{v} = \mathbf{M}\,\mathbf{X}_\infty = \mathbf{K}[\mathbf{I}\ \ 0]\begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} = \mathbf{K}\begin{bmatrix} a \\ b \\ c \end{bmatrix}$

Figure 6: Vanishing point

4. *How do you build an Ames room (the floor and the walls); try to sketch a concept*

   Answer. Have a look at Figure 7

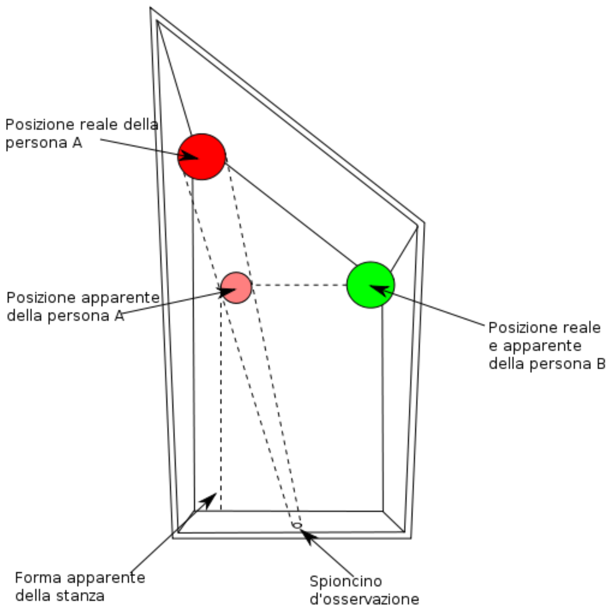

Posizione reale della persona A
Posizione apparente della persona A
Posizione reale e apparente della persona B
Forma apparente della stanza
Spioncino d'osservazione

Figure 7: Thin lens

5. **Relation between field of view and focal length.**

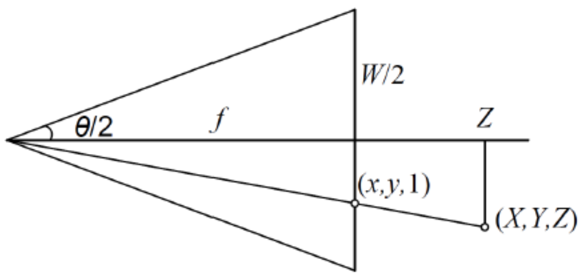


Figure 8: Scheme for angles.

Answer. With Figure 8, one can compute the **field of view**

$$\tan\left(\frac{\theta}{2}\right) = \frac{W}{2f} \Rightarrow f = \frac{W}{2}\left[\tan\left(\frac{\theta}{2}\right)\right]^{-1} \tag{13}$$

$\rightarrow$ smaller FOV = larger focal length.

6. (a) *Perspective projection equations including lens distortion and world to camera projection (derivation of perspective equations in matrix form using homogeneous coordinates).*

   Answer. The procedure reads

   i. Change of coordinates from 3D world point P_w to 2D camera frame point P_c .
   ii. Projection from the camera frame to the image plane (x, y).
   iii. Change in pixel (u, v).

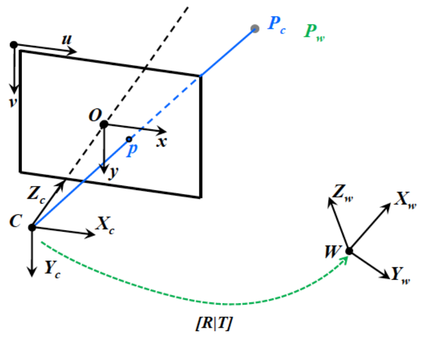


Figure 9: Frames. Z_c = optical axis

Perspective Projection
From similar triangles and Figure 10 one gets from Camera Point P_c to **image plane** coords

$$\frac{x}{f} = \frac{X_c}{Z_c} \Rightarrow x = \frac{fX_c}{Z_c}$$
$$\frac{y}{f} = \frac{Y_c}{Z_c} \Rightarrow y = \frac{fY_c}{Z_c} \tag{14}$$

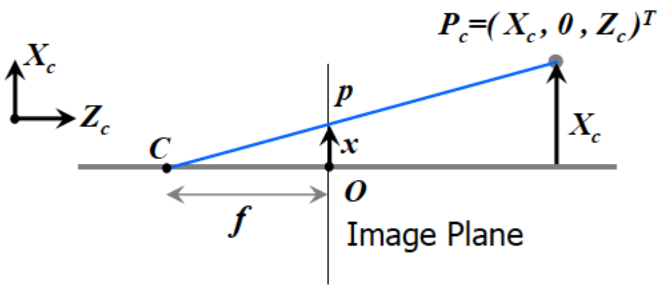


Figure 10: Figure for perspective projection.

Pixel Coordinates

From local **image plane** coords (x, y) to the **pixel** coords (u, v) with scale factors k_u, k_v and pixel coords of the camera optical center $O = (u_0, v_0)$:

$$\begin{aligned} u &= u_0 + k_u x \Rightarrow u = u_0 + \frac{k_u f X_c}{Z_c} \\ v &= v_0 + k_u x \Rightarrow v = v_0 + \frac{k_v f X_c}{Z_c} \end{aligned} \tag{15}$$

Homogeneous coords for linear mapping from 3D $\rightarrow$ 2D by introducing an extra scalar element (the depth of the scene point $\lambda = Z_c$)

$$p = \underbrace{\begin{pmatrix} u \\ v \end{pmatrix}}_{pixel} \rightarrow \tilde{p} = \underbrace{\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix}}_{homog.} = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{16}$$

with (15) and (16) expressed in matrix form reads

$$\begin{pmatrix} \lambda u \\ \lambda v \\ \lambda \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = K \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}, \tag{17}$$

with α_u, α_v focal lengths in pixels and K **calibration matrix**.
At the end, it holds

$$\lambda \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{(K)}_{3\times 3,\ intrinsic} \cdot \underbrace{(\mathbb{I}_{3\times 3}|0)}_{3\times 4} \cdot \underbrace{\begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix}}_{4\times 4,\ extrinsic} \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}. \tag{18}$$

We use the notation from homogeneous to euclidean:

$$\underbrace{\begin{pmatrix} u \\ v \\ w \end{pmatrix}}_{Homog.} \rightarrow \underbrace{\begin{pmatrix} u/w \\ v/w \\ 1 \end{pmatrix}}_{Eucl.} \tag{19}$$

Radial Distortion This is a transformation from ideal to distorted coordinates. For most lenses, one writes

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \underbrace{(1 + k_1 r^2)}_{\substack{\text{higher order dep on} \\ \text{amount of distortion } r^4 \\ \text{etc.}}} \cdot \begin{pmatrix} u - u_0 \\ v - v_0 \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}, \tag{20}$$

with

$$r^2 = (u - u_0)^2 + (v - v_0)^2. \tag{21}$$

Based on the Brown-Conrady model. **Barrel** distortion has distortion coef $k_1 < 0$. **Ask someone:** what is the relation between order of the term to the amount of distortion????

(b) *Normalized image coordinates and geometric explanation.*

7. (a) *Definition of general PnP problem (whats the minimum number of points and what are the degenerate configurations).*

Answer. Given the realitve spatial locations of n control points and given the angle to every pair of control points from an additional point called the Center of Perspective C_P, find the lengths of the line segments joining C_P to each of the control points.
We assume we know the camera intrinsic parameters. Given known 3D landmarks in the world and their image correspondence in the camera frame, determine the 6DOF pose of the camera in the world frame. **Where is the camera?**

- Given 1 point: ∞ solutions.
- Given 2 points: ∞ *bounded* solutions.
- Given 3 **non collinear** points: Finitely many (up to 4) solutions.
- Given 4 points: unique solution.

With 3 points one can use the fact that the angles inscribed in the triangle are the same: the Carnot's theorem reads

$$s_{1,2,3}^2 = L_{B,A,A}^2 + L_{C,C,B}^2 - 2L_{B,A,A}L_{C,C,B}\cos(\theta_{BC,AC,AB}) \tag{22}$$

In general, n independent polynomials with n unknowns, can have no more solution than the **product** of their degrees: here 8.
$\rightarrow$ fourth point to **disambiguate** the solutions! By defining $x = \frac{L_B}{L_A}$ we can reduce the system to a 4^{th} order equation

$$G_0 + G_1 x + G_2 x^2 + G_3 x^3 + G_4 x^4 = 0. \tag{23}$$

This applies to camera pose estimation from known $3D - 2D$ correspondences (e.g. hololens).

- L: distance from Camera frame origin C to World point A,B,C
- θ: inscribed angles between e.g. L_A and L_B
- s: distance between World points e.g. A and B

(b) *Working principle of P3P algorithm (non-linear algorithm for calibrated cameras: what are the algebraic trigonometric equations that it attempts to solve?).*

(c) *How do we solve PnP using a linear algorithm (derive DLT equations for 3D object or planar grids) and what is the minimum number of point correspondences it requires, why?*

Answer. Direct Linear Transform

$$\begin{aligned}
\text{image point} = \tilde{p} = \begin{pmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{pmatrix} &= \lambda \begin{pmatrix}u\\ v\\ 1\end{pmatrix} = K[R|T]\cdot\begin{pmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{pmatrix}\\
&= \begin{pmatrix}\alpha_u & 0 & u_0\\ 0 & \alpha_v & v_0\\ 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}r_{11} & r_{12} & r_{13} & t_1\\ r_{21} & r_{22} & r_{23} & t_2\\ r_{31} & r_{32} & r_{33} & t_3\end{pmatrix}\cdot\begin{pmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{pmatrix}\\
\textit{assuming indep. elements} \qquad &= \underbrace{\begin{pmatrix}m_{11} & m_{12} & m_{13} & m_{14}\\ m_{21} & m_{22} & m_{23} & m_{24}\\ m_{31} & m_{32} & m_{33} & m_{34}\end{pmatrix}}_{M}\cdot\begin{pmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{pmatrix}\\
&= \begin{pmatrix}m_1^T\\ m_2^T\\ m_3^T\end{pmatrix}\cdot\underbrace{\begin{pmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{pmatrix}}_{P}.
\end{aligned} \tag{24}$$

It follows

$$\begin{aligned}
u &= \frac{\tilde{u}}{\tilde{w}} = \frac{m_1^T\cdot P}{m_3^T\cdot P}\\
v &= \frac{\tilde{v}}{\tilde{w}} = \frac{m_2^T\cdot P}{m_3^T\cdot P}
\end{aligned} \tag{25}$$

and hence

$$\begin{aligned}
(m_1^T - u_i m_3^T)\cdot P_i &= 0\\
(m_2^T - v_i m_3^T)\cdot P_i &= 0.
\end{aligned} \tag{26}$$

Rearranging the terms you have

$$\begin{pmatrix}P_1^T & 0^T & -u_1P_1^T\\ 0^T & P_1^T & -v_1P_1^T\end{pmatrix}\cdot\begin{pmatrix}m_1\\ m_2\\ m_3\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}. \tag{27}$$

For n points we have a big $2n\times 12$ matrix Q.
The problem hence reads

$$Q\cdot M = 0, \tag{28}$$

where Q is known and M is unknown.

Minimal Solution:

- Rank 11 to have unique non-trivial (up to scale) solution M (Q known!).
- Each 3D/2D correspondence provides 2 independent equations.
- $5 + \frac{1}{2}$ correspondences are needed (in fact 6).

Overdetermined Solution:

- More than 6 points.
- Minimize $||QM||^2$ with the constraint $||M|| = 1 \rightarrow$ SVD. The solution is the eigenvector corresponding to the smallest eigenvalue of Q^TQ. That's because this is the unit vector x that minimizes $||Qx||^2 = x^TQ^TQx$. This can be done in matlab with
```
[U,S,V] = svd(Q);
 M = V(:,12);
```

Degenerated Configurations:

- Points lying on a plane and or along a line passing through the projection center.
- Camera and points on a twisted cubic (degree 3).

Once we have M, we know from its definition

$$M = K(R|T). \tag{29}$$

Remark.

- We are not enforcing orthogonality of R.
- QR factorization of M, whith R (orth.) and T (upper triangular matrix).
- Or: Using SVD and enforcing that every eigenvalue is 1.
- We can estimate the scale factor (M comes with a scale factor) by dividing the Frobenius norm of the computed matrix $\tilde{R}$ with the one of the current estimate.

DLT vs. PnP

- If the camera is calibrated, only R and T need to be determined. PnP leads to smaller error regarding calibration using same number of points to estimate pose.
- PnP's computation time is quite constant with increasing number of points, while DLT's computation time increases with the increasing number of points.

8. (a) *Omnidirectional cameras (only definition of central and non central cameras):*

Answer. A vision system is said to be central when the optical rays of the viewed objects intersect in a single point in 3D called projection center or **single effective viewpoint**. For hyperbolic and elliptical mirrors, the single viewpoint property is achieved by ensuring that the camera center coincides with one of the foci of the hyperbola (ellipse), as reported in Figure 11. The camera is **non-central**, if the rays intersect in various points.

(b) *What type of mirror ensure central projection?*

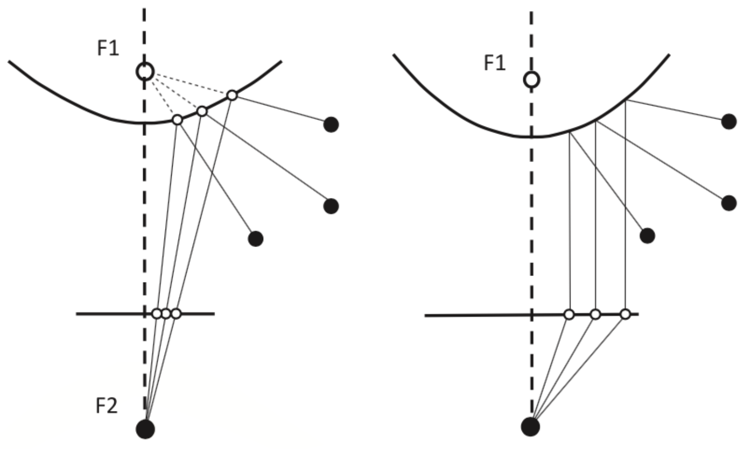


Fig. 3. Central catadioptric cameras can be built by using hyperbolic and parabolic mirrors. The parabolic mirror requires the use of an orthographic lens.

Figure 11: Central Catadioptric cameras

Answer. Class of rotated (swept) conic shapes: hyperbolical, parabolical, elliptical mirrors ensure central projection.

(c) *Spherical model: illustrate equivalence between perspective and omnidirectional model.*

Answer. With their landmark paper from 2000, Geyer and Daniilidis showed that every catadioptric (parabolic, hyperbolic, elliptical) and standard perspective projection is equivalent to a projective mapping from a sphere, centered in the single viewpoint, to a plane with the projection center on the perpendicular to the plane and distant ε from the center of the sphere, as you can see in Figure 12. Let $P = (x, y, z)$ be a scene point in the mirror reference frame centered in C. We assume that the axis of symmetry of the mirror is perfectly aligned with the optical axis of the camera. We also assume that the x and y axes of the camera and mirror are aligned. Therefore, the camera and mirror frames differ only by a translation along z.

i. Projecting the image point $P = (x, y, z)$ from the **mirror reference frame** to the **unit sphere**

$$P_s = \frac{P}{\|P\|} = (x_s, y_s, z_s) \tag{30}$$

ii. The point coordinatess are converted to the coordinates of the **new** reference frame centered in $C_\varepsilon = (0, 0, -\varepsilon) : P_\varepsilon = (x_s, y_s, z_s + \varepsilon)$. We can observe that ε ranges between 0 (*planar* mirror) and 1 (*parabolic mirror*). Its exact value can be obtained by knowing the distance d between the foci and the latus rectum l.

iii. Project the point P_ε onto the **normalized image plane** distant 1 from C_ε

$$\tilde{m} = (x_m, y_m, 1) = \left(\frac{x_s}{z_s + \varepsilon}, \frac{y_s}{z_s + \varepsilon}, 1\right) = g^{-1}(P_s) \tag{31}$$

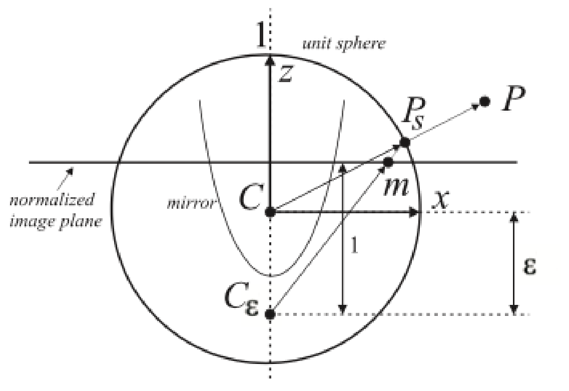


Fig. 5. Unified projection model for central catadioptric cameras of Geyer and Daniilidis.

Figure 12: Spherical Model

iv. Point $\tilde{m}$ is mapped to the **camera** image point $\tilde{p} = (u, v, 1)$ through K $\rightarrow \tilde{p} = K \cdot \tilde{m}$, where K is given by

$$K = \begin{pmatrix} \alpha_u & \alpha_u \cot(\theta) & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{32}$$

By inverting g given by Eq. 31, we can project points from the normalized image plane back to the **unit sphere**

$$P_s = g(m) \sim \begin{bmatrix} x_m \\ y_m \\ 1 - \varepsilon \frac{x_m^2 + y_m^2 + 1}{\varepsilon + \sqrt{1 + (1-\varepsilon^2)(x_m^2 + y_m^2)}} \end{bmatrix} \tag{33}$$

Note that the last row of P_s is obtained by imposing that P_s lies on the sphere, i.e. $x_s + y_s + z_s = 1$

Eq. 33 is the core of the projection model of **central catadioptric** camera. In case of a planar mirror, ε center of a sphere = 0, Eq. 33 is the same projection equation of a **perspective** camera $P_s \sim (x_m, y_m, 1)$.

(d) *What do we mean by normalized image coordinates on the unit sphere?*

Answer. Projecting the image point $P = (x, y, z)$ from the **mirror reference frame** to the **unit sphere** using Eq. 30.

9. *Given an image and the associated camera pose, how would you superimpose a virtual object on the image (for example, a virtual cube). Describe the steps involved.*

Answer. For undistorted images:

(a) Convert the image to grayscale

(b) Create a matrix containing all the 3D positions of the checkerboard corners (meshgrid).

(c) Write a function to project the corners on the image plane (maps points from the world to the camera frame).

- From matrix K one can recover the rotation matrix R with the Rodriguez formula
$$R = \mathbb{I} + \sin(\theta)[k]_\times + (1 - \cos(\theta))[k]_\times^2. \tag{34}$$
- The camera poses are the vector t.
- Find the normalized coordinates (divide by third component of the vector).
- Convert to pixel coordinates applying K, i.e. $K \cdot x_{\text{dist}}$.

(d) Undistort images with bilinear transformation. :

- Artifacts due to the facts that some coordinates are non-integer. This is usually solved by backward warping: warping pixel locations from destination image (undistorted) to source image (distorted:
$$I_u(u, v) = I_d(\Gamma(u, v)). \tag{35}$$
To deal with non-integers, we use nearest neighbor interpolation (closest integer). Short explanation:
When an image needs to be scaled up, each pixel of the original image needs to be moved in a certain direction based on the scale constant. However, when scaling up an image by a non-integral scale factor, there are pixels (i.e., holes) that are not assigned appropriate pixel values. In this case, those holes should be assigned appropriate RGB or grayscale values so that the output image does not have non-valued pixels. Bilinear interpolation can be used where perfect image transformation with pixel matching is impossible, so that one can calculate and assign appropriate intensity values to pixels. Unlike other interpolation techniques such as nearest-neighbor interpolation and bicubic interpolation, bilinear interpolation uses values of only the 4 nearest pixels, located in diagonal directions from a given pixel, in order to find the appropriate color intensity values of that pixel. Bilinear interpolation considers the closest 2 ×2 neighborhood of known pixel values surrounding the unknown pixel's computed location. It then takes a weighted average of these 4 pixels to arrive at its final, interpolated value

(e) Define the world points of the cube vertex and then project them using the same transform as before.

Lecture 04

Filtering

1. *Convolution vs correlation*

   Answer. Correlation is a metric for similarity between two different signals. Convolutions applies one signal to the other.
   Convolution:One of the sequences is flipped before sliding over the other. The resulting sequence expresses the amount of overlap of one sequence when it is shifted over another sequence. Linearity, associativity, commutativity. Notation $f * g$. In 1D:

   $$f * g = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\mathrm{d}\tau. \tag{36}$$

   Here g is reversed and shifted over f.
   In 2D:

   $$\begin{aligned} G[i,j] &= H * F \\ &= \sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v]F[i-u,j-v]. \end{aligned} \tag{37}$$

   Flip the filter in both dimensions, then slide the filter over the image. In other words: replacing each pixel with a linear combination of its neighbors. The filter H is also called kernel or mask. One can change the weights.
   Correlation: no flipping before shifting and displays the measure of similarity between the two sequence, sliding dot product, used in pattern recognition

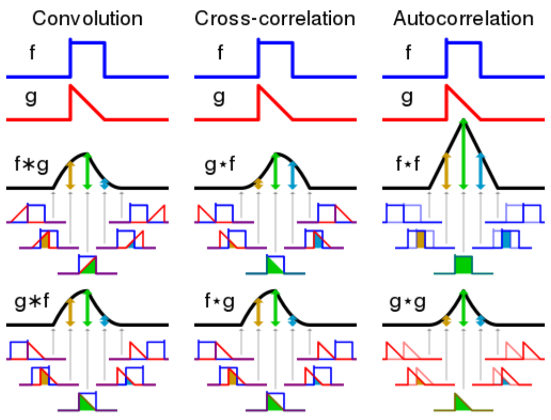


Figure 13: Convolution. Cross-correlation. Autocorrelation

2. *Box filter vs Gaussian filter (what are the pros and cons of either one?)*

   Answer. Box Filter: spatial domain linear filter in which each pixel in the resulting image has a value equal to the average value of its neighboring pixels in the input image.

   (a) low pass filter (blur)

(b) mask with positive entries that sum to 1

(c) all weights are equal

(d) size : $m \times m$

(e) +: faster than sliding window algorithm

(f) −: aliasing, blurs out details stronger than Gaussian Filter

Gaussian Filter: What if we want the **closest** pixels to have a higher influence on the output? More weight on the central pixels, less to the neighbors. (non-uniform low pass filter)

$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}. \tag{38}$$

- +: fastest filter from the 3
- −: blurs edges and reduces contrast

What parameter matter?

- Size of the kernel. The Gaussian has generally infinite support but discrete filters use finite kernels.
- The **variance** σ^2 of Gaussian: determines extent of smoothing (larger variance, larger smoothing).

3. *Gaussian filters: why should we increase the size of the kernel if sigma is large (i.e. sigma close to the size of the filter kernel?)*

   Answer. Maintain the Gaussian nature of the filter. If σ too close to size of the kernel, the coefficient at the edge of the mask is not close to 0. $\rightarrow$ abrupt change, not rotational symmetric .

4. *Median filter (when do we need a median filter?)*

   Answer. Removes spikes: good for impulse and salt and pepper noise (linear smoothing filters do not alleviate that, random high values create high value patches). Nonlinear filter, which computes the median value and replaces the high value with that.

   - +: Preserves sharp transitions/ edges.
   - −: Removes small brightness variations.

5. *Boundary issues*

   Answer. The filter window falls off the edge of the image. We need to pad the image borders with

   - Zero padding (black)
   - Wrap around
   - Copy edge
   - Reflect across edge

Edge Detection

1. *Working principle with 1D signal*

   Answer. Edges are nothing else than sharp intensity changes. The intensity function along a horizontal scan line $f(x)$ has low values for dark parts. Taking the first derivative of the intensity function, edges correspond to **extrema of derivative**.

2. *Noise effects*

   Answer. Considering only a single row or column of the image, if the image has a lot of noise, the first derivative is even noisier. Hence, we cannot see the edge in the change of intensity. In practice, differenciating a noisy signal leads to too many peaks. Solution: first smooth the image (convolution with h), then differentiate.

3. *Differential property of convolution*

   Answer.
   $$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x} h) \star f \tag{39}$$
   First derive the filter $h(x)$, then convolution. Smoothed and derived to detect the noisy intensity function f.

4. *How do we compute the first derivative along x and y?*

   Answer. For discrete data, it holds
   $$\frac{\mathrm{d}f(x,y)}{\mathrm{d}x} \approx \frac{f(x+1,y)-f(x,y)}{1}. \tag{40}$$
   Partial derivatives of an image are in x (-1,1), in y $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$. The **Gradient** of an image if given by
   $$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{pmatrix} \tag{41}$$
   The **gradient direction** is given as
   $$\Theta = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right). \tag{42}$$
   The **edge strength** is given by
   $$||\nabla f|| = \sqrt{(\frac{\partial f}{\partial y})^2 + (\frac{\partial f}{\partial y})^2}. \tag{43}$$
   First smoothing with a Gaussian (or other filter), then taking directional derivative **equals** convolving image with a Sobel filter = *directional/ oriented* filter or first take directional derivative of the Gaussian filter,the convolve the image with the derivative.

5. *Laplacian of Gaussian operator: why should we use it and what effect does it have on the image?*

Answer. This operator is separable. It can be split into directions. Consider

$$\frac{\partial^2}{\partial x^2}(h \star f) \tag{44}$$

From above we know the Gaussian filter smooths the image and then taking the derivatives **equals** convolving the image directly with the *Laplacian of Gaussian* (LoG) filter. Taking the second derivative of the filter, it acts as a *band-pass filter* which filters out both low and high frequencies.

6. *Properties of smoothing and derivative filters (what is the sum of the coefficients of a smoothing filter, and of a derivative filter?)*

   Answer. Smoothing Filter

   - Has positive values
   - Sums to 1$\rightarrow$ preserves brightness of constant regions
   - Removes high frequency components.

   Derivative Filter

   - Has opposite signs, used to get high response in regions of high constrast.
   - Sums to 0 $\rightarrow$ no response in constant regions. If not, uniform zones would have a gradient in spatial domain, which is not what we want.
   - Highlights high frequency components.

7. *Illustrate Canny edge detection.*

   Answer. The process of Canny edge detection algorithm

   (a) Convert colour image to grayscale by replacing each pixel by the mean value of its RGB components
   (b) Apply Gaussian filter to smooth the image in order to remove the noise
   (c) Compute the gradient of the smoothed image in both directions. (or replace step 2. and 3. by convolving the image with x and y derivatives of Gaussian filters)
   (d) Discard pixels whose gradient magnitude is below a certain threshold.
   (e) Apply non-maximal suppression to get rid of spurious response to edge detection

   The *scale parameter* σ is selected based on

   - the desired level of detail: fine edges vs global edges;
   - the noise level;
   - the localisation-detection trade off: see template matching.

8. *What is non-maxima suppression and how is it implemented?*

Answer. The local maxima is tracked along gradient direction. High intensity means high probability of the presence of an edge: this is not enough. Only local maxima can be considered as part of an edge. A local maxima can be found where the gradient derivative is 0. Due to noise and the multiple response, the filtered image may contain wide ridges around the local maxima.

(a) From each position (x, y), step in the two directions perpendicular to edge orientation θ(x, y).

(b) Denote the initial pixel (x, y) by C, the two neighboring pixels in the perpendicular directions by A and B.

(c) If the M(A) $>$ M(C) or M(B) $>$ M(C), discard the pixel (x, y) by setting M(x, y) $= 0$.

The non-maxima suppression sets all pixels to zero that are not actually local maxima $\rightarrow$ a **thin line** in the output.

Lecture 05/06

Point feature detection

1. *What is template matching and how is it implemented? (Mathematical expression)*

   Answer. Find location in an image that are similar to a **template**. If we look at filters as templates, we can use **correlation** to detect these locations. In the resulting correlation map, the brighter the spots the more the template correlates to something on the image.
   We consider images H and F as vectors and express the correlation between them as

   $$\langle H, F\rangle = ||H|| \cdot ||F|| \cdot \cos(\theta). \tag{45}$$

   If we use **Normalized Cross Correlation** (NCC) (highest complexity), we consider the unit vectors of H and F, hence we measure their similarity based on the angle θ. For identical vectors one gets $NCC = 1$.

   $$\begin{aligned}\cos(\theta) &= \frac{\langle H, F\rangle}{||H|| \cdot ||F||}\\ &= \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)F(u,v)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)^2} + \sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F(u,v)^2}} > 0.8.\end{aligned} \tag{46}$$

2. *What are the limitations of template matching? Can I use it to recognize any car?*

   Answer. Template Matching will only work if **scale, orientation, illumination** and in general the appearance of the template and the object to detect are very similar. Hence, this process can only detect cars very similar to the template car, but not generally cars. $\rightarrow$ **can't use to recognize any car**.

3. *Similarity metrics: SSD, SAD, NCC, Census transform. What is the intuitive explanation behind SSD and NCC (hint: represent images as vectors)?*

   Answer.

   - The **normalized cross correlation (NCC)** takes values between -1 and 1, 1 equals identical. NCC is invariant to linear intensity changes! It holds

   $$\begin{aligned}NCC &= \frac{\langle H, F\rangle}{||H|| \cdot ||F||}\\ &= \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)F(u,v)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)^2} + \sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F(u,v)^2}}.\end{aligned} \tag{47}$$

   - Other methods are the **Sum of Absolute Differences (SAD) (simplest)**

   $$SAD = \sum_{u=-k}^{k}\sum_{v=-k}^{k} |H(u,v) - F(u,v)|, \tag{48}$$

   - the **Sum of Squared Differences (SSD) (high computational complexity)**

   $$SSD = \sum_{u=-k}^{k}\sum_{v=-k}^{k} (H(u,v) - F(u,v))^2. \tag{49}$$

- **Census Transfrom**: It maps an image patch to a bit string. The general rule is that **if a pixel is greater than the center pixel** its corresping bit is set to 1, else to 0. For a $n \times n$ window the string will be $n^2 - 1$ bits long. The 2 bit strings are compared using the **Hamming distance** (if bigger than previous 1, else 0, starting from right). The **Advantages** are
  - More **robust to object background** problem
  - No square roots or divisions are required. Efficient!
  - Intensities are considered relative to the center pixel of the patch making it **invariant to monotonic intensity** changes.

SAD and SSD are **not** invariant to linear illumination changes. To cope with the difference, the mean value of each image can be subtracted: Zero-mean Normalized Cross Correlation is invariant to linear illumination change intensity.
The similarity is computed between the gray intensity levels of the two vectors. If I_1 and I_2 are perfect matches, the resultant SAD and SDD will be 0. Intuition is missing!

4. *Feature extraction: what are good feature to track: definition of corners and blobs and their pros and cons.*

Answer. Good features are always **perceivable** and easily detectable from the environment. **Corners, blobs**, lines and points are *low-level features (geometric primitives)*. Better for detection because they

- can be abstracted from raw date $\rightarrow$ high conservation of information
- provide lower volume of data while increasing the distinctiveness of each feature

Corners

- Intersection of one or more edges, image gradient has **two or more** dominant directions
- Repeatable and distinctive
- Examples of corner detectors are Harris, Shi-Tomasi, SUSAN, FAST.
- $+$: high localization accuracy $\rightarrow$ good for VO
- $-$: less distinctive than a blob
- Change of angle between images should be little

Blob:

- Any other image pattern which is not a corner, that differs significantly from its neighbors in intensity and texture.
- Examples of blob detectors are MSER, LOG, DOG, SIFT, SURF, CenSurE.
- $+$: for place recognition because more distinctive than a corner
- $-$: Localization accuracy
- Center might shift due to shear

5. *Harris corner detector*:

(a) *Intuitive illustration using Moravec definition of corner, flat region, and edge*

Answer. Shifting a window in **any direction** should give a **large change** in intensity (e.g. in SDD) in a least 2 directions

- **Flat** region: no intensity change! (SSD$\approx$ 0 in all directions).
- **Edge**: no change along the edge direction (SSD $\approx$0 along adge but >> 0 in other directions).
- **Corner:** significant change in at least two directions (SSD >> 0 in at least 2 directions).

(b) *Show how to get to the second moment matrix from the definition of SSD and first order approximation (show that this is a quadratic expression) and what is the intuitive interpretation of the second moment matrix using ellipse (what does the ellipse represent?).*

Answer. Let I be a gray scale image. We consider the reference patch centered at (x, y) and the shifted window centered at $(x + \Delta x, y + \Delta y)$. The patch has size P. We compute

$$SSD(\Delta x, \Delta y) = \sum_{x,y \in P} (I(x, y) - I(x + \Delta x, y + \Delta y))^2 . \tag{50}$$

We define

$$I_x = \frac{\partial I(x, y)}{\partial x}, \quad I_y = \frac{\partial I(x, y)}{\partial y}, \tag{51}$$

and approximate with first order Taylor expansion:

$$\begin{aligned} I(x + \Delta x, y + \Delta y) &\approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y \\ \Rightarrow SSD(\Delta x, \Delta y) &\approx \sum_{x,y \in P} (I_x(x, y)\Delta x + I_y(x, y)\Delta y)^2 \end{aligned} \tag{52}$$

Simple quadratic function in the deltas! We can write this in matrix form:

$$SSD(\Delta x, \Delta y) \approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \cdot \underbrace{\sum_{x,y \in P} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}}_{M} \cdot \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}, \tag{53}$$

where M is the second moment matrix. The elements from the matrix are **pixel-wise products**!

(c) *What is the M matrix for an edge, for a flat region, for an axis-aligned 90-degree corner, and for a non-axis- aligned 90-degree corner?*

Answer.

- Edge along x: $M = \begin{pmatrix} 0 & 0 \\ 0 & \lambda_2 \end{pmatrix}$.
- Flat region: $M = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
- Aligned corner: $M = \begin{pmatrix} \cos(45) & -\sin(45) \\ \sin(45) & \cos(45) \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \cdot \begin{pmatrix} \cos(45) & \sin(45) \\ -\sin(45) & \cos(45) \end{pmatrix}$

- Non-axis aligned corner: M is symmetric, decompose it to $M = R^{-1}\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}R$ where the angle $\neq \pi/4$ in the rotation matrix R

(d) *What do the eigenvalues of M reveal?*

Answer. The Eigenvaluess give information about, if the image patch is an edge or not. The two Eigenvectors identify the directions of largest and smallest changes of SSD.

Claim 2. One can visualize this as an ellipse with axis lengths determined by eigenvalues $(1/\sqrt{\lambda_{\text{max,min}}})$ and two axes determined by the eigenvectors of M (columns of R).

Proof. Since M is symmetric, we can always come to its eigenvalue decomposition. Let's consider

$$M = \begin{pmatrix}v_1 & v_2\end{pmatrix}\cdot\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}\cdot\begin{pmatrix}v_1^T\\ v_2^T\end{pmatrix} = 1 \tag{54}$$

Then, using the quadratic form one gets

$$\begin{aligned} x^T\cdot\begin{pmatrix}v_1 & v_2\end{pmatrix}\cdot\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}\cdot\begin{pmatrix}v_1^T\\ v_2^T\end{pmatrix}\cdot x &= 1\\ \lambda_1 x^T v_1 v_1^T x + \lambda_2 x^T v_2 v_2^T x &= 1\\ \lambda_1 (v_1^T x)^T (v_1^T x) + \lambda_2 (v_2^T x)^T (v_2^T x) &= 1\\ \frac{(v_1^T x)^2}{\left(\frac{1}{\sqrt{\lambda_1}}\right)^2} + \frac{(v_2^T x)^2}{\left(\frac{1}{\sqrt{\lambda_2}}\right)^2} &= 1, \end{aligned} \tag{55}$$

from which is clear that the eigenvectors v_1, v_2 represent the axis directions of the ellipse and $\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}$ their length. □

Large ellipses/circles denote flat region or edges, small ones a corner!
A corner can then identified by checking whether the minimum of the two eigenvalues of M is larger than a certain user-defined threshold. Mathematically, this is the **Shi-Tomasi corner detector**

$$R = \min(\lambda_1, \lambda_2) > \text{threshold}. \tag{56}$$

Harris corner detector

$$R = \lambda_1\cdot\lambda_2 - k(\lambda_1+\lambda_2)^2 = det(M) - k\cdot\text{trace}^2(M), \quad k\in(0.04, 0.15) \tag{57}$$

- Corner: $\lambda_{1,2}$ are large, $R >$ threshold, SSD increases in each direction.
- Edges: $\lambda_1 >> \lambda_2$ or vice-versa.
- Both small: flat region.

(e) *Harris detection vs Shi-Tomasi detection.*

Answer. Harris: approximation for the cornerness function

- +: computationally cheaper, most stable corner dtector

- $-$: sometimes fails to detect corners

Shi-Thomasi: only uses the Eigenvalues of M

- $+$: can detect corners even when Harris detection fails
- $-$: computationally expensive

(f) *Is Harris rotation, illumination and scale invariant? Why?*

Answer.

- Corner response R is **invariant** to image **rotation** because shape (i.e. eigenvalues) remains the same
- **Not** invariant against **scale** changes: refer to Figure 14.

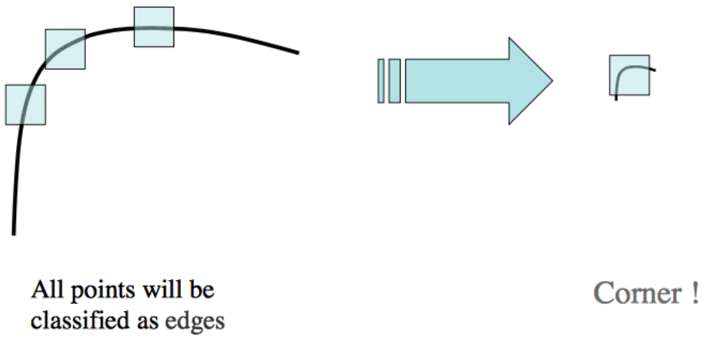


Figure 14: Harris is not scale invariant.

- Invariant to **affine intensity changes**: eigenvalues are scaled by a constant factor but the position of the maxima remains the same.

(g) *What is the repeatability of the Harris detector after a rescaling of 2?*

Answer. Repeatability$= \frac{\#\text{correspondences detected}}{\#\text{correspondences present}}$. 18 % of the possible correspondences are redetected

6. *Scale-invariant detection:*

(a) *How does automatic scale selection work?*

Answer. A possible solution is to rescale the patch (bring it to the canonical scale). A problem with that, is that you have to do it **individually** for all patches in one image (complexity $(N \cdot M)^2$). $\rightarrow$ We take a local maximum of the function: the region size for which the maximum is achieved, should be invariant to image scale. **This scale invariant region size is found in each image independently**.

i. Design a function on the image patch, which is scale invariant, i.e., which has the same value for corresponding regions, even if they are at different scales.

ii. For a point in one image, we can consider the average intensity as a function of region size (x-axis).

iii. By comparing the intensity of the **same feature** in a different scaled image, we can retrieve the scale of the images when the intensity are the **same**.

iv. When the right scale is found, the patch must be **normalized**.

(b) *What are the good and the bad properties that a function for automatic scale selection should have or not have?*

Answer.

- Good function: single and sharp peaks! (LoG)
- Bad function: flat peak or multiple peaks → assign **more** region sizes to have a unique feature. Blobs and corners are the ideal locations!

(c) *How can we implement scale invariant detection efficiently? (show that we can do this by resampling the image vs rescaling the kernel).*

Answer. Rescaling the Kernel: Convolve image with kernel to identify sharp discontinuities:

$$f = \text{Kernel} * \text{Image} \tag{58}$$

It has been shown that the Laplacian of Gaussian kernel is optimal under certain assumptions

$$\text{LoF} = \nabla^2 G(x,y) = \frac{\partial G(x,y)}{\partial x^2} + \frac{\partial G(x,y)}{\partial y^2}, \tag{59}$$

Then, the **correct scale** is found as local maxima or minima across consecutive smoothed images. This should be done for **several** region sizes.
Resampling the Image: an efficient implementation of multi-scale detection uses the method **scale-space pyramid**: instead of varying the window size of the feature detector, the idea is to generate upsampled (enlarge the image, interpolating) or downsampled versions of the **same image**. Then, the **correct scale** is found as local maxima or minima across consecutive smoothed images.

(d) *What is the Harris Laplacian and what is its repeatability after a rescaling of 2?*

Answer. The **Harris-Laplacian** is a multiscale Harris detector and has a much higher repeatability rate across different scales than the normal Harris detector. After rescaling of 2, the repeatability rate is ca. 65 %.

Point feature descriptor and matching

1. *What is a feature descriptor? (patch of intensity value vs histogram of oriented gradients). How do we match descriptors?*

   Answer. A feature descriptor is an algorithm which takes an image and outputs feature descriptors/feature vectors. Feature descriptors encode interesting information into a series of numbers and act as a sort of numerical *fingerprint* that can be used to differentiate one feature from another.

   - **Simplest Descriptor**: **Intensity** values within a squared patch or gradient histogram.

     (a) Start with an *empty* canonical patch (all pixels set to 0).

     (b) For each pixel in the empty patch apply the **warping function** $W(x, y)$ of the corresponding position in the detected image.

(c) Interpolate the intensity values of the 4 closest pixels in the detected image using *nearest neighbor* or *bilinear interpolation*

$$I(x, y) = I(0, 0)\cdot(1-x)\cdot(1-y)+I(0, 1)\cdot(1-x)\cdot y+I(1, 0)\cdot x\cdot(1-y)+I(1, 1)\cdot xy.$$

Using the eigenvalues (size) and the eigenvectors (direction) of the **Harris** detector as **warp information**, slight view-point invariance can be achieved. Cons: if not warped, very small errors in rotation, scale and view-point will affect matching score significantly, **computationally expensive**

- **Census transform** or **Histograms of Oriented Gradients (HOG)**.
  (a) Compute a histogram of orientations of **intensity gradients**
  (b) Peaks in histogram = dominant orientations
  (c) **Keypoint orientation = histogram peak**, if there are multiple candidate peaks, construct a different keypoint for each such orientation
  (d) Rotate patch according to this angle → puts the patches into a canonical orientation

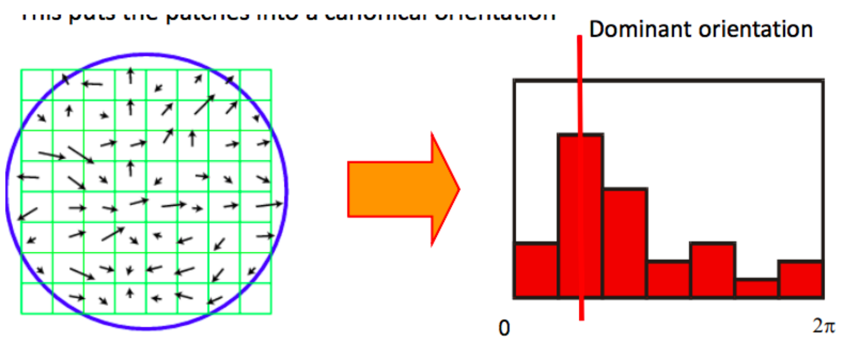


Figure 15: Rotation of a patch. Sum up the magnitude of vectors with the same orientation

The **matching** can be done using **Hamming Distance (Census)** or **(Z)SSD,(Z)SAD, (Z)NCC**.

2. *Scale Invariant Feature Transform detection and descriptor:*

   (a) *How is the keypoint detection done in SIFT and how does this differ from Harris Laplacian?*

   Answer. The SIFT algorithm:

   - Identification of Keypoint **location and scale**.
   - Orientation assignment.
   - Generation of keypoint descriptor.
   - Can handle changes in viewpoint (up to 60 degree out-of-plane rotation).
   - Can handle significant changes in illumintation.
   - Computationally expensive.

   SIFT keypoints are local extrema (maxima and minima) in both **space and scale** of the DoG images:

i. The initial image is **incrementally convolved with Gaussians** $G(k\sigma)$ to produce images separated by a constant factor k in scale space.
   A. The initial Gaussian $G(\sigma)$ has $\sigma = 1.6$.
   B. k is chosen such that $k = 2^{\frac{1}{s}}$, where s in an integer (typically $s = 3$).
   C. For efficiency reasons, when k reaches 2, the image is downsampled by a factor of 2 and then the procedure is repeated up to 4 or 6 octaves (pyramid levels).

   One image blurred with different strong Gaussian filter $\rightarrow$ shows different level of details
ii. Adjacent image scales are then subtracted to produce the difference-of-Gaussian (DoG) images.
   eg Image(kernel=3)-image(kernel=5) $\rightarrow$ fine details as twigs and hair can be seen
   $\uparrow$ kernel $\rightarrow$ larger patches bright and dark spots
iii. Detect maxima and minima of Difference-of-Gaussian in scale space.
iv. Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below.
v. For each max and min found, the output is the location and the scale: this is a candidate keypoint.

Main difference to Harris-Laplacian:**keypoint location** While in Harris, the keypoint is identified in the image plane as local maximum of the corner function, in SIFT the keypoint is a local minimum or maximum of the DoG image in both position **and** scale.

(b) *How does SIFT achieve orientation invariance?*

Answer. The **descriptor** makes the SIFT robust to rotation, small changes of illumination, scale and viewpoint. Each keypoint is assigned to a specific orientation to make it invariant to image rotation.

i. for every pixel around the keypoint, the intensity gradient (mag + orientation) is computed
ii. pixel weighted by mag $\rightarrow$ HOG
iii. orientation corresponding to the highest peak is assigned to the keypoint
iv. all the properties of the keypoint will be measured relative to the keypoint orientation

(c) *How is SIFT descriptor built?*

Answer. Descriptor computation:

i. Divide the patch into 4×4 sub-patches=16 cells.
ii. Compute HOG (8 bins, i.e. 8 directions) for all pixels inside each sub-patch. see Figure 15
iii. Concatenate all HOGs into a single 1D vector. This is the resulting SIFT descriptor: $4 \times 4 \times 8 = 128$ values.
iv. Descriptor matching: SSD (euclidean-distance).

(d) *What is the repeatability of the SIFT detector after a rescaling of 2?*

Answer. The **repeatability** can be expressed as

$$\frac{\text{number of correspondences detected}}{\text{number correspondences present}} \tag{60}$$

The highest repeatability is obtained when sampling 3 scales per octave.
Detected $\sim 90\%$
Correctly matched around 80 %

(e) *And for a 50 degree viewpoint change?*

Answer. **Correctly matched**, location only 73 %, location and orientation 65 %

(f) *Illustrate the 1st to 2nd closest ratio of SIFT detection: whats the intuitive reasoning behind it?*

Answer. While brut-force feature **matching**, issues with closest descriptor can happen. This can give good scores to very ambiguous (bad) matches, due to background clutter or were not detected in the image 1. The better approach is to compute ration of distances to 1st to 2nd closes match:

$$\frac{d(f_1)}{d(f_2)} < \text{Threshold (usually 0.8)}, \tag{61}$$

where

$$\begin{aligned} &d(f_1) \text{ is the distance of the closest neighbor} \\ &d(f_2) \text{ is the distance of the second closest neighbor} \end{aligned} \tag{62}$$

Correct matches need to have the closest neighbor significantly closer than the closest incorrect match, to achieve reliable matching. Moreover, for **false matches**, there will likely be a number of other false matches within similar distances due to the high dimensionality of the feature space. (aka **curse of dimensionality**). We can think of the second closest match as providing an estimate of the density of false matches within this portion of the feature space, and at the same time identifying specific instances of feature ambiguity.

(g) *Where does the 0.8 threshold come from?*

Answer.

- Eliminates 90% of the false matches,
- Discards less than 5% of the correct matches.

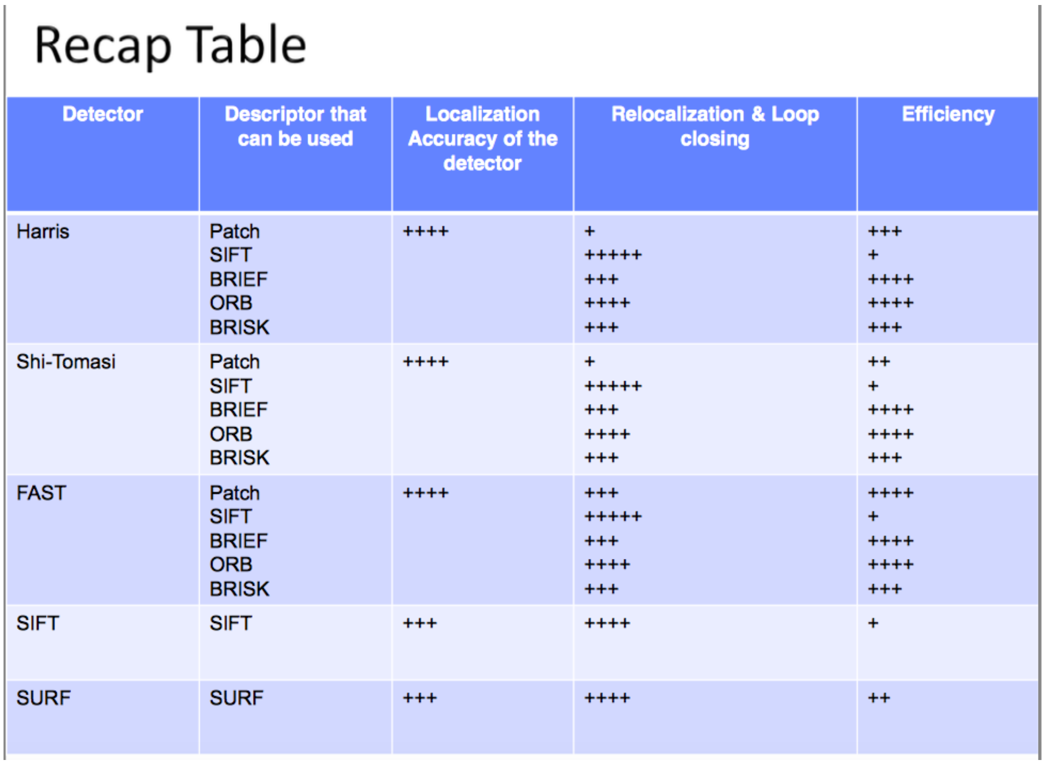
3. *Brief overview of FAST, SURF, BRIEF, ORB and BRISK. Pros and cons of Harris, SIFT, SURF and FAST and BRIEF, ORB and BRISK in terms of localization accuracy, relocalization, and efficiency (see recap table in the slides).*

   Answer. Summary Table 1 and Figure 16.

4. *Describe two different ways of tracking features between frames (hint: exercises and VO project)* ????

Detector/Descriptor	Brief Overview	Pros	cons
Harris detector	corner	rotation invariant	no blob detection, not scale and affine invariant
SIFT both	blob	rotation, scale and affine invariant	no corner detection, inefficient
SURF both	Speeded Up Robust Features. Based on ideas similar to SIFT. blob detector. Approximated computation for detection and descriptor using box filters.	rotation, scale and affine invariant, faster computation, shorter descriptor	no corner detection, inefficient
FAST detector	Feastures from Accelerated Segment Test. Studies intensity of pixels around candidate pixel C. C is FAST corner **if** a set on N contiguous pixels on circle are all brighter than $intensity(C) + threshold$ or all darker than $intensity(C) + threshold$.	very fast detector	no blob detection, not scale and affine invariant
BRIEF descriptor	Binary Robust Independent Elementary Features. For **binary** see summary. The **pattern is generated randomly** (or by ML) only once: then, same pattern is used for all patches.	very fast Hamming distance matching: count the number of bits that are different in the descriptors matched.	not scale or rotation invariant
ORB descriptor	Oriented FAST and Rotated BRIEF. Keypoint detector based on FAST. BRIEF descriptors are steered according to keypoint orientation	binary features are learned by minimizing the correlation on a set of training pathces	
BRISK descriptor	Binary Robust Invariant Scalable Keypoints. Detect corners in scale-space using FAST. compare short and long distance pairs for orientation assignment and descriptor formation	rotation and scale invariant, 10 times faster than SURF	

Table 1: Summary Descriptor Table



Recap Table

Detector	Descriptor that can be used	Localization Accuracy of the detector	Relocalization & Loop closing	Efficiency
Harris	Patch SIFT BRIEF ORB BRISK	++++	+ +++++ +++ ++++ +++	+++ + ++++ ++++ +++
Shi-Tomasi	Patch SIFT BRIEF ORB BRISK	++++	+ +++++ +++ ++++ +++	++ + ++++ ++++ +++
FAST	Patch SIFT BRIEF ORB BRISK	++++	+++ +++++ +++ ++++ +++	++++ + ++++ ++++ +++
SIFT	SIFT	+++	++++	+
SURF	SURF	+++	++++	++

Figure 16: Recap for detectors and descriptors

Answer. Find a set of likely feature locations in a first image and to then search for their corresponding locations in subsequent images.

(a) Detect good features which have high eigenvalues in the auto-correlation matrix to provide stable locations at which to find correspondences. Use Harris corner detector

(b) Assign descriptors to the keypoints using HOG

(c) Or use SIFT detection and descriptor in step 1. and 2.

(d) In subsequent images, search for locations where the corresponding descriptor has low SSD.

Lecture 07 Multiple view geometry 1

1. *SFM vs 3D reconstruction: definition.*

   Answer.

   - **3D reconstruction from multiple views**
     - **Assumption:** K, T, R are **known**.
     - **Goal:** Recover the 3D structure from images.
   - **Structure from motion**
     - **Assumption:** K, T, R are **unknown**.
     - **Goal:** Recover simultaneously 3D scene structure and camera poses (up to scale) from multiple images.

2. *Stereo vision*:

   (a) *Definition of disparity. Simplified case and general case*

   Answer. Disparity means a great difference. Stereopsis is a term that is most often used to refer to the perception of depth and 3-dimensional structure obtained on the basis of visual information of a left and a right image. $\rightarrow$ Stereo Vision
   The basic principle behind stereo vision is **triangulation**.

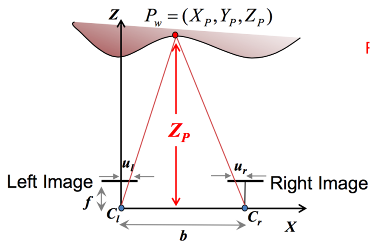
   - Gives reconstruction as intersection of two rays.
   - Requires **camera pose (calibration) and point correspondence** (matching pairing points of the two images which are the projection of the same point in the scene).

   There are basically two cases

   i. Simplified case: identical cameras are **aligned**. The two cameras are identical, meaning that they have the same **focal length**, and are **aligned** with the x-axis.

   ii. General case: different cameras are **not aligned**. Two identical cameras do not exist in nature! Aligning both cameras on a horizontal axis is very hard. in order to use a stereo camera $\rightarrow$ Find Extrinsic (R, t) and Intrinsic $(f, C, radialdistortion)$ parameters. $\rightarrow$ camera calibration (Tsai or Homographies)

   (b) *Mathematical expression of depth as a function of baseline, disparity and focal length?*



Simple Case of Stereo Vision

If we have a world point P_w, a distance from the axis to the point Z_P, a distance between the cameras b, focal length f and pixel coordinates from the corresponding world points u, v, we can use similar triangles from Figure 2b and get

$$\begin{aligned} \frac{f}{Z_P} &= \frac{u_l}{X_P} \\ \frac{f}{Z_P} &= \frac{-u_r}{b - X_P} \end{aligned} \Rightarrow X_P = \frac{u_l \cdot b}{u_l - u_r} \Rightarrow Z_P = \frac{b \cdot f}{u_l - u_r}. \tag{63}$$

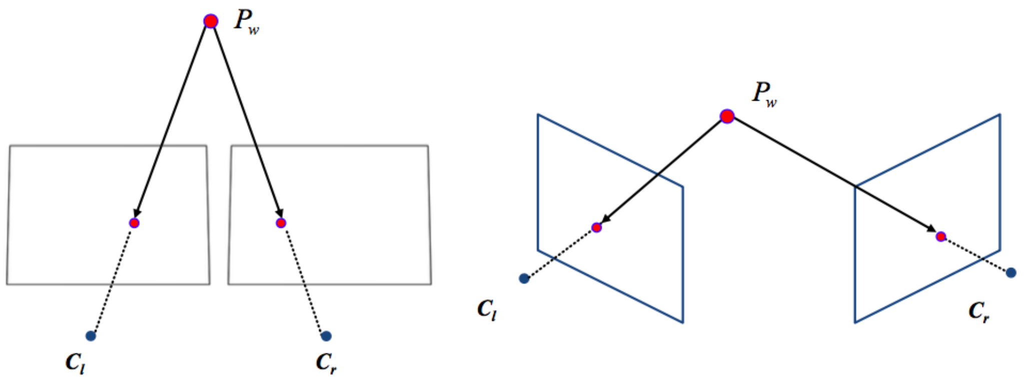


Figure 17: Simplified case and General case

Observation from this equation are

- Distance Z_P is inversely proportional to **disparity**: the distance to near objects can be measured more accurately than that to distant objects.
- **Disparity**: $u_l - u_r$ and proportional to b. For a given disparity error, the accuracy of depth estimate increases with increasing baseline b. increasing disparity with the distance of the objects from the fixation point
- As b is increased, since the distance of the two cameras is increased, some objects may appear in one camera but not in the other (field of view of cameras). These objects won't have disparity.
- If the **baseline** b is unknown it is possible to reconstruct the scene up to a scale (structure from motion!)

(c) *Apply error propagation to derive expression of depth uncertainty. How can we improve the uncertainty?*

Answer.

$$Z_P = \frac{b \cdot f}{u_l - u_r} \rightarrow \sigma_{depth} = \sqrt{(\frac{\partial Z_p}{\partial b})^2\sigma_b^2 + (\frac{\partial Z_p}{\partial f})^2\sigma_f^2 + (\frac{\partial Z_p}{\partial u_l})^2\sigma_{ul}^2 + (\frac{\partial Z_p}{\partial u_r})^2\sigma_{ur}^2} \tag{64}$$

Increase baseline or focus. Decrease depth, u_l or u_r

(d) *Large baseline vs small baseline.*

Answer.

- **Too small:**
  - Large depth error
  - Quantification of error as a function of the disparity?
- **Too large:**
  - Minimum measurable distance increases.
  - Difficult search problem for close objects.

(e) *What is the closest depth a stereo camera can measure? Can you derive it mathematically?*

Answer. In a simplified case, u_l = image width/2, u_r = −image widht/2, where the depth $Z_P = \frac{b \cdot f}{\underbrace{u_l - u_r}_{\uparrow}} \quad \rightarrow \downarrow Z_P$

(f) *Stereo vision general case: show mathematically how we can compute the intersection of two lines, both linearly and non linearly.*

Answer. Lines do not always intersect in the 3D space: we want to minimize the error. For the two cameras we have

$$\tilde{p}_l = \lambda_l \cdot \begin{pmatrix} u_l \\ v_l \\ 1 \end{pmatrix} = K_l \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix}, \quad \tilde{p}_r = \lambda_r \cdot \begin{pmatrix} u_r \\ v_r \\ 1 \end{pmatrix} = K_r \cdot R \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + T. \tag{65}$$

Triangulation: The problem of determining the 3D position of a point given a set of corresponding image locations and known camera poses.
In order to triangulate, we use **least-squares** approximation:

$$\begin{aligned} \lambda_1 \cdot \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} &= K \cdot [I|0] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \Rightarrow \lambda_1 p_1 = M_1 \cdot P \qquad \text{left camera.} \\ \lambda_2 \cdot \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} &= K \cdot [R|T] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \Rightarrow \lambda_2 p_2 = M_2 \cdot P \qquad \text{right camera.} \end{aligned} \tag{66}$$

Triangulation: least-squares approximation We solve for P and get a system $A \cdot \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = b$, which cannot be inverted (A is 3×2). We use pseudoinverse approximation (least squares) and get

$$A^T \cdot A \cdot \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = A^T \cdot b \Rightarrow \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = (A^T \cdot A)^{-1} \cdot A^T \cdot b. \tag{67}$$

Remark. This is a problem with 6 equations and 5 unknowns: the 3 element of the coordinates of the world point and the two depth factors λ_1, λ_2.

Triangulation: Nonlinear approach: We want to find P that minimizes the **sum of squared reprojection error**

$$SSRE = d^2(p_1, \pi_1(P)) + d^2(p_2, \pi_2(P)), \tag{68}$$

where

$$d(p_1, \pi_1(P)) = ||p_1 - \pi_1(P)|| \tag{69}$$

is called **reprojection error**. The observed point is p_1, p_2 and the reprojected one is M_1P, M_2P. In practice, this is done by initializing P using linear approach and then minimize SSRE using Gauss-Newton of Levenberg-Marquardt.

(g) *What is the geometric interpretation of the linear and non linear approaches and what error term do they minimize? (write it mathematically).*

Answer. Geometric Interpretation: Given the projections $p_{1,2}$ of a 3D point P in two or more images, we want to find the coordinates of the 3D point by intersecting the two rays corresponding to the projections. We want to find the shortest segment connecting the two viewing rays and let P be the **midpoint** of the segment. The two rays won't meet exactly because of **noise** and **numerical errors**.
Reprojection error

$$d(p_1, \pi_1(P)) = ||p_1 - \pi_1(P)|| \tag{70}$$

(h) *Correspondence problem: epipolar geometry; definition of epipole, epipolar line, and epipolar plane.*

Answer. Given a point p in a first image, where is its corresponding point p' in the right image?
Correspondence Search: Identify image patches in the left and in the right images, corresponding to the same scene structure. Similarity measures:

- (Z)ZNCC
- (Z)SSD
- (Z)SAD
- Census Transform

To make the search less computationally expensive, search in 1D. Potential matches for p **have to lie** on the corresponding epipolar line l'.

- **epipole** is the projection of the optical center on the other camera image.
- **epipolar line** is the projection of the infinite ray $\pi^{-1}(p)$ corresponding to p in the other camera image. Since $\pi(p) = \lambda K$, $\pi(p)^{-1} = \lambda K^{-1}p$. This makes sense: if we observe a projection p_1 in the left camera, this can correspond to each world point lying on the infinite ray (every one of these points would project into p_1). All these points have a different projection on the right camera, which in 2D forms the epipolar line.
- **epipolar plane** is uniquely defined by the two optical centers C_l, C_r and one image point p

(i) *Draw epipolar lines for two converging cameras, for a forward moving camera, for a side-moving camera.*

Answer. Convergence Plane is where the focus of two cameras meet / attention of screen.

- **Two converging cameras:** see Figure 19 tilted. All epipolar lines intersect at the epipole! As the position of the 3D point varies, the epipolar line *rotates* about the baseline.
- Forward moving camera the line are radiating from the epipolar point. in both images the coordinates of the epipolar point is the **same**
- Side-moving camera the line is horizontal and the epipolar point is at infinity

3. *Stereo rectification and mathematical derivation of rectifying homographies.*

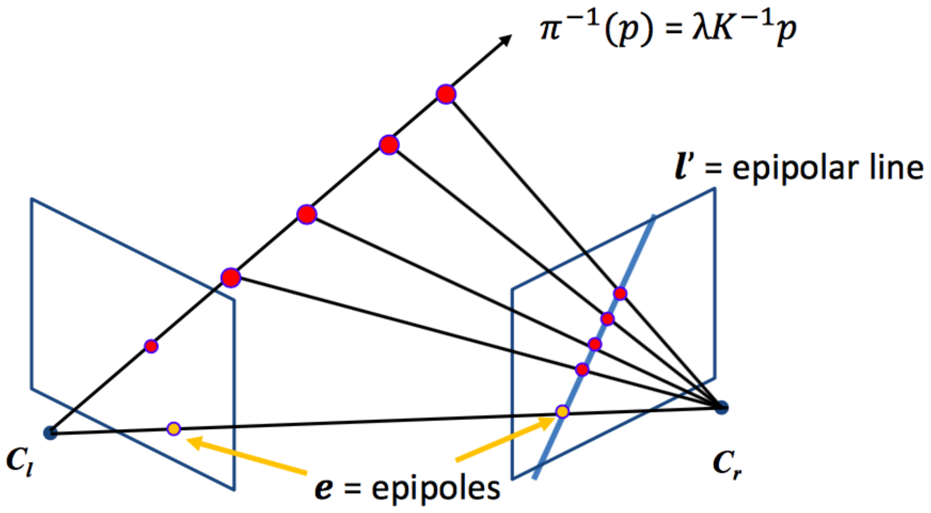


Figure 18: Epipolar lines and epipoles.

Answer. Stereo rectification waprs left and right images into new rectified images, whose epipolar lines are aligned to the baseline.
Goal: make correspondence search simpler and more efficient, because search is done along the horizontal line of the recitified images.

(a) compute homographies for each imput image reprojection
(b) project image planes onto a common plane parallel to the baseline
(c) epipolar lines are horizontal and the scanlines of the left and right image are aligned

Mathematical derivation

(a) The perspective equation for a point in the world is

$$\text{image point} = \tilde{p} = \begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K[R|T] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}. \tag{71}$$

This can be rewritten in a more convenient way by considering $[R|T]$ as the transformation from the world to the Camera frame (T expressed as C):

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot R^{-1} \cdot \left(\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C \right) \tag{72}$$

(b) We can then write the Perspective Equation for the Left and Right cameras: we assume for generality that they have the same intrinsic parameters:

$$\begin{aligned} \lambda_L \cdot \begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} &= K_L \cdot R_L^{-1} \cdot \left(\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_L \right) \quad \text{left camera} \\ \lambda_R \cdot \begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} &= K_R \cdot R_R^{-1} \cdot \left(\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_R \right) \quad \text{right camera} \end{aligned} \tag{73}$$

(c) The goal of stereo rectification is to warp left and right camera images such that their focal planes are **coplanar** and the intrinsic parameters are identical. It follows

$$\lambda_L \cdot \begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} = K_L \cdot R_L^{-1} \cdot \left(\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_L \right) \rightarrow \overline{\lambda}_L \cdot \begin{pmatrix} \overline{u}_L \\ \overline{v}_L \\ 1 \end{pmatrix} = \overline{K} \cdot \overline{R}^{-1} \cdot \left(\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_L \right)$$
$$\lambda_R \cdot \begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} = K_R \cdot R_R^{-1} \cdot \left(\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_R \right) \rightarrow \overline{\lambda}_R \cdot \begin{pmatrix} \overline{u}_R \\ \overline{v}_R \\ 1 \end{pmatrix} = \overline{K} \cdot \overline{R}^{-1} \cdot \left(\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_R \right) \tag{74}$$

(d) By solving for $\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix}$ for each camera, we can compute the Homography (or warping) that needs to be applied to rectify each camera image:

$$\begin{aligned} \overline{\lambda}_L \cdot \begin{pmatrix} \overline{u}_L \\ \overline{v}_L \\ 1 \end{pmatrix} &= \lambda_L \cdot \underbrace{\overline{K} \cdot \overline{R}^{-1} \cdot R_L \cdot K_L^{-1}}_{\text{homography left camera}} \cdot \begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} \\ \overline{\lambda}_R \cdot \begin{pmatrix} \overline{u}_R \\ \overline{v}_R \\ 1 \end{pmatrix} &= \lambda_R \cdot \underbrace{\overline{K} \cdot \overline{R}^{-1} \cdot R_R \cdot K_R^{-1}}_{\text{homography right camera}} \cdot \begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} \end{aligned} \tag{75}$$

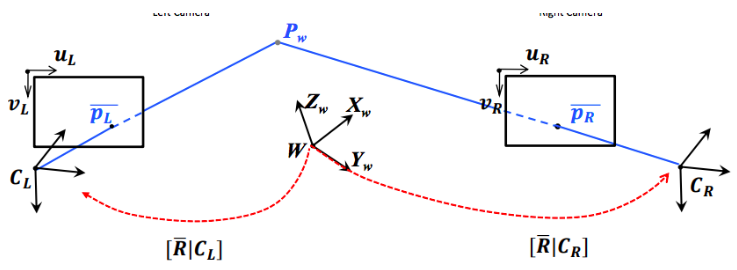
(e) The new $\overline{K}, \overline{R}$ can be chosen as

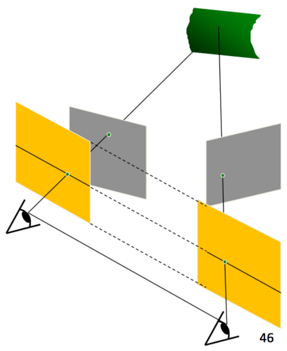
$$\begin{aligned} \overline{K} &= \frac{K_L + K_R}{2} \\ \overline{R} &= [\overline{r}_1, \overline{r}_2, \overline{r}_3], \end{aligned} \tag{76}$$

where $\overline{r}_1, \overline{r}_2, \overline{r}_3$ are the column vectors of $\overline{R}$. These can be computed as

$$\begin{aligned} \overline{r}_1 &= \frac{C_2 - C_1}{||C_2 - C_1||} \\ \overline{r}_2 &= r_3 \times \overline{r_1} \text{ where } r_3 \text{ is the third column of } R_L \\ \overline{r}_3 &= \overline{r}_1 \times \overline{r}_2. \end{aligned} \tag{77}$$

For more details have a look at *A compact alg. for rectification of stereo pairs.*





4. *Disparity map. How is it computed?*

Answer. A disparity map appear as a grayscale image where the intensity of every pixel point is proportional to the disparity of that pixel in the left and right image: objects that are closer to the camera appear lighter, while farther objects appear darker. Input to dense 3D reconstruction

(a) For each pixel in the left image, find its corresponding point in the right image.
(b) Compute the disparity for each pair of correspondences.
(c) Visualized in gray-scale or color coded image.

Close objects experience bigger disparity. They appear brighter in disparity map. The depth Z can be computed from the disparity by recalling that

$$Z_P = \frac{bf}{u_l - u_r} \tag{78}$$

This is really useful for obstacle avoidance. Challenges include: occlusion, repetition, non-lambertian surfaces (specularities), textureless surfaces. This is important for obstacle avoidance.

5. *How to establish stereo correspondences with subpixel accuracy?*

   Answer. From slides and Szeliski book ,

   - To average noise effect, use a window around the point of interest
   - Neighborhoods of corresponding points are similar in intensity patterns(NCC, SSD etc.)
   - Occluded areas can be detected using cross-checking, i.e., comparing left-to-right and right-to-left disparity maps.
   - Associate confidences with per-pixel depth estimates, which can be done by looking at the curvature of the correlation surface, i.e., how strong the minimum in the DSI image is at the winning disparity. $\uparrow$ confidence $\propto$ strong texture

6. *Describe one or more simple ways to reject outliers in stereo correspondences.*

   Answer. A median filter can be applied to clean up spurious mismatches, and holes due to occlusion can be filled by surface fitting or by distributing neighboring disparity estimates

   - Uniqueness: only one match in right image for every point in left image.
   - Ordering: points on same surface will be in same order in both views
   - Disparity gradient: disparity changes smoothly between points on the same surface.

7. *Is Stereo Vision the only way of estimating depth information? If not, list alternative options.*

   Answer.

   - **Graph Cuts:** is formulated in terms of energy minimization. solving a maximum flow problem in a graph
   - **Ground Truth:** information provided by direct observation

Lecture 08/09 Multiple view geometry 2 and 3

1. *Whats the minimum number of correspondences required for calibrated SFM and why?*

   Answer. In *stereo vision*, only **5 degrees of freedom** are measurable. The camera relative pose is unknown: this is e.g. the case when the two images are taken from the same camera but at different times and positions.
   Problem formulation: Given n point correspondences between two images, $\{p_1^i = (u_1^i, v_1^i),\ p_2^i = (u_2^i, v_2^i)\}$, simultaneously estimate the 3D points P^i, the camera relative-motion parameters (R, T), and the camera intrinsics K_1, K_2 that satisfy:

$$\begin{aligned} \lambda_1 \cdot \begin{pmatrix} u_1^i \\ v_1^i \\ 1 \end{pmatrix} &= K_1 \cdot [I|0] \cdot \begin{pmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{pmatrix}, \\ \lambda_2 \cdot \begin{pmatrix} u_2^i \\ v_2^i \\ 1 \end{pmatrix} &= K_2 \cdot [R|T] \cdot \begin{pmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{pmatrix} \end{aligned} \tag{79}$$

   We have two cases then:

   Calibrated Cameras (K_1, K_2 known)

   For convenience, we use *normalized image coordinates*

$$\begin{pmatrix} \bar{u} \\ \bar{v} \\ 1 \end{pmatrix} = K^{-1} \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}. \tag{80}$$

   We want to find R, T, P^i which satisfy

$$\begin{aligned} \lambda_1 \cdot \begin{pmatrix} \bar{u}_1^i \\ \bar{v}_1^i \\ 1 \end{pmatrix} &= K_1 \cdot [I|0] \cdot \begin{pmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{pmatrix}, \\ \lambda_2 \cdot \begin{pmatrix} \bar{u}_2^i \\ \bar{v}_2^i \\ 1 \end{pmatrix} &= K_2 \cdot [R|T] \cdot \begin{pmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{pmatrix}. \end{aligned} \tag{81}$$

   Scale Ambiguity: If we rescale the entire scene by a constant factor (i.e. similarity transformation), the projections (in pixels) of the scene points in both images remain the **same** (because the angles remain the same).

   - In *monocular* vision it is **not possible** to recover the absolute scale of the scene.
   - In *stereo vision*, only **5 degrees of freedom** are measurable:

  - 3 parameters to describe the **rotation**.
  - 2 parameters for the **translation up to a scale** (we can only compute the direction of translation but not its length (magnitude)).

How many knowns and unknowns?

- $4n$ knowns: n correspondences, each one (u_1^i, v_1^i) and (u_2^i, v_2^i), $i = 1, \dots, n$.
- $5+3n$ unknowns: 5 for the motion up to a scale (3 rotation and 2 translation) and $3n$ which is the number of coordinates of the n 3D points.

It should hold

$$\begin{aligned} 4n &\geq 5 + 3n \\ \Rightarrow n &\geq 5. \end{aligned} \tag{82}$$

2. *Derive the epipolar constraint.*

   Answer.

$$\bar{p}_1 = \begin{pmatrix} \bar{u}_1 \\ \bar{v}_1 \\ 1 \end{pmatrix}, \quad \bar{p}_2 = \begin{pmatrix} \bar{u}_2 \\ \bar{v}_2 \\ 1 \end{pmatrix} \tag{83}$$

   We can observe that p_1, p_2, T are coplanar:

$$\begin{aligned} p_2^T \cdot n = 0 \quad &\rightarrow \quad p_2^T \cdot (T \times p_1') = 0 \quad \rightarrow \quad p_2^T \cdot (T \times (Rp_1)) = 0 \\ &\rightarrow \quad p_2^T \cdot [T]_x \cdot Rp_1 = 0 \quad \rightarrow \quad p_2^T \cdot E \cdot p_1 = 0 \end{aligned} \tag{84}$$

   is the epipolar constraint, where $E = [T]_x \cdot R$ is the essential matrix.

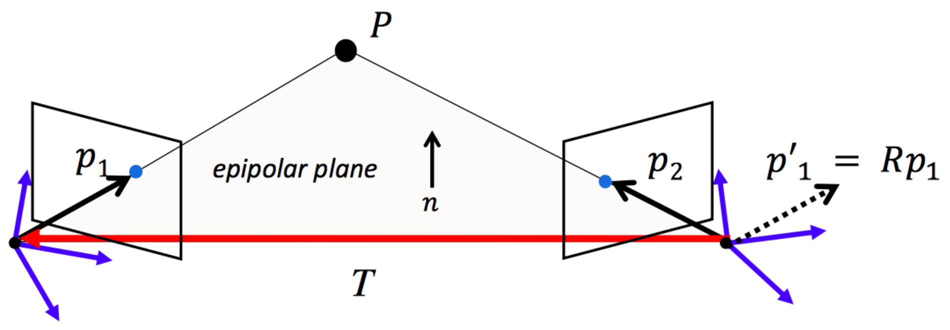


Figure 19: Epipolar Constraint.

3. *Definition of Essential matrix.*

   Answer. Essential matrix **E** is a 3x3 matrix which maps a point p_1 from image 1 onto a line $l_2 = Ep_1$ in image 2 , since $p_2^T l_1 = 0$

$$p_2^T \cdot E \cdot p_1 = 0 \tag{85}$$

   Skew-symmetric matrix of a

$$a \times b = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \cdot \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = [a]_x \cdot b \tag{86}$$

4. *The 8-point algorithm (derivation).*

   Answer. Starting from
   $$p_2^T \cdot E \cdot p_1 = 0 \tag{87}$$
   , each pair of point correspondences provides a linear equation. For n points we can write

   $$\underbrace{\begin{pmatrix} \bar{u}_2^1 \cdot \bar{u}_1^1 & \bar{u}_2^1 \cdot \bar{v}_1^1 & \bar{u}_2^1 & \bar{v}_2^1 \cdot \bar{u}_1^1 & \bar{v}_2^1 \cdot \bar{v}_1^1 & \bar{v}_2^1 & \bar{u}_1^1 & \bar{v}_1^1 & 1 \\ \bar{u}_2^2 \cdot \bar{u}_1^2 & \bar{u}_2^2 \cdot \bar{v}_1^2 & \bar{u}_2^2 & \bar{v}_2^2 \cdot \bar{u}_1^2 & \bar{v}_2^2 \cdot \bar{v}_1^2 & \bar{v}_2^2 & \bar{u}_1^2 & \bar{v}_1^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{u}_2^n \cdot \bar{u}_1^n & \bar{u}_2^n \cdot \bar{v}_1^n & \bar{u}_2^n & \bar{v}_2^n \cdot \bar{u}_1^n & \bar{v}_2^n \cdot \bar{v}_1^n & \bar{v}_2^n & \bar{u}_1^n & \bar{v}_1^n & 1 \end{pmatrix}}_{Q \text{ (known)}} \cdot \underbrace{\begin{pmatrix} e_{11} \\ e_{12} \\ e_{13} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{31} \\ e_{32} \\ e_{33} \end{pmatrix}}_{\bar{E} \text{ unknown}} = 0 \tag{88}$$

   This problem can be written as
   $$Q \cdot \bar{E} = 0. \tag{89}$$

   Two types of solution

   - **Minimal Solution**
     - $Q_{n\times 9}$ should have rank 8 to have unique (up to scale) non trivial solution $\bar{E}$.
     - Each point correspondence provides 1 independent equation.
     - Thus, 8 point correspondences are needed.
   - **Over-determined Solution**
     - $n > 8$ points.
     - A solution is to minimize $||Q \cdot \bar{E}||^2$ subject to the constraint $||\bar{E}||^2 = 1$. The solution is the eigenvector corresponding to the smallest eigenvalue of matrix $Q^T \cdot Q$.
     - This can be solved with Singular Value Decomposition = $[U, E, V]$. Last column of the V $\rightarrow$ E $\rightarrow$ reshape(E)
   - **Degenerate Solution** if 3D points are coplanar. There is the 5 point algorithm which holds also for coplanar points.

5. *How many rotation and translation combinations can the essential be decomposed in?*

   Answer. 4, but only one solution where points are in front of both cameras. A valid R has $det(R) = 1$. Ff not, invert the sign of the matrix.

6. *Geometric interpretation of the epipolar constraint.*

Answer. If the epipolar constraint

$$p_2^T \cdot E \cdot p_1 \neq 0 \tag{90}$$

It means that the projection from p_1 is not *orthogonal* to p_2. Hence,

$$\bar{p}_2^T \cdot E \cdot \bar{p}_1 = ||\bar{p}_2|| \cdot ||E \cdot \bar{p}_1|| \cdot \cos(\theta) \tag{91}$$

which is not zero is p_1, p_2, T are not co-planar and $\theta \neq 90$.

7. *Relation between Essential and Fundamental matrix.*

   Answer. If the cameras are **uncalibrated** $\Rightarrow K_1$, K_2 are unknown. Assumption, Intrinsic parameters are known. It holds

$$\bar{p}_2^T \cdot E \cdot \bar{p}_1 = 0, \tag{92}$$

   where

$$\begin{pmatrix} \bar{u}_1^i \\ \bar{v}_1^i \\ 1 \end{pmatrix} = K_1^{-1} \cdot \begin{pmatrix} u_1^i \\ v_1^i \\ 1 \end{pmatrix}, \quad \begin{pmatrix} \bar{u}_2^i \\ \bar{v}_2^i \\ 1 \end{pmatrix} = K_2^{-1} \cdot \begin{pmatrix} u_2^i \\ v_2^i \\ 1 \end{pmatrix}. \tag{93}$$

   By rewriting the constraint, one obtains

$$\begin{aligned} \begin{pmatrix} u_2^i \\ v_2^i \\ 1 \end{pmatrix}^T \cdot K_2^{-T} \cdot E \cdot K_1^{-1} \cdot \begin{pmatrix} u_1^i \\ v_1^i \\ 1 \end{pmatrix} &= 0 \\ \begin{pmatrix} u_2^i \\ v_2^i \\ 1 \end{pmatrix}^T \cdot F \cdot \begin{pmatrix} u_1^i \\ v_1^i \\ 1 \end{pmatrix} &= 0, \end{aligned} \tag{94}$$

   where F is the **fundamental matrix**, which can be computed as

$$F = K_2^{-T} \cdot E \cdot K_1^{-1} = K_2^{-T} \cdot [T]_x \cdot R \cdot K_1^{-1}. \tag{95}$$

8. *Why is it important to normalize the point coordinates in the 8-point algorithm? Describe one or more possible ways to achieve this normalization.*

   Answer. In case the camera is uncalibrated, matrix Q of equation $Q \cdot F = 0$ contains orders of magnitudes difference between columns. $\rightarrow$ least-squares yields poor results. $\rightarrow$ Poor numerical conditioning, which makes results very sensitive to noise.

9. *Normalized 8-point algorithm.*

   Answer. This estimates the Fundamental matrix on a set of **Normalized correspondences** (with better numerical properties) and then **unnormalizes** the result to obtain the fundamental matrix for the original given correspondences.
   Idea: Transform image coordinates so that they are in the range $[-1, 1] \times [-1, 1]$. One way is to apply the following rescaling and shift A more popular is to rescale the

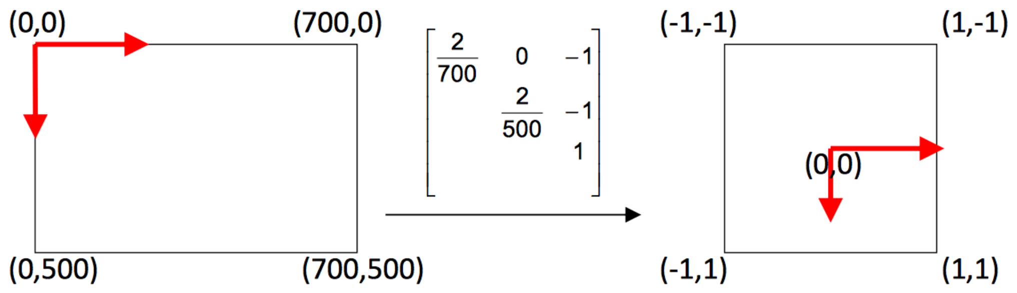

(0,0)
(700,0)
(0,500)
(700,500)
$\begin{bmatrix} \frac{2}{700} & 0 & -1 \\ & \frac{2}{500} & -1 \\ & & 1 \end{bmatrix}$
(-1,-1)
(1,-1)
(0,0)
(-1,1)
(1,1)

Figure 20: Shift for normalized algorithm.

two point sets such that the centroid of each is 0 and the mean standard deviation $\sqrt{2}$. This can be done for every point as follows

$$\hat{p}^i = \frac{\sqrt{2}}{\sigma} \cdot (p^i - \mu), \tag{96}$$

where

$$\mu = \frac{1}{N} \sum_{i=1}^{n} p^i \tag{97}$$

is the centroid of the set and $\sigma = \frac{1}{N} \sum_{i=1}^{n} ||p^i - \mu||^2$ is the mean standard deviation. This transformation can be expressed in matrix form

$$\hat{p}^i = \begin{pmatrix} \frac{\sqrt{2}}{\sigma} & 0 & -\frac{\sqrt{2}}{\sigma}\mu^x \\ 0 & \frac{\sqrt{2}}{\sigma} & -\frac{\sqrt{2}}{\sigma}\mu^y \\ 0 & 0 & 1 \end{pmatrix} \cdot p^i. \tag{98}$$

The algoritm at the end reads

(a) Normalize point correspondences: $\hat{p}_1 = B_1 \cdot p_1$, $\hat{p}_2 = B_2 \cdot p_2$.

(b) Estimate $\hat{F}$ using normalized coordinates $\hat{p}_1, \hat{p}_2$.

(c) Compute F from $\hat{F}$:

$$\begin{aligned} \hat{p}_2^T \cdot \hat{F} \cdot \hat{p}_1 &= 0 \\ p_2^T \cdot B_2^T \cdot \hat{F} \ \cdot B_1 \cdot p_1 &= 0 \\ &\Rightarrow F = B_2^T \cdot \hat{F} \cdot B_1 \end{aligned} \tag{99}$$

A valid fundamental matrix must have # rank = # points and hence det(F)=0

10. *Quality metrics for Fundamental matrix estimation (directional error, epipolar line, and reprojection error).*

Answer. We need to check the numerical results because there are orders of magnitude difference between the columns → least-squares yields poor results
Directional Error Sum of the angular distances to the Epipolar plane: $err = \sum_i (\cos(\theta_i))^2$, where

$$cos(\theta) = \left(\frac{p_2^T \cdot E \cdot p_1}{||p_2^T|| \cdot ||E \cdot p_1||} \right) \tag{100}$$

Epipolar Line Distance Sum of **Squared Epipolar-Line-to-point Distances**

$$err = \sum_{i=1}^{N} d^2(p_1^i, l_1^i) + d^2(p_2^i, l_2^i). \tag{101}$$

Cheaper than reprojection error: does not require point triangulation!

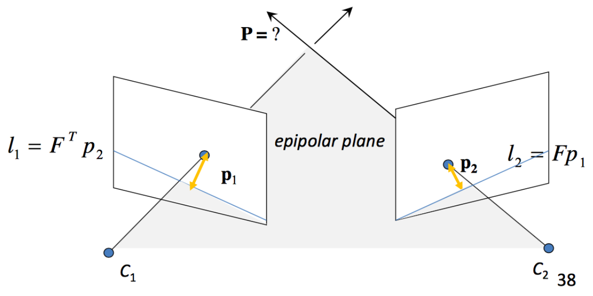


Figure 21: Epipolar Line Distance.

Reprojection Error

Sum of the **Squared Reprojection Errors**

$$err = \sum_{i=1}^{N} ||p_1^i - \pi_1(P^i)||^2 + ||p_2^i - \pi_2(P^i, R, T)||^2 \tag{102}$$

Computation is expensive because of point triangulation, but is the **most accurate**!

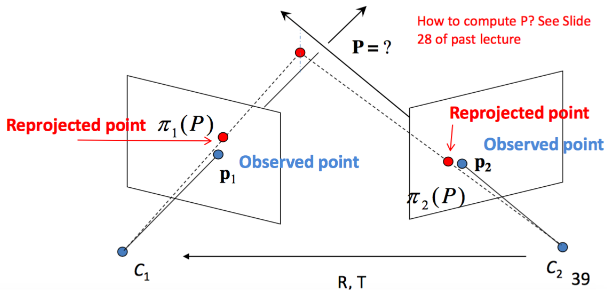


Figure 22: Reprojection Error.

RANSAC

1. *Why do we need RANSAC?*

Answer. Matched points are usually contaminated by **outliers**. Causes for this are

- Change in view point and illumination

- Image noise
- Occlusions
- Blur

The task of removing them is for **Robust Estimation**.

2. *What is the theoretical maximum number of combinations to explore?*

   Answer. Ransac is the standard method for **model fitting in the presence of outliers** (noise points or wrong data). It can be applied to all problems where the goal is to estimate parameters of a model from the data. An easy example is RANSAC for **line fitting:**

   (a) Select sample of 2 points at random.
   (b) Calculate model parameters that fit the data in the sample.
   (c) Calculate error function for each data point.
   (d) Select data that supports current hypothesis.
   (e) Repeat.
   (f) Select the set with the maximum number of inliers obtained within k iterations.

   Theoretical **maximum** number combinations to explore $N(N-1)/2$. Computationally unfeasibe if N is too large!

3. *After how many iterations can RANSAC be stopped to guarantee a given success probability?*

   Answer. Let w be the number of inliers$/N$, N be the total number of data points. We can think of w as

   $$w = P(\text{selecting an inlier-point out of the dataset}). \tag{103}$$

   We assume that the 2 points necessary to estimate a line are selected independently, i.e.

   $$\begin{aligned} w^2 &= P(\text{both selected points are inliers}) \\ 1 - w^2 &= P(\text{at least one of these two points is outlier}) \end{aligned} \tag{104}$$

   Let k indicate the number of RANSAC iterations so far, then

   $$(1-w^2)^k = P(\text{RANSAC never selected two points both inliers}) \tag{105}$$

   Let p be the probability of success:

   $$\begin{aligned} 1 - p &= (1-w^2)^k \\ \Rightarrow k &= \frac{\log(1-p)}{\log(1-w^2)}. \end{aligned} \tag{106}$$

4. *What is the trend of RANSAC iterations k vs the fraction of outliers $\varepsilon = 1 - w$ vs the minimum number of points to estimate the model?*

Answer.

- As observed, k is exponential in the number of points s necessary to estimate the model. We can see that k increases exponentially with the fraction of outliers ε.
- The 8-point algorithm is extremely simple and was very successful; however it requires more than 1177 iterations.
- The 5-point algorithm only requires 145 iterations, but can return up to 10 solutions of E.
- The 2-point algorithm (e.g. line fitting) requires 16 iterations. Motion constraints need to be set to apply this algorithms for robots
- The 1-point algorithm requires only 1 iteration and is **only** used to find the inliers. the motion is then estimated from them in 6DOF and only 1 DOF

5. *How do we apply RANSAC to the 8-point algorithm vs DLT?*

   Answer. DLT 6 points , don't need to since it's monocular and only one image. No outlier rejection for matching required, since there is no matching.
   RANSAC 8 points. Needs outlier rejection, since stereo view $\rightarrow$ two images. The model with eight point algorithm is the matrix E.

6. *How can we reduce the number of RANSAC iterations for the SFM problem (1- and 2-point RANSAC)?*

   Answer. By choosing lower p or another w, we can reduce the iterations.
   Planar Motion (1-,2-point RANSAC) Planar motion is described by three parameters ϑ, φ, ρ

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad T = \begin{pmatrix} \rho\cos(\varphi) \\ \rho\sin(\varphi) \\ 0 \end{pmatrix} \tag{107}$$

Let's compute the Epipolar Geometry

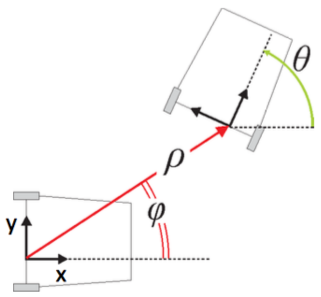


Figure 23: Planar motion.

$$
\begin{aligned}
E &= [T]_x \cdot R \\
&= \begin{pmatrix} 0 & 0 & \rho\sin(\varphi) \\ 0 & 0 & -\rho\cos(\varphi) \\ -\rho\sin(\varphi) & \rho\cos(\varphi) & 0 \end{pmatrix} \cdot \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 0 & 0 & \rho\sin(\varphi) \\ 0 & 0 & -\rho\cos(\varphi) \\ -\rho\sin(\varphi-\theta) & \rho\cos(\varphi-\theta) & 0 \end{pmatrix}.
\end{aligned} \tag{108}
$$

E has 2 DoF (θ, φ), because ρ is the scale factor. Thus, 2 correspondences are sufficient to estimate them.
But: can we use **less** than 2 point correspondences? Yes, if we exploid wheeled vehicles with **non-holonomic** constraints. Wheeled vehicles like cars, follow locally-planar circular motion about the instantaneous Center of Rotation (ICR). Since $\varphi = \theta/2$, meaning that we have only 1 DoF. Only 1 point correspondence is needed. **This is the smallest parametrization possible and results in the most efficient algorithm for removing outliers (Scaramuzza)**. This updates the problem to be

$$
R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad T = \begin{pmatrix} \rho\cos(\frac{\theta}{2}) \\ \rho\sin(\frac{\theta}{2}) \\ 0 \end{pmatrix} \tag{109}
$$

and

$$
\begin{aligned}
E &= [T]_x \cdot R \\
&= \begin{pmatrix} 0 & 0 & \rho\sin(\frac{\theta}{2}) \\ 0 & 0 & -\rho\cos(\frac{\theta}{2}) \\ \rho\sin(\frac{\theta}{2}) & -\rho\cos(\frac{\theta}{2}) & 0 \end{pmatrix}.
\end{aligned} \tag{110}
$$

With the Epipolar Geometry constraint leads to

$$
\theta = -2\tan^{-1}\left(\frac{v_2 - v_1}{u_2 + u_1}\right). \tag{111}
$$

Only one iteration: compute θ for every point correspondence. Up to 1000 Hz, 1-point RANSAC in only used to find the inliers. Motion is then estimated from them in 6DOF.

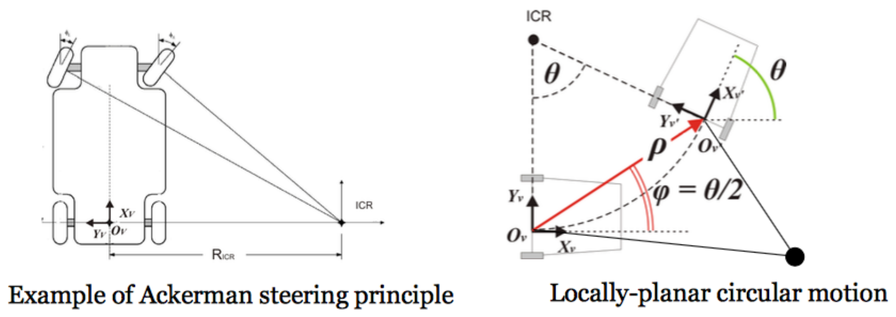


Figure 24: Non-holonomic.

7. *In practice, can you fully rely on the formula that predicts the optimal number of iterations? (hint: especially when the inliers themselves are noisy, RANSAC exercise).*

   Answer.

Bundle Adjustment

1. *Definition of Bundle Adjustment (mathematical expression and illustration).*

   Answer. Nonlinear, simultaneous refinement of structure and motion (i.e. R, T, P^i). It is used after linear estimation of R and T. This, computes R, T, P^i by minimizing the Sum of Squared Reprojection Errors:

$$(R, T, P^i) = \text{argmin}_{R,T,P^i} \sum_{i=1}^{N} ||p_1^i - \pi_1(P^i, C_1)||^2 + ||p_2^i - \pi_2(P^i, C_2)||, \qquad (112)$$

   where C_1, C_2 are the **pose** of the camera in the **world frame**. This can be minimized using *Lavenberg-Marquardt* (more robust than Gauss-Newton to local minima). **It is better to initialize it close to the minimum.** Same for multiple views!

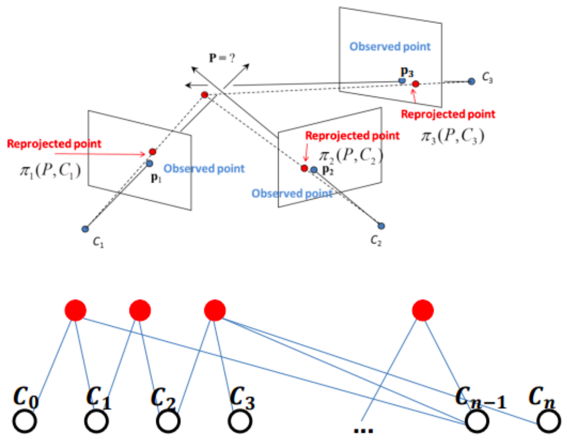


Figure 25: Bundle Adjustment n views.

2. *Hierarchical SFM and sequential SFM for monocular VO.*

   Answer. Hierarchical SFM

   (a) Extract and match features between nearby frames.
   (b) Identify clusters consisting of 3 nearby frames:
   (c) Compute SFM for the 3 frames:
      - Compute SFM between 1 and 2 and build pointcloud.
      - Merge 3rd view running 3-point RANSAC between point cloud and 3rd view.
   (d) Merge clusters pairwise and refine (BA) both structure and motion.

Example is *building Rome in one day.*
Sequential SFM
With n views. Also called Visual Odometry (VO).

(a) **Bootstrapping**

- Initialize structure and motion from 2 views: e.g. 8-point algorithm + RANSAC.
- Refine structure and motion (BA)
- How far should the frames be? If too small baseline, large depth uncertainty. If too large baseline, small depth uncertainty. Remember the picture with ellipses which describe the depth uncertainty. If baseline increases, the ellipses decrease in size.

(b) **Localization**

- Compute camera pose from known 3D-to-2D feature correspondence.
  - Extract correspondences by solving for R and t (K is known).

$$\lambda \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot [R|T] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{113}$$

- What is the minimal number of required point correspondences
  - 6 for linear solution (DLT algorithm).
  - 3 for a non linear solution (P3P algorithm).
  - 3 point RANSAC.

(c) **Extend Structure**

- Extract and triangulate new features from keyframes

By denoting the relative motion between adjacent keyframes as

$$T_k = \begin{pmatrix} R_{k,k-1} & t_{k,k_1} \\ 0 & 1 \end{pmatrix}, \tag{114}$$

we can concatenate transformations to find the full trajectory of the camera as

$$C_k = T_{k,k-1} \cdot C_{k-1} \tag{115}$$

A non-linear refinement (BA) over the last m poses (+visible structure) can be performed to get a more accurate estimate of the local trajectory.

3. *What are keyframes? Why do we need them and how can we select them?*

Answer. A data set may have a lot of images, but we don't need to run SFM algorithms on all images (computationally too expensive) to reconstruct a trajectory. A keyframe is an image for which the pose is estimated. Landmarks are triangulated using 2 keyframes as input.
When frames are taken at nearby positions compared to the scene distance, 3D points will exibit large uncertainty ⇒ One way to avoid this consists of **skipping**

frames until average uncertainty of the 3D points decreases below a certain threshold. The selected frames are called **keyframes**. In general

$$\frac{\text{keyframe distance}}{\text{average-depth}} > \text{threshold } (10-20\%). \tag{116}$$

4. *Definition of loop closure detection (why do we need loops?).*

   Answer.

   - Relocalization problem: during VO, tracking can be lost (due to occlusions, low tecture, quick motion, illumination change).
   - Solution is to re-localize camera pose and continue.
   - Loop closing problem: when go back where you already have been:
     - Loop detection: to avoid map duplication (e.g. same crossing rotated)
     - Loop correction: to compensate the accumulated drift!
   - In both cases places recognition is needed (lecture 12)

5. *List the most popular open source VO and VSLAM algorithms.*

   Answer.

   - PTAM: Parallel Tracking and Mapping for Small AR Workspaces
   - ORB-SLAM: Feature based, FAST corner + Oriented Rotated Brief descriptor includes loop closing, relocalization, final optimization, real time
   - LSD-SLAM: Direct (photometric error) + Semi-Dense formulation, includes loop closing, relocalization, final optimization, real time
   - DSO: Direct (photometric error) + Sparse formulation, real time, BA sliding window
   - SVO: Direct (minimizes photometric error), Feature-based (minimizes reprojection error), mapping, real time fast

6. *Differences between feature-based and direct methods.*

   Answer. Feature-based Methods

   (a) Extract and match features (+RANSAC)

   (b) Minimize Reprojection Error:

   $$T_{k,k-1} = \text{argmin}_T \sum_i ||u_i' - \pi(p_i)||^2_\Sigma \tag{117}$$

   Good: Large frame-to-frame motions, accuracy and efficient optimization of SFM (BA).
   Bad: Slow due to costly feature extraction and matching, matching outliers (RANSAC).

   Direct Methods (all pixels)

(a) Minimize **photometric error**:

$$T_{k,k-1} = \text{argmin}_T \sum_i ||I_k(u_i') - I_{k-1}(u_i)||_\sigma^2, \tag{118}$$

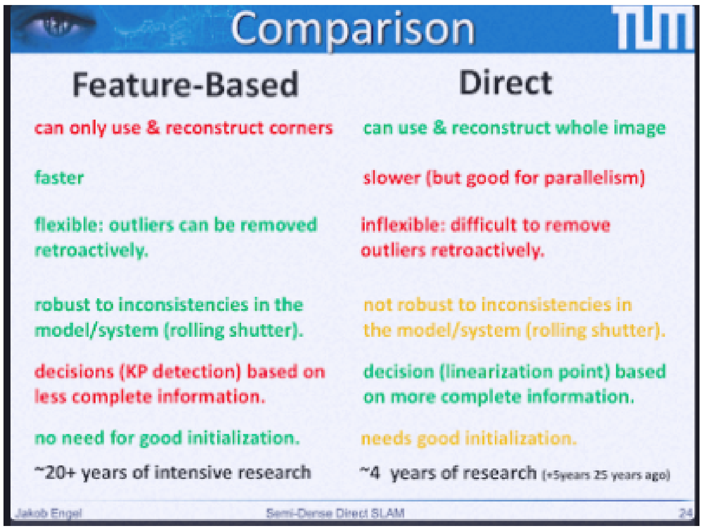
where

$$u_i' = \pi(T \cdot (\pi^{-1}(u_i) \cdot d)) \tag{119}$$

Good: All information in the image can be exploited. Increasing camera frame-rate reduces computational cost per frame.
Bad: Limited frame to frame motion. Joint optimization of dense structures and motion too expensive.

Feature based methods process the image to find corners to compare. This is an issue, as it doesn't work well in human environments, as many any straight or curved edges would be discarded, meaning that the information is less complete. Also, a picture which has most of its features concentrated in a small area is of less interest to the algorithm as a picture with many details all over, as the *features* cannot overlap. Another issue with feature based methods is that storing the processed features can quickly become very costly. However, since this method eliminates all data that cannot be used (non features points), it is faster than direct methods. It is possible to reconstruct dense maps from feature based methods by estimating the camera positions to find what was at the given location.

Direct Methods however, compare the entire images to each other to reference them to each other, finding which parts go together. It can create semi dense 3D maps in real time on a smartphone using semi dense filtering algorithms. This means it provides more information about he environment, making it more interesting to use in robotics or Augmented Reality, as well as giving a more meaningful representation to the human eye. Some disadvantages of Direct methods are that they cannot handle outliers very well, as they will always try to process them an implement them into the final map, and that they are slower than feature based variants.



Comparison

Feature-Based	Direct
can only use & reconstruct corners	can use & reconstruct whole image
faster	slower (but good for parallelism)
flexible: outliers can be removed retroactively.	inflexible: difficult to remove outliers retroactively.
robust to inconsistencies in the model/system (rolling shutter).	not robust to inconsistencies in the model/system (rolling shutter).
decisions (KP detection) based on less complete information.	decision (linearization point) based on more complete information.
no need for good initialization.	needs good initialization.
~20+ years of intensive research	~4 years of research (+5years 25 years ago)

Jakob Engel Semi-Dense Direct SLAM 24

Figure 26: Comparison direct vs feature based.

Lecture 10 Multi view stereo

1. *Working principle (aggregated photometric error).*

   Answer. For the 3D reconstruction from multiple views, we assume that camera are calibrated

   - **intrinsically** (K is known for each camera), and
   - **extrinsically** (T and R between cameras are known, for instance, from SFM).

   For the multi-view stereo, we have as:
   Input: calibrated images from several viewpoints.
   From a *dense* region of pixels (hence not only from corners) estimate the structure.
   The workflow is:

   (a) **Local methods**: estimate depth for every pixel independently. (Not all the pixels can be matched reliably, due to viewpoint changes, occlusions.)
   We take advantage of *many* small baseline views, where *high quality* matching is possible.

   (b) **Global methods**: refine the depth surface as a whole by enforcing smoothness constraint.
   We use the **photometric error** (SSD): the aggregated photometric error plot is derived for every combination of the reference image and any further image.
   IDEA: optimal depth minizes the photometric error in all images as a function of the depth in the first image.
   The SSD between corresponding patches of intensity values (min patch size: 1×1 pixels) = photometric error $\rightarrow$ dense correspondences $\rightarrow$ dense reconstruction

   Output: 3D object dense reconstruction.
   Recall: The two camera centers and the image point p determine the epipolar plane, which intersects each camera image plane in the epipolar lines. Since we use the epipolar constraint, corresponding points only need to be searched along epipolar lines.

2. *What are the differences in the behavior of the aggregated photometric error for corners, flat regions, and edges?*

   Answer. Plot: x-axis inverse depth d, y-axis aggregated photometric error C(a,d), a= image point

   - The aggregated photometric error for *flat regions* and *edges* parallel to the epipolar line show **flat valleys** (noise!).
   - For distinctive features like corner, the aggregated photometric error has one clear minimum.
   - Repetitive texture shows multiple minima.

3. *What is the Disparity Space Image (DSI) and how is it built in practice?*

Answer. For a given image point (u, v) and for discrete depth hypotheses d, the aggregate photometric error $C(u, v, d)$ with respect to the reference image I_r can be stored in a volumetric 3D grid called the *Disparity Space Image (DSI)*, where each voxel (group of u, v, d) has value

$$C(u,v,d) = \sum_k \underbrace{\rho\left(\tilde{I}_k(u',v',d) - I_r(u,v)\right)}_{photometricerror(SSD)}, \tag{120}$$

where $\tilde{I}_k(u', v', d)$ is the patch of intensity values in the k-th image centered on the pixel (u', v') corresponding to the patch $I_r(u, v)$ in the reference image I_r an depth hypothesis d.

4. *How do we extract the depth from the DSI?*

Answer. The depth is the solution to a function $d(u, v)$ in the DSI that satisfies:

$$\begin{aligned}&\text{Minimum aggregated photometric error (i.e. } argmin_d C\text{)}\\ &\text{AND}\\ &\text{Piecewise smooth (global methods)}\end{aligned} \tag{121}$$

Interpolating while not overfitting!

5. *How do we enforce smoothness (regularization) and how do we incorporate depth discontinuities (mathematical expressions)?*

Answer. Global Methods: We smooth the image in terms of energy minimization. The objective is to find a surface $d(u, v)$ that minimizes a global energy

$$E(d) = \underbrace{E_d(d)}_{\text{data term}} + \underbrace{\lambda \cdot E_s(d)}_{\text{regularization term}}, \tag{122}$$

where

$$E_d(d) = \sum_{(u,v)} C(u, v, d(u, v)) \tag{123}$$

and

$$E_s(d) = \sum_{(u,v)} \rho_d(d(u, v) - d(u + 1, v)) + \rho_d(d(u, v) - d(u, v + 1)). \tag{124}$$

ρ_d is a norm (e.g. the $L_{1,2}$ or Huber norm).

- **The regularization term** $E_s(d)$
  - **Smooths** non smooth surfaces (result of noisy measurements) as well as discontinuities.
  - Fills the holes.
- Popular assumption: discontinuities in intensity **coincide** with discontinuities in depth.

- Control **smoothness penalties** according to image gradient (discrete)

$$\rho_d(d(u,v) - d(u+1,v)) \cdot \rho_I(||I(u,v) - I(u+1,v)||) \tag{125}$$

- ρ_I is some monotically *decreasing* function of intensity differences: **lower** smoothness cost for **high intensity gradients** (if there are high intensity gradients, you don't want to smooth them as they are a crucial information in your image.

6. *What happens if we increase lambda (the regularization term)? What if lambda is 0? And if lambda is too big?*

   Answer. λ controls the tradeoff data (regularization)

   - Higher λ = higher **smoothing**!
   - $\lambda = 0 \rightarrow$ no smoothing
   - λ too big $\rightarrow$ oversmoothing. can't recognize different depths anymore

7. *What is the optimal baseline for multi-view stereo?*

   Answer.

   - Too small: large depth error.
   - Too large: difficult search problem.

   Solution:

   - Obtain depth map from small baselines
   - When baseline becomes too large create new reference frame (keyframe) and start new depth computation

   A possible approach is **depth map fusion** (different depth maps with different perspectives gives a complete image).

8. *What are the advantages of GPUs?*

   Answer. General Purpose Computing on Graphics Processing Unit. It can perform more demanding calculations than CPU because

   - GPUs run thousand of lightweight threads **in parallel**
     - more transistors for data processing.
     - Typically on consumer hardware: 1024 threads per multiprocessor, 30 multiprocessors: 30000 threads. CPU with 4 cores which supports 32 threads.
   - well suited for **data-parallelism**
     - the same instructions executed on multiple data in parallel
     - high **arithmetic intensity**: arithmetic operations/ memory operations

   Those characteristics lead to

   - Fast pixel processing (ray tracing, draw textures, shaded triangles,..)

- Fast matrix/vector operations (transform vertices)
- Programmable (shading, bump mapping)
- Floating-point support (accurate computations)
- Deep learning.

Lecture 11 Tracking

1. *Illustrate tracking with block matching.*

   Answer. Block matching is an approach for point tracking. given two images, estimate the motion of a pixel point from image_1 to image_2

   - Search for the corresponding patch in a *neighborhood* around the point.
   - Use SSD, SAD, NCC to search for corresponding patches in a local neighborhood of the point. The search region usially is a $D \times D$ squared patches. We have to perform $D \times D$ comparisons, computationally demanding.

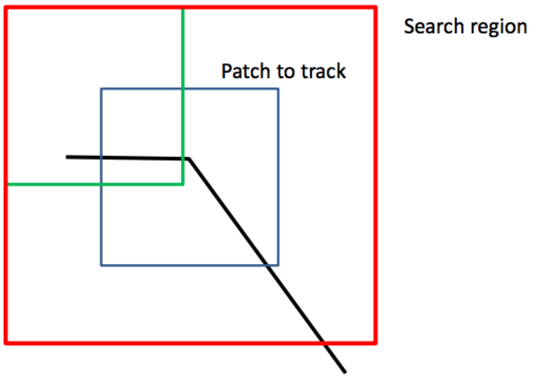


Figure 27: Block matching.

2. *Differential Methods*

   (a) *Describe the underlying assumptions, derive the mathematical expression, and meaning of the M matrix.*

   Answer.

   - Look at the local brightness changes at the **same** location. **NO patch shift** is performed. (centered in the same point!)

   Spatial Coherency We assume that all the pixels in the patch undergo the same motion (same u and v). Also, assume that the time interval between the two images I_0 and I_1 is small. We want to find the motion vector (u, v) that minimizes the Sum of Squared Differences (SSD). As we did for Harris, we look at the first order Taylor approximation of the sum:

$$
\begin{aligned}
SSD &= \sum (I_0(x,y) - I_1(x+u, y+v))^2 \\
&\approx \sum (I_0(x,y) - I_1(x,y) - I_x \cdot u - I_y \cdot v)^2 \\
&= \sum (\Delta I - I_x \cdot u - I_y \cdot v)^2 \\
&= E,
\end{aligned}
\tag{126}
$$

which is a simple quadratic function in two variables (u, v). To minimize the E, we differentiate with respect to (u, v) and equate to 0.

$$\begin{aligned} \frac{\partial E}{\partial u} &= 0 \Rightarrow -2I_x \sum(\Delta I - I_x \cdot u - I_y \cdot v) = 0 \\ \frac{\partial E}{\partial v} &= 0 \Rightarrow -2I_y \sum(\Delta I - I_x \cdot u - I_y \cdot v) = 0 \end{aligned} \tag{127}$$

Linear system of two equations in two unknowns. We can write this in matrix form

$$\begin{aligned} \begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} &= \begin{pmatrix} \sum I_x \cdot \Delta I \\ \sum I_y \cdot \Delta I \end{pmatrix} \\ \begin{pmatrix} u \\ v \end{pmatrix} &= \underbrace{\begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{pmatrix}^{-1}}_{M} \cdot \begin{pmatrix} \sum I_x \cdot \Delta I \\ \sum I_y \cdot \Delta I \end{pmatrix} \end{aligned} \tag{128}$$

These are not matrix products, but pixel-wise products!

(b) *When Is this matrix invertible and when not?*

Answer. For M to be invertible, its determinant should be non 0. From the decomposition

$$M = R^{-1} \cdot \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \cdot R, \tag{129}$$

we know that $\det(M)$ is non zero when its eigenvalues are large (i.e. not a flat region and not an edge). In practice, it should be a corner or in general contain **texture**.

(c) *What is the aperture problem and how can we overcome it?*

Answer. If we look at local brightness changes through a small aperture, we cannot always determine the motion direction, because **infinite** motion directions (solutions) may exist (along a edge). The solution is to **increase** the aperture size.

(d) *What is optical flow?*

Answer. Optical flow is an application of **differential method** and is the pattern of apparent motion of objects in a visual scene, caused by the relative motion between the observer (eye or camera) and the scene. It tracks the motion of every pixel between two consecutive frames. For each pixel, we compute

- The vector direction and,
- The vector length (amount of movement).

An issue could be the choice of the right patch size. It can also be applied to corner tracking.

3. *Pros and Cons of block-based vs differential methods for tracking*

Answer. Block-based Methods: search for the corresponding patch in a neighborhood of the point to be tracked. The search region is usually a square of $n \times n$ pixels.

- +: **robust to large motions**
- −: Can be computationally expensive ($n \times n$ comparisons for a single point track)

Differential Methods:

- +: Much more efficient than block-based methods. Thus, can be used to track the motion of every pixel in the image. It avoids searching in the neighborhood of the point by analyzing the **local intensity changes** of an image patch at a specific location (no search is performed).
- −: Works only for **small** motions (high frame rate). For larger motion, multi-scale implementations are used, but are then expensive.

4. Lucas-Kanade algorithm

   (a) *Working principle of KLT and derivation of the underlying mathematical expression (only first two slides titled derivation of the Lucas-Kanade algorithm, slide pp. 55-56)*

   Answer. Template Warping Given the template image $T(x)$, take all the pixels from the template image and warp them using the function $W(x,p)$ parameterized in terms of parameters p. The goal of template-based tracking is to find the set of warp parameters p such that

   $$I(W(x,p)) = T(x). \tag{130}$$

   Assumptions are:

   - No errors in the template image boundaries: only the appearance of the object to be tracket appears in the template image.
   - No occlusion: the entire template is visible in input image.
   - Brightness consistency assumption: the intensity of the object appearance is always the same across different views.

   The **algorithm**:

   i. Warp $I(x)$ with $W(x,p)$.
   ii. Compute the error.
   iii. Compute **warped** gradients, ∇I evaluated at $W(x,p)$.
   iv. Evaluate the Jacobian of the warping $\frac{\partial W}{\partial p}$.
   v. Compute the inverse Hessian H^{-1}.
   vi. Multiply steepest descend with error.
   vii. Comptue Δp.
   viii. Update parameters $p \leftarrow p + \Delta p$.
   ix. Repeat until $\Delta p < \varepsilon$.

   The algorithm follows a predict-correct cycle. A prediction $I(W(x,p))$ of the warped image is computed from an initial estimate.
   It uses the Gauss-Newton method for minimization, i.e.

   - Applies a first order approximation of the warp,

- Attempts to minimize the SSD iteratively.

This is solved by determining p that minimizes the Sum of Squared Differences

$$E = SSD = \sum_{x \in T} \left[I(W(x,p)) - T(x) \right]^2 . \tag{131}$$

Then, we want to find the increment Δp that minimizes

$$\sum_{x \in T} \left[I(W(x,p+\Delta p)) - T(x) \right]^2 . \tag{132}$$

The first order Taylor approximation of the term in brackets reads

$$I(W(x,p+\Delta p)) \approx I(W(x,p)) + \underbrace{\nabla I}_{\text{image gradient}} \underbrace{\frac{\partial W}{\partial p}}_{\text{Jabobianof the warp}} \Delta p. \tag{133}$$

The image gradient $\nabla l = [I_x, I_y]$ is evaluated at W(x,p). By replacing that in the equation we get

$$E = \sum_{x \in T} \left[I(W(x,p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right]^2 . \tag{134}$$

In order to minimize it, we differentiate and equate to 0, i.e.

$$\frac{\partial E}{\partial \Delta p} = 0. \tag{135}$$

(b) *What is the Hessian matrix and for which warping function does it coincide to that used for point tracking?*

Answer.

$$\Delta p = H^{-1} \sum_{x \in T} \left(\nabla I \frac{\delta W}{\delta p} \right)^T (T(x) - I(W(x,p))), \tag{136}$$

where

$$H = \sum_{x \in T} \left(\nabla I \frac{\partial W}{\partial p} \right)^T \left(\nabla I \frac{\partial W}{\partial p} \right) \tag{137}$$

is the second moment matrix of the warped image (Hessian).
Translation ?

$$\nabla W_p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \nabla I \nabla W_p = \begin{bmatrix} I_x & 0 \\ 0 & I_y \end{bmatrix} \Rightarrow H = \begin{bmatrix} I_x I_x & 0 \\ 0 & I_y I_y \end{bmatrix} \tag{138}$$

(c) *Lucas-Kanade failure cases and how to overcome them*

Answer.
- If the initial estimate is too far, the linear approximation does not longer hold. Or aliasing because many pixel have the same intensity. Solution are **pyramidal implementations** / coarse-to-fine implementation

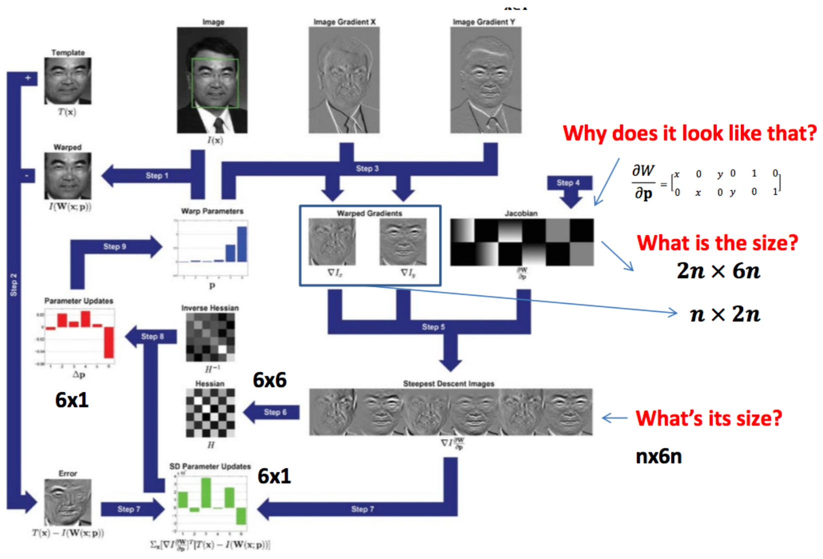


Figure 28: KLT algorithm

- if Illumination changes, object deformations and Occlusions are in the initial image, the tracking will drift. A solution can be to update the template with newest image.
- In order to deal with wrong prediction, it can be implemented in a Particle-Filter fashion.

(d) *How do we get the initial guess?*

Answer. We set the initial guess to zero $p = 0$, because there is not initial guess available at the deepest pyramidal level.

(e) *Illustration of coarse-to-fine Lucas-Kanade implementation.*

Answer. Due to the small motion assumption, regular optical flow methods work bad if the object we are tracking moves a long distance. Building image pyramids for each image and doing optical flow on each layer of the pyramid (to get rid of small motion constraints).For building the pyramid, first reduce the image resolution using Gaussian blur and then scale the image down.

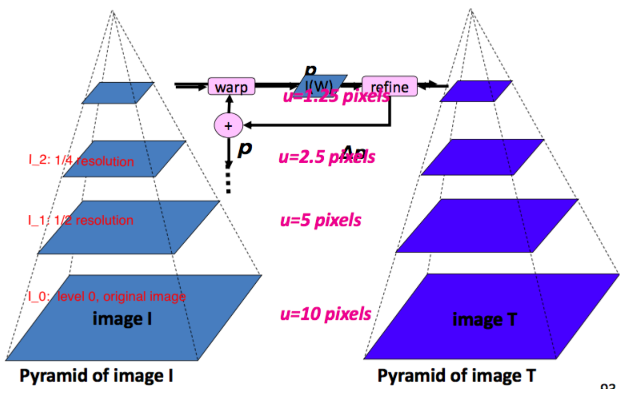


Figure 29: Pyramidal implementation.

(f) *llustrate alternative tracking using point features.*

Answer.

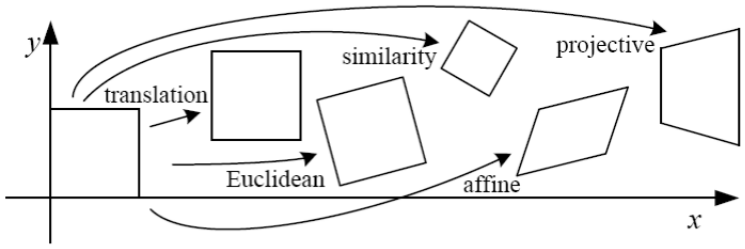
i. Keypoint detection and matching. Invariant to scale, rotation or perspective.
ii. Geometric verification (RANSAC)

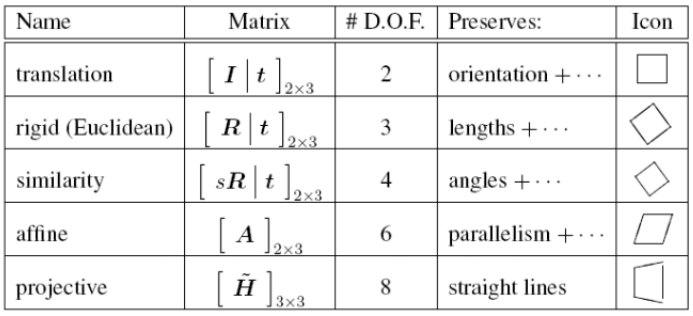
Issues here are:

- How to segment the object to track from background?
- How to initialize the warping?
- How to handle occlusions?
- How to handle illumination changes andn on modeled effects?

5. *List one or more possible ways of discarding wrong feature tracks in practice.*

Answer. ???





Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\, \boldsymbol{I} \mid \boldsymbol{t} \, \right]_{2\times3}$	2	orientation + ⋯	□
rigid (Euclidean)	$\left[\, \boldsymbol{R} \mid \boldsymbol{t} \, \right]_{2\times3}$	3	lengths + ⋯	◇
similarity	$\left[\, s\boldsymbol{R} \mid \boldsymbol{t} \, \right]_{2\times3}$	4	angles + ⋯	◇
affine	$\left[\, \boldsymbol{A} \, \right]_{2\times3}$	6	parallelism + ⋯	▱
projective	$\left[\, \tilde{\boldsymbol{H}} \, \right]_{3\times3}$	8	straight lines	⏢

- Warping function $W(x, p)$
- **p** set of parameters $p = (a_1, a_2, ...a_n)$
- **Translation**: 2 DOF

$$W(x,p) = \begin{pmatrix} x + a_1 \\ y + a_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a_1 \\ 0 & 1 & a_2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{139}$$

- **Euclidean**: 3 DOF

$$W(x,p) = \begin{pmatrix} x\cos(\alpha) - y\sin(\alpha) + a_1 \\ x\sin(\alpha) + y\cos(\alpha) + a_2 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) & a_1 \\ \sin(\alpha) & \cos(\alpha) & a_2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{140}$$

- **Affine**: 6 DOF

$$W(x,p)=\begin{pmatrix}a_1x+a_3y+a_5\\a_2x+a_4y+a_6\end{pmatrix}=\begin{pmatrix}a_1&a_3&a_5\\a_2&a_4&a_6\end{pmatrix}\cdot\begin{pmatrix}x\\y\\1\end{pmatrix}. \tag{141}$$

- **Projective (homography)**: 8 DOF

$$x'=\frac{a_1x+a_2y+a_3}{a_7x+a_8y+1},\quad y'=\frac{a_4x+a_5y+a_6}{a_7x+a_8y+1}. \tag{142}$$

$$W(\tilde{x},p)=\begin{bmatrix}a_1&a_2&a_3\\a_4&a_5&a_6\\a_7&a_8&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix} \tag{143}$$

Recalling that the Jacobian of a function

$$F(x_1,x_2,\dots,x_n)=\begin{pmatrix}f_1(x_1,x_2,\dots,x_n)\\\vdots\\f_m(x_1,x_2,\dots x_n)\end{pmatrix} \tag{144}$$

is

$$J(F)=\nabla F=\begin{pmatrix}\frac{\partial f_1}{\partial x_1}&\cdots&\frac{\partial f_1}{\partial x_n}\\&\vdots&\\\frac{\partial f_n}{\partial x_1}&\cdots&\frac{\partial f_n}{\partial x_n}\end{pmatrix} \tag{145}$$

Displacement-model with Jacobians ∇W_p

Lecture 12 Place Recognition

Bag of Words

1. *Inverted file index*

   Answer. For text documents, an index is important. We want to find every image in which a feature occurs. How many SIFT or BRISK features exist ?

   - SIFT: infinite.
   - BRISK-128: $2^{128} = 2.4 \cdot 10^{38}$.

   For this reason, we need to create visual words, then put them into a vocabulary. Basically, we collect images and we extract features. A visual word is the centroid of a cluster. We then cluster the descriptors with the different words.
   An **inverted file index** lists all visual words in the vocabulary (extracted at training time). Each word points to a list of images from the all image Data Base, in which which that word appears. The DB grows as the robot navigates and collects new images.
   Voting Array: has as many cells as the images in the DB. Each word in the query image votes for an image.

2. *What is a visual word?*

   Answer. A visual word is the centroid of a cluster and is a part of a **visual vocabulary**.
   Image collection → extract features → order the features based on their descriptors in the descriptor space → cluster the descriptors

3. *Why do we need hierarchical clustering?*

   Answer. We need hierarchical clustering to build the **inverted file index** and enable the recognition.

4. *How does K-means clustering work?*

   Answer. This is an algorithm to partition n observations into k clusters in which each observation x belongs to the cluster S_i with centroid m_i. It minimizes the sum of squared Euclidean distances between points x and their nearest cluster centers m_i

   $$D(X, M) = \sum_{i=1}^{k} \sum_{x \in S_i} (x - m_i)^2. \tag{146}$$

   The algorithm reads

   (a) Randomly initialize k cluster centers
   (b) Iterate until convergence:
      - Assign each data point x_i to the nearest center m_i.
      - Recompute each cluster center as the mean of all points assigned to it.

5. *Explain and illustrate image retrieval with Bag of Words.*

Answer. Bag of Words can be applied to image classification, by treating image features as words. In document classification, a bag of words is a sparse vector of occurrence counts of words; that is, a sparse histogram over the vocabulary. In computer vision, a bag of visual words is a vector of occurrence counts of a vocabulary of local image features.
This algorithm simply computes the distribu-tion (histogram) of visual words found in the query image and compares this distribution to those found in the training images (vocabulary).

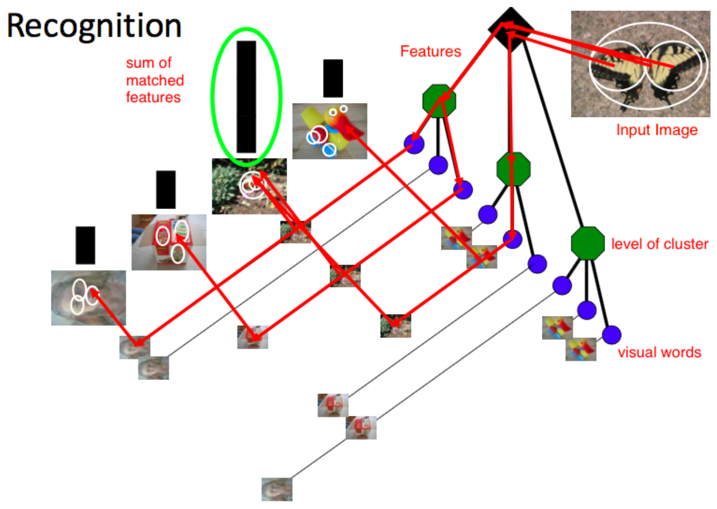


Figure 30: Bag-of-Words Retrieval.

6. *Discussion on place recognition: what are the open challenges and what solutions have been proposed?*

Answer. Visual Vocabulary discards the spatial relationships between features: two images with the same features shuffled around will return a 100% match when using only appearance information. This can be overcome with **geometric verification**: test the h most similar images to the query image for geometric consistency (5,8 point RANSAC) and retain the image with the smallest reprojection error and largest number of inliers. More words is better!

Lecture 13 Visual Inertial Fusion

Inertial Measurement Unit: angular velocity, linear acceleration

1. (a) *Why an IMU for VO?*

   Answer.

   - Monocular vision is scale ambiguous.
   - Pure vision is not robust enough (Tesla accident):
     - Low texture.
     - High dynamic range.
     - High speed motion.

   Why not just IMU?
   Pure IMU integration will lead to large drift (especially cheap IMUs). Integration of angular velocity to get orientation: error **proportional to** t. Double integration to get position: if there is a bias in acceleration, the error of position is **proportional to** t^2**.** The actually position error also depends on the error of orientation.

   (b) *How does a MEMS IMU work?*

   Answer.

   - Mechanical: spring/damper system.
   - Optical: Phase shift projected laser beams is proportional to angular velocity.
   - MEMS (accelerometer): a spring-like structure connects the device to a seismic mass vibrating in a capacitive divider. A capacitive divider converts the displacement of the seismic mass into an electric signal. Damping is created by the gas sealed in the device.
   - MEMS (gyroscopes): measure the Coriolis forces acting on MEMS vibrating structures. Their working principle is similar to the haltere of a fly.

   (c) *Whats the drift of an industrial IMU?*

   Answer. Accelerometer Bias Error: 3 mg
   Drift: 1s = 15 mm , 10s = 1.5m, 60s = 53 m

   - Integration of angular velocity to get orientation: error proportional to t Double integration of acceleration to get position: if there is a bias in
   - acceleration, the error of position is **proportional** to t^2
   - Worse, the actually position error also depends on the error of orientation.

2. *What is the IMU measurement model (formula)?*

   Answer.

$$\begin{aligned} \tilde{\omega}_{WB}^{B}(t) &= \omega_{WB}^{B}(t) + b^{g}(t) + n^{g}(t) \\ \tilde{a}_{WB}^{B}(t) &= R_{BW}(t) \cdot \left(a_{WB}^{W}(t) - g^{W}\right) + b^{a}(t) + n^{a}(t) \end{aligned} \tag{147}$$

where g stands for Gyroscope and a for accelerometer. The noise is additive Gaussian white noise. The bias has own dynamics

$$\dot{b}(t) = \sigma_b \cdot w(t), \tag{148}$$

i.e. the derivative of the bias is white Gaussian noise (random walk). In discrete time, one writes

$$b[k] = b[k-1] + \sigma_{bd} \cdot w[k], \quad w[k] \sim \mathcal{N}(0,1), \quad \sigma_{bd} = \sigma_b \cdot \sqrt{t} \tag{149}$$

3. *What causes the bias in an IMU?*

   Answer. Bias can be estimated.

   - Can change due to temperature change, mechanical pressure,..
   - Can change everytime the IMU is started.

4. *How do we model the bias?*

   Answer. Integration leads to

$$p_{Wt_2} = P_{Wt_1} + (t_2 - t_1)v_{Wt_1} + \int\int_{t_1}^{t_2} R_{Wt}(t)\left(\tilde{a}(t) - b^a(t) + g^w\right) \mathrm{d}t^2, \tag{150}$$

   which depends on initial position and velocity. The rotation $R(t)$ can be computed with a gyroscope.

5. *How do we integrate the acceleration to get the position (formula)?*

   Answer. ???

6. *Definition of Loosely coupled vs tightly coupled visual inertial fusion*

   Answer. Loosely Coupled Approach: Treats VO and IMU as two separate (not coupled black boxes). Each block estimates **pose and velocity** from visual and inertial data (pose and velocity up to a scale and inertial data in absolute scale). **Tightly Coupled Approach**: Makes use of the raw sensors' measurements: 2D

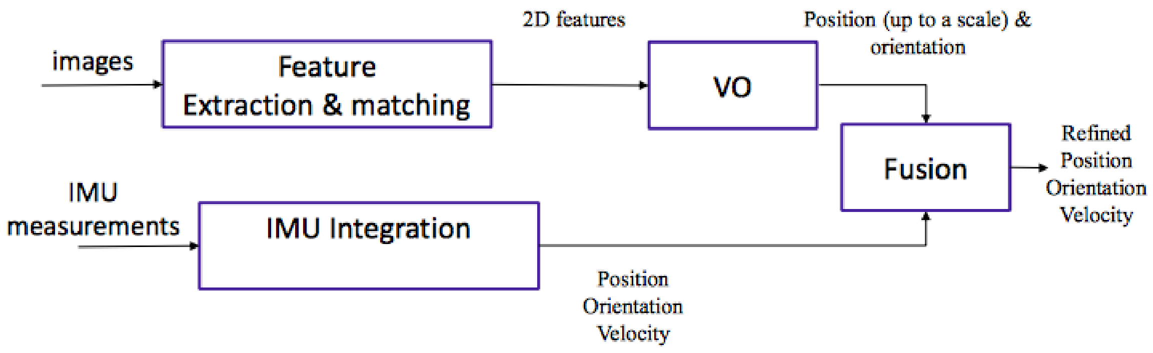

images
Feature Extraction & matching
2D features
VO
Position (up to a scale) & orientation
Fusion
Refined Position Orientation Velocity
IMU measurements
IMU Integration
Position Orientation Velocity

Figure 31: Loosely Coupled Approach.

features, IMU readings, more accurate, more implementation effort. System states are:

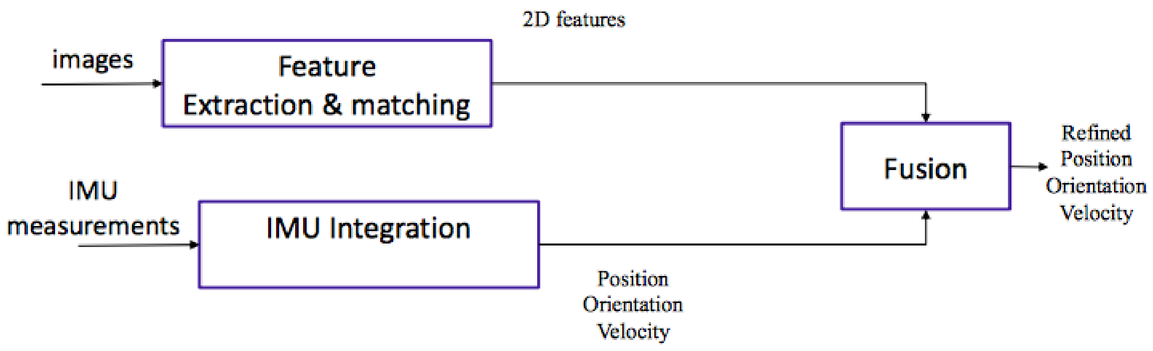

2D features
images
Feature Extraction & matching
Fusion
Refined Position Orientation Velocity
IMU measurements
IMU Integration
Position Orientation Velocity

Figure 32: Tightly Coupled Approach.

- **Tightly Coupled:** $X = \big(p_W(t); q_{WB}(t); v_W(t); b^a(t); b^g(t); L_{w,1}; \dots ; L_{w,K}\big)$, with L Landmarks.
- **Loosely Coupled** $X = \big(p_W(t); q_{WB}(t); v_W(t); b^a(t); b^g(t)\big)$

7. *How can we use non-linear optimization-based approaches to solve for visual inertial fusion (mathematical expression and graphical illustration of the pose graph)?*

   Answer. Maximum a Posterior Estimation (MAP): Fusion solved as a non-linear optimization problem. Increased accuracy over filtering methods. We have

$$x_k = f(x_{k-1}), \quad z_k = h(x_{i_k}, l_{i_j}), \tag{151}$$

where X are the robot states, L the 3D points and Z the features and IMU measurements. It holds

$$\begin{aligned}\{X^*, L^*\} &= \operatorname{argmax}_{X,L} P(X, L|Z)\\ &= \operatorname{argmin}_{X,L}\{\underbrace{\sum_{k=1}^{N} \|f(x_{k-1}) - x_k\|^2_{\Lambda_k}}_{\text{IMU residuals}} + \underbrace{\sum_{i=1}^{M} \|h(x_{i_k}) - z_i\|^2_{\Sigma_i}}_{\text{Reprojection residuals}}\}\end{aligned} \tag{152}$$

An open problem is consistency:

- Filters: Linearization around different values of the same variable may lead to error.
- Smoothing methods: may get stuck in local minima.

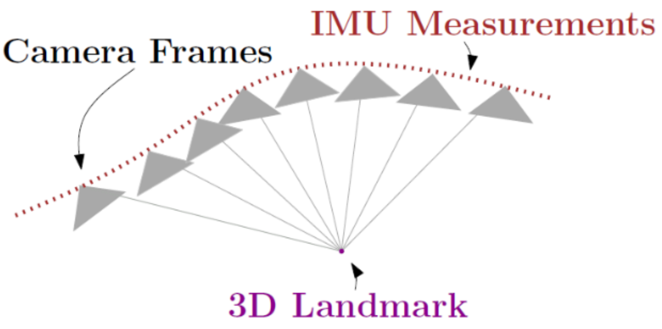


Figure 33: Max a Posterior Estimation (MAP).

Lecture 14 Event-based Vision

1. *Whats a DVS and how does it work?*

   Answer. Event cameras enable **low-latency sensory motor control** $< 1ms$. A DVS outputs **asynchronous** events at microsecond resolution. An event is generated each time a single pixel detects an **intensity changes**:

$$\text{event:}\langle t, < x, y >, \text{sign}\left(\frac{\mathrm{d}I(x,y)}{\mathrm{d}t}\right)\rangle \tag{153}$$

   All pixels are independent from another. Implements **level-crossing** sampling. Reacts to **logarithmic** brightness changes. Each pixel is independent of all the other pixels. Events are generated everytime a single pixel sees a change of the logarithm of the brightness that is equal to C, i.e.

$$|\log(I)| = |\log(I(t+\Delta t) - \log(I(t))| = C, \tag{154}$$

   where $C \in [0.15, 0.20]$ is called **contrast sensitivity** and can be tuned by the user. Since brightness can be either positive or negative, we have ON event if $= C$ and OFF event if $= -C$. Traditional sampling is performed with the discriminant

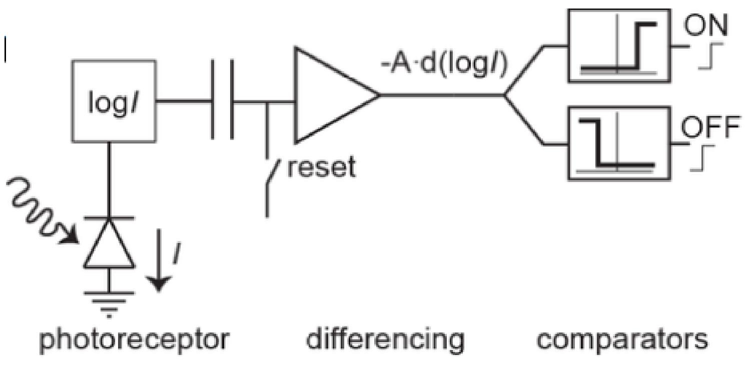


Figure 34: DVS circuit

   (time) on $x-$axis. Level-crossing sampling works with the change in intensity, in the $y-$axis.

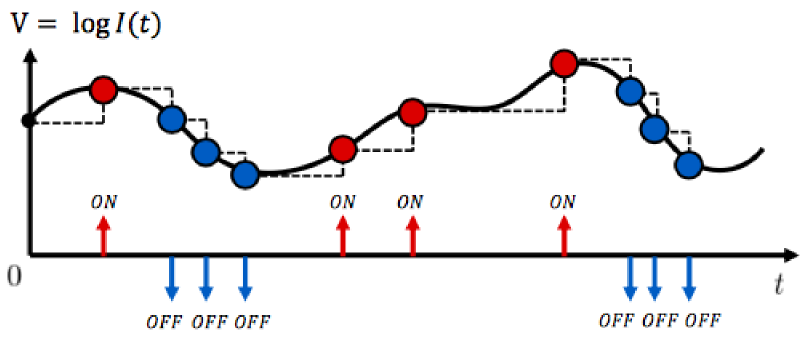


Figure 35: DVS.

2. *What are its pros and cons vs standard cameras and vs high speed cameras?*

   Answer. Advantages:

   - Low latency (1 micro second)
   - High dynamic range (140 dB instead of 60 dB)
   - Low power: 10mW instead of 1W

   Disadvantages:

   - Paradigm shift: requires totally new vision algorithms
     - Asynchronus pixels,
     - No intensity information (only binary intensity changes).

3. *Can we apply standard camera calibration techniques?*

   Answer. The standard pinhole camera model is still valid (same optics). Standard passive calibration cannot be used: we would need to move the camera. **Blinking patterns** (computer screen, LEDs).

4. *How can we compute optical flow with a DVS?*

   Answer. White pixels become black, i.e. the brightness decrease, i.e. negative events (black color). Events are represented by dots. At what speed is the edge moving? $v = \frac{\Delta x}{\Delta t}$. Two different approaches

   - **Event-by-event processing** (i.e. estimate the state event by event): Pros: low latency, Cons: with high speed motion, there are dozens of millions of events per seconds (GPU)
     Let's start with an approximation:

$$\begin{aligned}\Delta \log(I) &= \frac{\partial \log(I)}{\delta t}\Delta t \\ &= \frac{1}{I}\frac{\delta I}{\delta t}\Delta t \\ &= \frac{\partial I}{I}.\end{aligned} \tag{155}$$

- **Event-packet processing** (i.e. process the last N events): Pros: N can be tuned to allow real-time performance on a CPU. Cons: no longer microsecond resolution (when is this really necessary=)

5. *Intuitive explanation of why we can reconstruct the intensity.*

   Answer. The intensity signal at the event time can be reconstructed by integration of $\pm C$. Given the events and the camera motion (rotation), recover the absolute brightness. **Explanation:** An event camera naturally responds to edges, hence, if we know the motion, we can relate the events to world coordinates to get an edge/gradient map. Then, just integrate the gradient map to get absolute intensity.

   (a) Recover the gradient map of the scene. Let $L = \log(I)$. Then

   $$\Delta L(t) = L(t) - L(t - \Delta t) = C. \tag{156}$$

   In terms of the brightness map $M(x, y)$:

   $$M(p_m(t)) - M(p_m(t - \Delta t)) \approx g \cdot v \cdot \Delta t, \tag{157}$$

   with $g = \nabla M(p_m(t))$.

   (b) Integrate the gradient to obtain brightness. Poisson reconstruction: integrate the gradient map g to get absolute brightness M.

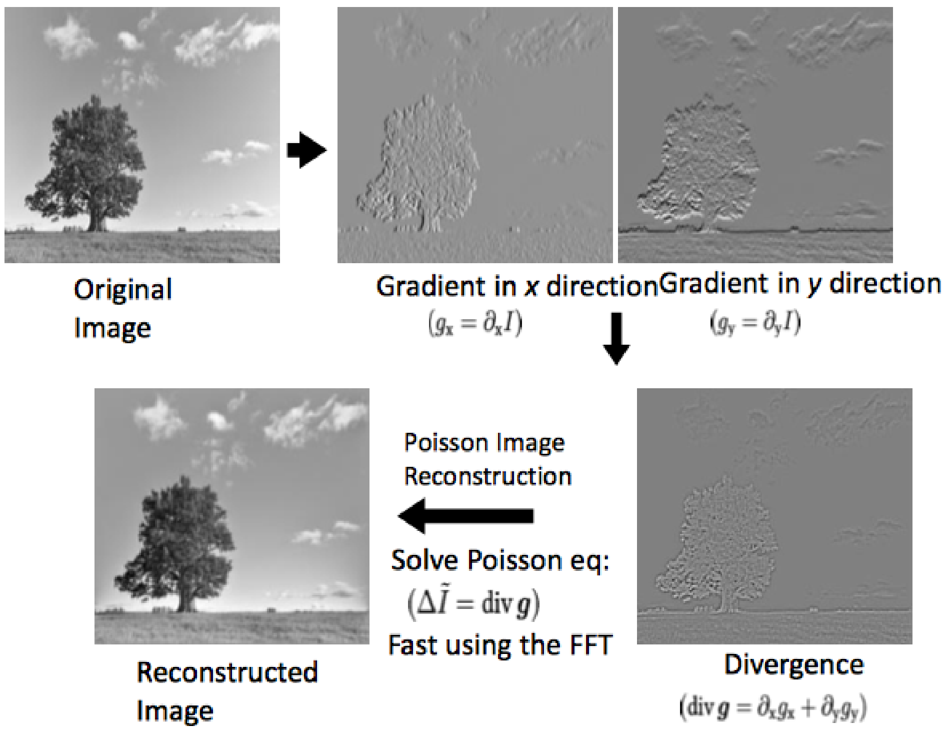


Figure 36: DVS vs High Speed Cameras.

6. *What is the generative model of a DVS?*

   Answer. ???

7. *What is a DAVIS sensor?*

   Answer. Combines an event sensor (DVS) with a standard camera in the same pixel array. Output are frames (at 30 Hz) and events (asynchronous). One can them perform SLAM with an IMU, which increases robustness and accuracy.
   Open problems for DVS are: noise modeling, asynchronous feature and object detection and tracking, sensor fusion, asynchronous learning and recognition, estimation and control, low power computation.

8. *Can you write the equation of the event generation model and its proof?*

   Answer. To simplify the notation, let's assume that $I(x, y, t) = \log(I(x, y, t))$. Consider a given pixel $p(x, y)$ moving with apparent motion $\vec{u} = (u, v)$ (i.e. induced by a moving 3D patch). It can be shown, that an event is generated if the scalar product between the gradient and the appearent motion vector u is equal to C.

$$-\nabla I \cdot u = C \tag{158}$$

   Proof. The proof comes from the brightness constancy assumption, which says that the intensity value of p, before and after the motion, must remained unchanged

$$I(x, y, t) = I(x + u, y + v, t + \Delta t) \tag{159}$$

   By replacing the right-hand term by its first order approximation at $t + \Delta t$, we get

$$\begin{aligned} I(x, y, t) &= I(x, y, t + \Delta t) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v \\ I(x, y, t + \Delta t) - I(x, y, t) &= -\frac{\partial I}{\partial x} u - \frac{\partial I}{\partial y} v \\ \Rightarrow \Delta I &= C = -\nabla \cdot u. \end{aligned} \tag{160}$$

   This equation described the linearized event generation equation for an event generated by a gradient ∇I that moved by a motion vector u (optical flow) during a time interval Δt. 1 Equation, 2 Unknowns, solution is to add events. □

Lecture 15 Visual (inertial) Odometry

1. *List one or more possible ways to decide when to triangulate new landmarks based on feature tracks.*

   Answer. Triangulate new landmarks when

   - Angle between landmark and camera is sufficient.
   - Keypoints were tracked over certain number of frames.

2. *What are keyframes in Visual Odometry?*

   Answer.

   - Frames, where new landmarks are triangulated.

3. *Why are they needed? Are they strictly necessary?*

   Answer.

   - For BA, you can optimize only on desired frames.
   - Reduces computational complexity without reducing accuracy.

4. *How to decide whether a given frame should be a keyframe?*

   Answer.
   - Possible to use every frame as keyframe in a continuous manner rather than discrete, by keeping track of first observation of landmarks (and pose) and triangulate, if angle large enough.

5. *Do you know any popular Visual (Inertial) Odometry algorithm? (one or more). Can you explain briefly how they work?*

   Answer. Kalman filtering Visual odometry (see MSCKF for planetary landing I quoted in the final report). It fuses in a probabilitistc framework both observations from the camera with predictions from an IMU. It can be coupled with GPS (for UAVs for example).
   About MSCKF (Multi-state constraint kalman filter): its particularity is that observations from the camera do not need to triangulate keypoints, so they don't need to be added to the state, reducing the memory/computation requirements. Also, only a sliding window of observations (2D pts) is kept for each keypoint, reducing the complexity.

6. *Suppose you have implemented a monocular Visual Odometry pipeline. How would change it to work as well with a stereo camera?*

   Answer.

   (a) You fuse it in a Kalman filter and don't change much, by taking observations of both cameras as observations with same processing and weights (only take care of offset).

   (b) Remove triangulation from normal pipeline and triangulate new keypoints each time using the 2 cameras, but would be pretty bad with its accuracy.

(c) Triangulate each time and refine the poses of the 3d pts until the interrsection of bundles (such as inSVO I think, and for event cameras algorithms) is precise enough (by computing/refining the uncertainty each time) until the 3d pts positions are very precise, in which case they're added to sparse map and used in odometry (P3P).

Application Questions

1. *Summarize the building blocks of a visual odometry (VO) or visual SLAM (VSLAM) algorithm.*

   Answer. In general the are three different VO algorithms:

   - 2D-2D: Compute Essential/Fundamental matrix between images and extract extrinsics, then concatenate relative movements.
   - 3D-3D: Determine the aligning transform of two 3D landmark sets, then concatenate.
   - 3D-2D: Use PnP to estimate absolute pose from known 3D landmarks and their most recent observation. No concatenation necessary!

   The detailed architecture of a 3D-2D VO is the following:

   (a) **Bootstrapping:**
      i. Initialize the algorithm by detecting and matching features in two frames. Use correspondences to triangulate a first set of landmarks.

   (b) **Continuous Operation** (Feature-based):
      i. Track or match features from previous image to get the correspondence of current 2D features to 3D landmarks.
      ii. Motion estimation by using RANSAC and Perspective from 3 Points (P3P). Use the 2D-3D correspondances.
      iii. Add additional candidate keypoints and landmarks over the next images. Triangulate new candidate landmarks from original observation and keyframe observation. To derermine if a landmark can be triangulated with sufficient accuracy, determine base line/bearing angle between the two observations. If this measure reaches some threshold, add candidate landmark to the landmarks used for motion estimation (PnP). Detect new features to track over the next images.

   (c) **(VSLAM) only**:
      i. Loop detection: use place recognition by BoW or other algorithms to update landmarks and transformations to align with loop closure.
      ii. Local Graph optimization: Refine graph by minimizing the reprojection error of landmarks over all frames in which they were observed. The argument over which is minimized are the camera poses.

2. *Augmented reality (AR) is a view of a physical scene augmented by computer-generated sensory inputs, such as data or graphics. Suppose you want to design an augmented reality system that super-imposes text labels to the image of real physical objects. Summarize the building blocks of an AR algorithm.*

   Answer.

   (a) Use SfM (Multiple View Geometry) to create 3D scene reconstruction.
      i. Option 1: Use features such as Harris, FAST or others and match them over subsequent frames. Use correspondences to extract extrinsics by PnP.

    ii. Option 2: Use checkerboard and extract homographies. Use homographies to compute extrinsics (position). This is essentially Zhang's camera calibration approach.

(b) Assign some set of landmarks to an object. which should be labelled. Possibly use markers (e.g. April Tags) to identify objects that should be augmented.

(c) For subsequent images: track keypoints that correspond to scene landmarks.

(d) Add text label output image and position every label to the centroids of all keypoints that belong to (some set of landmarks / some distinct object).

3. *Suppose that your task is to reconstruct an object from different views. How do you proceed?*

   Answer.

   (a) **Sparse reconstruction** (only features, no smooth surface):
   Do we know the relative position of the viewpoints?

      i. **YES** (stereo vision, sequential SfM): Get the 2D-2D correspondences by extracting and matching features. Use correspondences to determine relative camera poses and triangulate landmarks. Track keypoints over subsequent frames and use landmark-keypoint correspondences to determine the camera poses. Extract and triangulate additional landmarks if their base line / bearing angle allows sufficient accuracy.

      ii. **NO** (structure from motion, hierarchical SfM): Get 2D-2D correspondences by extracting and matching features. Estimate fundamental/essential matrix (8 point algorithm) and decompose E into R and t. Use correspondences and relative camera poses to triangulate landmarks.

   (b) **Dense reconstruction** (dense region of pixels, smooth surface, preferred)

      i. Choose reference image.

      ii. For every pixel in the reference image, calculate aggregated photometric error as a function of depth.

      iii. The photometric error of a pixel is the reprojection error that results if we reporoject the point to some other image (depending at what depth we set the point). We get it evaluating the (SSD,SAD,NCC) difference of a patch from all patches on the same epipole from another point of view.

      iv. Aggregated photometric error is the sum of error functions from all further images. The best depth estimate for a pixel in the reference image will minimize the aggregated photometric error function.

      v. Evaluate Disparity Space Image (DSI) by combining all error functions of all pixels in a 3D error funcion depending on (u,v and depth).

      vi. Global Regularization: Ensure smoothness by penalizing non-smooth areas and minimizing global energy.

4. *Building a panorama stitching application. Summarize the building blocks.*

   (a) **Feature Extraction:** find distinct features that are independent of changes in scale, rotation, illumination or viewpoint angle (e.g. corner or blobs).

(b) **Feature Description**: Establish descriptors for all features. (binar descriptors, census transform, intensity patches,...).

(c) **Feature Matching**: Use difference measure (SSD,SAD,NCC) to find close features.

(d) **Optional RANSAC:** Use RANSAC to filter outlier matches.

(e) **Feature Alignment**: Scale, rotate and warp images to align as many feature matches as possible.

5. *How would you design a mobile tourist app? The user points the phone in the direction of a landmark and the app displays tag with the name of it. How would you implement it?*

   (a) Indexing: use of landmarks to create image Bag Of Words index for each landmark.

   (b) Place recognition: compare camera image to image index and identify if any of the landmarks is present.

   (c) AR: If some landmarks is present, use the BoW inde to identify keypoints in the camera image that correspond to the landmark. Display tag with the name of the landmark located in the centroid of all keypoints identified to belong to some landmark.

6. *Assume that we have several images downloaded from flicker showing the two towers of Grossmunster. Since such images were uploaded by different persons they will have different camera parameters (intrinsic and extrinsic), different lighting, different resolutions and so on. If you were supposed to create a 3D model of Grossmunster, what kind of approach would you use? Can you get a dense 3D model or it will be a sparse one? Please explain the pipeline that you propose for this scenario.*

   (a) Hierarchical SfM:

      i. Identify clusters of nearby images by extracting and matching features.

      ii. Perform SfM for every cluster and build point cloud.

      iii. Merge clusters pairwise and refine poses and structure by minimiting the reprojection errors of landmarks across different viewpoints.

7. *Assume that you move around a statue with a camera and take pictures in a way that the statue is not far from the camera and always completely visible in the image. If you were supposed to find out where the pictures were taken, what would you do with the images? What kind of approach would you use? Since the camera motion is around the statue, the images contain different parts of the statue. How do you deal with this problem?*

   (a) Feature matching across sequential frames.

   (b) Estimate relative motion between frames using 2D-2D correspondences (8 point, 5 point algorithms) and extracting R and t from the Fundamental matrix (uncalibrated!) or the Essential matrix (calibrated!).

   (c) Optional: refine poses when a loop is detected and features from the first image are detected.

8. *Suppose that you have two robots exploring an environment, explain how the robots should localize themselves and each other with respect to the environment? What are the alternative solutions?*

   (a) **Localization:**

      i. Monocular Visual SLAM or Stereo Camera Visual SLAM for drift resistant odometry and map building (to prevent drift).

      ii. IMU for accurate localization and to add absolute scale to the localization.

      iii. GPS to improve global accuracy.

      iv. Laser triangulation sensor for map building (LIDAR)

      v. Combine multiple of these options and use a filter to get the final localization (e.g. Kalman Filter).

   (b) **Communication**

      i. Wirelessly share built maps and current position.

      ii. Possibly visual tracking and recognition of close robots using markers or other templates.

      iii. Possibly measurement of distance and heading of close robots by radio transmission (measure time and signal direction).

5 Mini Project Exam Preparation