# Lecture Summary

## Lecture 01: Introduction

**Definition 1. Computer Vision**: Automatic extraction of meaningful information from images and videos.

- Semantic information: meaning of objects.
- Geometric information: shapes.

### Vision in Humans

Vision is the most powerful sense:

- Retina $\approx 1000mm^2$.
- Contains 130 million **photoreceptors**
- 3GBytes/$s$ information flow, would need 500 Megapixel camera (8 Megapixel and range 15 degrees)

Why hard?

- only numbers, viewpoint variations, illumination challenges, motion, intra-class variations, scale and shape ambiguities

**Origin:** L.G. Roberts, MIT, 1963 , Solids Perception.

### Visual Odometry (VO)

**Definition 2.** 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

#### Why VO?

- $\neq$ Wheel Odometry, VO **not affected** by wheel slippage and adverge conditions in general.
- More accurate. Relative position error 0.1-2 %.
- Crucial for flying, walking and underwater robots.

#### Assumptions

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

**History**

- 1980: NASA, Moraveck, Mars Rovers, sliding camera.
- 1980-2000: NASA: mission to mars.
- 2004: David Nister: Visual Odometry paper.

**VO vs VSLAM vs SFM**

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

**Structure From Motion (SFM)**: more general than VO, tackles the problem of 3D reconstruction and 6DOF pose estimation from **unordered image sets**.
$\rightarrow$ VO focuses on estimating 3D motion **sequentially** and in **real time**.
**Visual Simultaneous Localization And Mapping (VSLAM)**:

- 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.

**Working Principle:**

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

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

2. Concatenate them to recover full trajectory

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

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

**Direct Image Alignment**

$\rightarrow$ The whole problem is a **minimization** of the **per-pixel intensity difference**

- Dense: $\approx$ 300'000 pixels.
- Semi-Dense: $\approx$ 10'000 pixels.
- Sparse: $\approx$ 2'000 pixels. (100-200 features x 4x4 patch).

**VO Flowchart**

VO computes the camera path incremental, pose per pose

1. Image sequence,
2. Feature detection,
3. Feature matching (tracking)
4. Motion estimation (2D-2D,3D-3D,3D,2D),
5. Local optimization.

# Lecture 02: Image Formation

## How to Form an Image

Placing a film in front of an object and illuminating it, the light is then reflected on the film. Not **reasonable** image! The rays don't converge in the same point, unsharp, blur.

### Pinhole Camera

Adding a barrier with a pinhole → camera obscura:

- Opening is the **aperture**.
- Reduces blurring.
- **ideal pinhole**: only one ray of light reaches each point on the film. Bigger aperture, blurry image.

Why not as small as possible? Diffraction effects as we near the wavelenth and less light gets through.

### Converging Lens

- Rays passing through the **Optical Center** are not deviated.
- All rays parallel to the **Optical Axis** converge at the **Focal point**

Using similar triangles and Figure 1 one gets

$$\begin{aligned} \frac{B}{A} &= \frac{e}{z} \text{ and} \\ \frac{B}{A} &= \frac{e-f}{f} = \frac{e}{f} - 1. \end{aligned} \tag{3}$$

Toghether we get the **thin lens equation**.

$$\frac{e}{f} - 1 = \frac{e}{z} \tag{4}$$

*Remark.*

- Any object point satisfying this equation is in **focus**.
- $z >> f$:
$$\frac{1}{f} = \frac{1}{e} \Rightarrow f \approx e. \tag{5}$$
→ we need to adjust the image plane such that objects at infinity are in focus, namely
$$\frac{h'}{h} = \frac{f}{z} \Rightarrow h' = \frac{f}{z}h. \tag{6}$$
The dependence of the apparent size of the object on its depth is known as **perspective**.

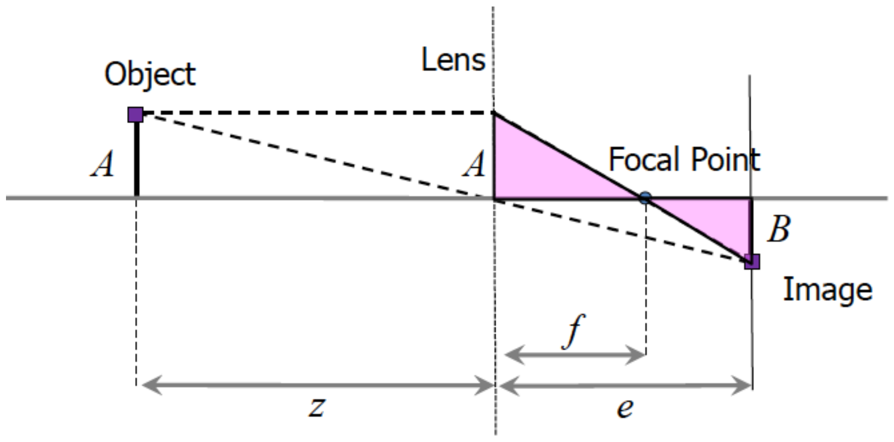


Figure 1: Thin lens

### In Focus and Blur Circle

- 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

$$R = \frac{L\delta}{2e} \tag{7}$$

$\rightarrow$ a minimal pinhole gives minimal $R$ and
$\rightarrow$ $R$ should remain smaller than image resolution.

### Projective Geometry

- 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**.

## Other Parameters

### Focus and Depth of Field

- DOF is the distance between the nearest and farthest objects in a scene that appear acceptably sharp.
- Decrease in sharpness is gradual on each side of the focused distance.
- Smaller aperture increases the range in which the object appears in focus but reduces the light.
- As $f$ gets smaller: **wide angle** image.
- As $f$ gets bigger: **narrow angle** image.

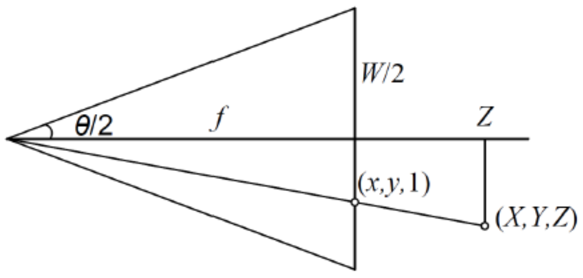


Figure 2: Scheme for angles.

With Figure 2, one can compute the **field of view**

$$\tan\left(\frac{\theta}{2}\right) = \frac{W}{2f} \Rightarrow f = \frac{W}{2}\left[\tan\left(\frac{\theta}{2}\right)\right]^{-1} \tag{8}$$

$\rightarrow$ smaller FOV = larger focal length.

## Digital Camera

Instead of using a film we use a sensor array and we convert informations in numbers (e.g. $[0, 255]$ for 8 bytes), as in Figure 3.

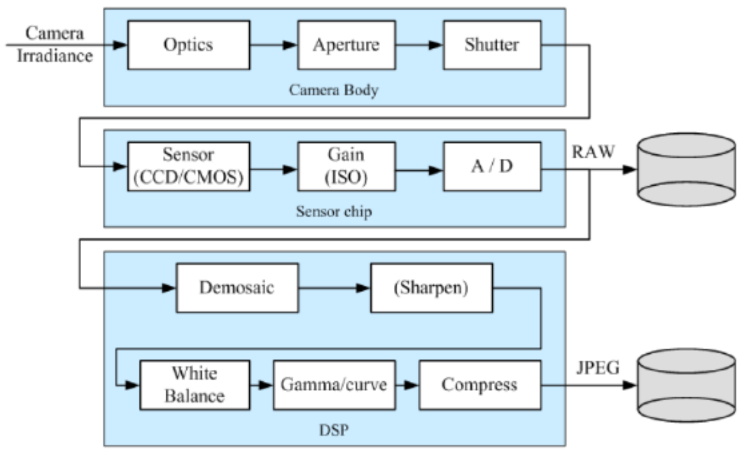


Figure 3: Procedure digital cameras.

### Color Sensing

- Byer pattern (1976) places green filters over half of the sensors and red and blue over remaining ones. Humans are more sensintive to high frequency detail in luminance than chrominance.
- Estimate missing components from neighboring values: **demosaicing**

## Perspective Camera Model

The procedure reads

1. Change of coordinates from 3D world to adapted frame.
2. Projection from the camera frame to the image plane.
3. Change in pixel.

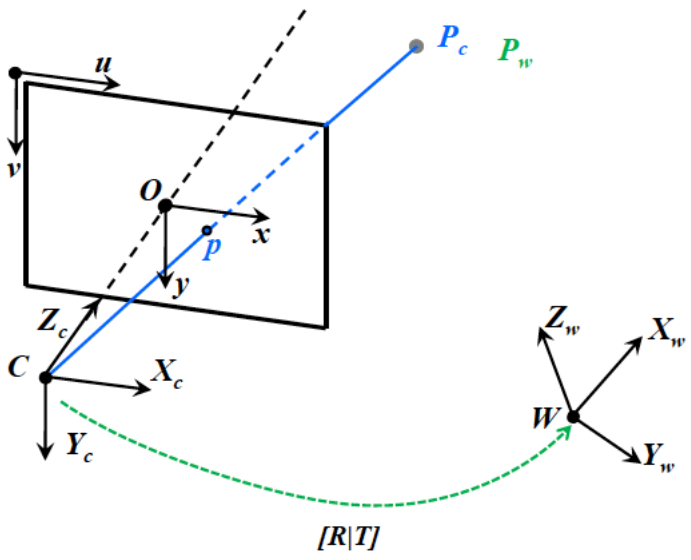


Figure 4: Frames.

### Perspective Projection

We use the notation from euclidean to homogeneous:

$$\underbrace{\begin{pmatrix} u \\ v \\ w \end{pmatrix}}_{Homog.} \rightarrow \underbrace{\begin{pmatrix} u/w \\ v/w \\ 1 \end{pmatrix}}_{Eucl.} \tag{9}$$

From similar triangles and Figure 5 one gets

$$\begin{aligned} \frac{x}{f} = \frac{X_c}{Z_c} \Rightarrow x = \frac{fX_c}{Z_c} \\ \frac{y}{f} = \frac{Y_c}{Z_c} \Rightarrow y = \frac{fY_c}{Z_c} \end{aligned} \tag{10}$$

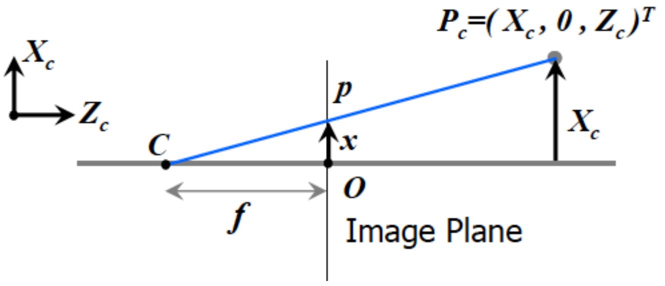


Figure 5: Figure for perspective projection.

### Pixel Coordinates

From local image plane coords $(x, y)$ to the pixel coords $(u, v)$:

$$\begin{aligned} u &= u_0 + k_u x \Rightarrow u = u_0 + \frac{k_u f X_c}{Z_c} \\ v &= v_0 + k_u x \Rightarrow v = v_0 + \frac{k_v f X_c}{Z_c}, \end{aligned} \tag{11}$$

which expressed in matrix form reads

$$\begin{pmatrix} \lambda u \\ \lambda v \\ \lambda \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = K \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}, \tag{12}$$

with $\alpha_u, v$ focal lengths and $K$ **calibration matrix**. At the end, it holds

$$\lambda \cdot \begin{pmatrix} u \\ v \\ v \end{pmatrix} = \underbrace{(K)}_{3\times 3,\ intrinsic} \cdot \underbrace{(\mathbb{I}_{3\times 3}|0)}_{3\times 4} \cdot \underbrace{\begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix}}_{4\times 4,\ extrinsic} \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix}. \tag{13}$$

## Lens Distortion

### Radial Distortion

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

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = (1 + k_1 r^2) \cdot \begin{pmatrix} u - u_0 \\ v - v_0 \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}, \tag{14}$$

with

$$r^2 = (u - u_0)^2 + (v - v_0)^2. \tag{15}$$

# Lecture 03: Image Formation 2

## Pose determination from $n$ Points (PnP) Problem

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: $\infty$ solutions.
- Given 2 points: $\infty$ *bounded* solutions.
- Given 3 **non collinear** points: Finitely many (up to 4) solutions.
- Given 4 points: unique solution.

With 3 points one has the Carnot's theorem

$$s_{1,2,3}^2 = L_{B,A,A}^2 + L_{C,C,B}^2 - 2L_{B,A,A}L_{C,C,B}\cos(\theta_{BC,AC,AB}) \tag{16}$$

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

$$G_0 + G_1x + G_2x^2 + G_3x^3 + G_4x^4 = 0. \tag{17}$$

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

## Camera Calibration

Determine **intrinsic** and **extrinsic** parameters of the camera model.
*Tsai, 1987*: Measure the 3D position of more than 6 points on a 3D calibration target and the 2D coordinates of theis projection.

**Direct Linear Transform**

$$\begin{aligned}
\text{image point} = \tilde{p} = \begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} &= \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K[R|T] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} \alpha_u & 0 & 0 \\ 0 & \alpha_v & \nu_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \\
\textit{assuming indep. elements} \qquad &= \underbrace{\begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix}}_{M} \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} m_1^T \\ m_2^T \\ m_3^T \end{pmatrix} \cdot \underbrace{\begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}}_{P}.
\end{aligned} \tag{18}$$

It follows

$$\begin{aligned}
u &= \frac{\tilde{u}}{\tilde{w}} = \frac{m_1^T \cdot P}{m_3^T \cdot P} \\
v &= \frac{\tilde{v}}{\tilde{w}} = \frac{m_2^T \cdot P}{m_3^T \cdot P}
\end{aligned} \tag{19}$$

and hence

$$\begin{aligned}
(m_1^T - u_i m_3^T) \cdot P_i &= 0 \\
(m_2^T - v_i m_3^T) \cdot P_i &= 0.
\end{aligned} \tag{20}$$

Rearranging the terms you have

$$\begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{21}$$

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

$$Q \cdot M = 0. \tag{22}$$

**Minimal Solution:**

- Rank 11 to have unique non-trivial 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 $||QM||^2 \rightarrow$ SVD. The solution is the eigenvector corresponding to the smallest eigenvalue of $Q^TQ$.

**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

$$M = K(R|T). \tag{23}$$

*Remark.*

- We are not enforcing orthogonality of $R$.
- QR factorization of $M$, whith $R$ (orth.) and $T$ (upper triangular matrix).

### Tsai Method

1. Edge detection.
2. Straight line fitting to the detected edges.
3. Intersecting the lines to obtain the image corners (¡0.1 pixels accuracy).
4. Use more than 6 points (more than 20) and **not** all on a plane.

Originally pixels were not squared (parallelogramms, skew, no rectangle). Most cameras today are well manufactured: $\frac{\alpha_u}{\alpha_v} = 1$ and $K_{12} = 0$.
**Residual**: Average reprojection error, computed as the distance (in pixels) between the observed point and the camera-reprojected 3D point. Accuracy of calibration.
What if $K$ is known? Nothing changes!

### Calibration from Planar Grids

*Zhang, Microsoft* Use a planar grid (chessboard) and a few image of it at different orientations. Setting $Z_w = 0$ we get

$$\begin{aligned}
\begin{pmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{pmatrix} &= \begin{pmatrix}\alpha_u & 0 & 0\\ 0 & \alpha_v & \nu_0\\ 0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}r_{11} & r_{12} & r_{13} & t_1\\ r_{21} & r_{22} & r_{23} & t_2\\ r_{31} & r_{32} & r_{33} & t_3\end{pmatrix} \cdot \begin{pmatrix}X_w\\ Y_w\\ 0\\ 1\end{pmatrix}\\
&= \begin{pmatrix}\alpha_u & 0 & 0\\ 0 & \alpha_v & \nu_0\\ 0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}r_{11} & r_{12} & t_1\\ r_{21} & r_{22} & t_2\\ r_{31} & r_{32} & t_3\end{pmatrix} \cdot \begin{pmatrix}X_w\\ Y_w\\ 1\end{pmatrix}\\
&= \begin{pmatrix}h_1^T\\ h_2^T\\ h_3^T\end{pmatrix} \cdot \begin{pmatrix}X_w\\ Y_w\\ 1\end{pmatrix}.
\end{aligned} \tag{24}$$

Hence, one more time

$$\begin{aligned} u &= \frac{\tilde{u}}{\tilde{w}} = \frac{h_1^T \cdot P}{h_3^T \cdot P} \\ v &= \frac{\tilde{v}}{\tilde{w}} = \frac{h_2^T \cdot P}{h_3^T \cdot P} \end{aligned} \tag{25}$$

and hence

$$\begin{aligned} (h_1^T - u_i h_3^T) \cdot P_i &= 0 \\ (h_2^T - v_i h_3^T) \cdot P_i &= 0. \end{aligned} \tag{26}$$

Rearranging the terms you have

$$\begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \end{pmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{27}$$

For $n$ points we have a big $2n \times 9$ matrix $Q$. The problem hence reads

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

**Minimal Solution:**

- Rank 8 to have unique non-trivial solution $H$ ($Q$ known!).
- Each point correspondence provides 2 independent equations.
- 4 **non-collinear** points are needed.

**Overdetermined Solution:**

- More than 4 points.
- Minimize $||QM||^2 \rightarrow$ SVD. The solution is the eigenvector corresponding to the smallest eigenvalue of $Q^T Q$. We can decompose as before.

This transformation is called **Homography**. Applications are

- Augmented reality
- Beacon-based localization.

**DLT vs. PnP**

- If the camera is calibrated, only $R$ and $T$ need to be determined. Pnp leads to smaller error.

## Non Conventional Camera Models

### Omnidirectional Cameras

- Wide FOV dioptric cameras (e.g. fisheye (180)).
- Catadioptric cameras (e.g. mirrors (>180)).
  - Mirror: **central**, mirror (surface of revolution of a conic), single effective view point.
  - Perspetive: hyperbola+perspective / parabola+orthographic lens.
- Polydioptric cameras (e.g. multiple overlapping cameras).

If the camera is central, we can unwarp parts of omnidirectional image into perspective. We can transform image points in the unit sphere. We can apply algorithms for perspective geometry. Perspective and omnidirectional model are equal!

# Lecture 04: Image Filtering

*Filtering*: Accepting or rejecting certain frequency components.

- **low-pass** filter smooths an image.
- **high-pass** filter retains the edges of an image.

## Low-pass Filtering

We want to reduce noise! There different typesof noise:

- **Salt and pepper noise**: randum occurences of black and white pixels.
- **Impulse noise**: random occurences of white pixels.
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution.

### Moving Average

Replaces each pixel with an average of all the values in its neighborhood.

- Pixels like neighbors.
- Noise process independent from pixel to pixel.

### Weighted Moving Average

Can add weights to moving average

- Uniform weights.
- Non-uniform weights.

Nothing else than convolution!

### Convolution

One of the sequences is flipped before sliding over the other. Linearity, associativity, commutativity. Notation $f * g$. In 2D:

$$\begin{aligned} G[i,j] &= H * F \\ &= \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]. \end{aligned} \tag{29}$$

In other words: replacing each pixel with a linear combination of its neighbors. The filter $H$ is also called kernel or mask.

### Gaussian Filter

What if we want the closest pixels to have a higher influence on the output?

$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}. \tag{30}$$

What parameter matter?

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

### Boundary Issues

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

### Median Filter

Linear smoothing filters do not alleviate salt and pepper noise! Non-linear filter. Removes spikes: good for impulse and salt and pepper noise.
Computes the median value and replaces the high value with that.

- Preserves sharp transitions.
- Removes small brightness variations.

## High-pass Filtering (edge detection)

We want an idealized line drawing. The edge contours in the image correspond to important scene contours. Edges are nothing else than sharp intensity changes. Images can be expressed as functions $f(x,y)$. Edges correspond to extrema of derivative.

### Differentiation and Convolution

For discrete data, it holds

$$\frac{\mathrm{d}f(x,y)}{\mathrm{d}x} \approx \frac{f(x+1,y)-f(x,y)}{1}. \tag{31}$$

Partial derivatives of an image are in $x$ (-1,1), in $y$ $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$. Other finite differences methods are the

- **Prewitt Filter**

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}, \qquad G_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \tag{32}$$

- **Sobel Filter**

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \qquad G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{33}$$

The **Gradient** of an image if given by

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{pmatrix} \tag{34}$$

The **gradient direction** is given as

$$\Theta = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right). \tag{35}$$

The **edge strength** is given by

$$||\nabla f|| = \sqrt{(\frac{\partial f}{\partial y})^2 + (\frac{\partial f}{\partial y})^2}. \tag{36}$$

### Handling Noise

If we differentiate a noisy signal, we get infinite many peaks. Solutions are

- first **smooth** the signal (with a convolution with $h$). Then differentiate.
- Combining the two, convolute with $(\frac{\mathrm{d}}{\mathrm{d}x}x) * f$.

### Laplacian of a Gaussian

Consider

$$\frac{\partial^2}{\partial x^2}(h * f) \tag{37}$$

Where is the edge? Zero-crossin of bottom graph.

### Summary

- Smoothing filters:
  - Has positive values.
  - Sums to 1 $\rightarrow$ preserves brightness of constant regions.
  - Removes high frequency components.
- Derivative Filters
  - Has opposite signs, used to get high response in regions of high constrast.
  - Sums to 0 $\rightarrow$ no response in constant regions.
  - highlights high frequency components.

**The Canny Edge-Detection Algorithm**

- We compute the gradient of smoothed image in both directions.
- We discard pixels whose gradient magnitude is below a certain threshold.
- **Non-maximal suppression**: local maxima along gradient diretion.

# Lecture 05: Feature Detection 1

Goal: reduce amount of data to process in later stages, discard redoundancy to preserve only what is useful (lower bandwidth and memory storage). In general

- Edge detection.
- Template matcing.
- Keypoint detection.

## Filters for Template Matching

We want to find locations in an image that are similar to a **template**. If we look at filters as templates, we can use **correlation** to detect these locations. What if the template is not identical to the object we want to detect? This works only if

- Scale,
- orientation,
- illumination,
- appearance of the template and the object are similar.

What about the objects in the background?

### Correlation as Scalar Product

We consider images $H$ and $F$ as vectors an express the correlation between them as

$$\langle H, F\rangle = ||H|| \cdot ||F|| \cdot \cos(\theta). \tag{38}$$

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

$$\begin{aligned}\cos(\theta) &= \frac{\langle H, F\rangle}{||H|| \cdot ||F||}\\ &= \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)F(u,v)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)^2} + \sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F(u,v)^2}}.\end{aligned} \tag{39}$$

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

$$SAD = \sum_{u=-k}^{k}\sum_{v=-k}^{k} |H(u,v) - F(u,v)|, \tag{40}$$

the **Sum of Squared Differences (SSD)**

$$\sum_{u=-k}^{k}\sum_{v=-k}^{k} (H(u,v) - F(u,v))^2. \tag{41}$$

The **normalized cross correlation (NCC)** takes values between -1 and 1, 1 equals identical.
To account for the difference in mean of the two images (caused principally by illumination changes), we substract the mean value of each image:

- **Zero-mean Sum of Absolute Differences (ZSAD)**

$$ZSAD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} |(H(u,v) - \mu_H) - (F(u,v) - \mu_F)|. \tag{42}$$

- **Zero-mean Sum of Squared Differences (ZSSD)**

$$\sum_{u=-k}^{k} \sum_{v=-k}^{k} ((H(u,v) - \mu_H) - (F(u,v) - \mu_F))^2. \tag{43}$$

- **Zero-mean ormalized Cross Correlation (ZNCC)**

$$ZNCC = \frac{\sum_{u=-k}^{k} \sum_{v=-k}^{k} (H(u,v) - \mu_H) \cdot (F(u,v) - \mu_F)}{\sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} (H(u,v) - \mu_H)^2} + \sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} (F(u,v) - \mu_F)^2}}. \tag{44}$$

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

*Remark.* ZNCC is **invariant** to affine intensity changes.

### Census Transform

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.

### Point-feature Extraction and Matching

Keypoint extraction is the key ingredient of motion estimation! Furthermore, used for panorama stitching, object recognition, 3D reconstruction, place recognition, google images.
Why problematic? We need to align images! How? Detect point features in both images and find corresponding pairs to align them. Two big problem arise

- Problem 1: Detect the **same** points **independently** in both images. No chance to match, need **repeatable** feature detector.

- Problem 2: for each point, identify its correct correspondence. Need **reliable**and **distinctive** feature descriptor. Robust to **geometric** and **illumination** changes.

**Geometric changes:** rotation, scale and viewpoint.
**Illumination changes:** Affine illumination changes

$$I'(x, y) = \alpha I(x, y) + \beta. \tag{45}$$

**Invariant local features:** Subset of local feature types designed to be invariant to common geometric and photometric transformations. In general

1. Detect distinctive interest points,
2. extract invariant descriptors.

**Distinctive features s.t. repeatable?** Some features are better than others (angles, not uniform color,...).
**Corners:** a corner is defined as the intersection of one or more edges. It has **high localization accuracy**. Less distinctive than a **blob**.
**Blob:** is any other image pattern which is not a corner, that differs significantly from its neighbors in intensity and texture. This has less localization accuracy, but better for place recognition because more distinctive than a corner.

### Corner Detection

In the region around a corner, the image gradient has two or more dominant directions. Corners are repeatable and distinctive.

### The Moravec Corner Detector (1980)

We can easily recognize the point by looking through a small window: by shifting the window, one can give large change in intensity.

- **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.

*Sums of squares of differences of pixels adjacent in each of four directions (horizontal, vertical and two diagonals) over each window are calculated, and the window's interest measure is the minimum of these four sums.*[Moravec,80]

### The Harris Corner Detector (1988)

Implements Moravec corner detector without physically shifting the window and hence just by looking at the patch itself: using **differential calculus**.

**Implementation:** We consider the reference path centered at $(x, y)$ and the shifted window centered at $(x + \Delta x, y + \Delta y)$. The patch has size $P$. We compute

$$SSD(\Delta x, \Delta y) = \sum_{x,y \in P} \left(I(x, y) - I(x + \Delta x, y + \Delta y)\right)^2. \tag{46}$$

We define

$$I_x = \frac{\partial I(x, y)}{\partial x}, \quad I_y = \frac{\partial I(x, y)}{\partial y}, \tag{47}$$

and approximate with first order Taylor expansion:

$$\begin{aligned} I(x + \Delta x, y + \Delta y) &\approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y \\ \Rightarrow SSD(\Delta x, \Delta y) &\approx \sum_{x,y \in P} \left(I_x(x, y)\Delta x + I_y(x, y)\Delta y\right)^2 \end{aligned} \tag{48}$$

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

$$SSD(\Delta x, \Delta y) \approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \cdot \underbrace{\sum_{x,y \in P} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}}_{M} \cdot \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}. \tag{49}$$

Let's analyze some special cases:

- 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: $M = \begin{pmatrix} \cos(45) & -\sin(45) \\ \sin(45) & \cos(45) \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \cdot \begin{pmatrix} \cos(45) & \sin(45) \\ -\sin(45) & \cos(45) \end{pmatrix}$

What if the corner is not aligned with the image axis? The general case has $M$ symmetric, which can always be decomposed into

$$M = R^{-1} \cdot \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \cdot R. \tag{50}$$

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

*Remark.* Small ellipses denote flat region, big ones a corner!

**Interpreting the eigenvalues** 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 detector**

$$R = \min(\lambda_1, \lambda_2) > \text{threshold}. \tag{51}$$

- 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.

**Problem:** The eigenvalues are expensive to compute: *Harris and Stephens* suggested to use the fact

$$R = \lambda_1 \cdot \lambda_2 - k(\lambda_1 + \lambda_2)^2 = det(M) - k \cdot \text{trace}^2(M), \quad k \in (0.04, 0.15) \tag{52}$$

**Algorithm:**

(I) Compute derivatives in $x$ and $y$ directions e.g. with *Sobel filter*.

(II) Compute $I_x^2, I_y^2, I_xI_y$.

(III) Convolve $I_x^2, I_y^2, I_xI_y$, with a box filter to get the sums of each element, which are the entries of the matrix $M$. (optionally use Gaussian filter instead of box filter to give more imoportance to central pixels).

(IV) Compute Harris Corner Measure $R$ (with Shi-tomasi or Harris).

(V) Find points with large corner response ($R >$ threshold).

(VI) Take the points of local maxima of $R$.

Harris vs. Shi-Tomasi???
**Repeatability** Can it re-detect the same image patches (Harris corners) when the image exhibits changes?

- Corner response $R$ is **invariant to image rotation**. Shape (eigenvalues) remains the same.
- Not invariant to **image scale**. Scaling the image by $\times 2$ results in 18 % of correspondences get matched.

# Lecture 06: Point Feature Detection 2

## Scale Changes

A possible solution is to rescale the patch, i.e. bring it to the canonical scale. The problem by scale search is that it is scale consuming: we need to do it individually for all patches in one image. In fact, the complexity would be $(NM)^2$ (assuming $N$ features per image and $M$ scale levels for each image). $\Rightarrow$ a possible solution is to assign each feature its own scale.

### Automatic Scale Selection

- 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.
- For a point in one image, we can consider it is a function of region size.

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

- Good function: single and sharp peaks!
- If multiple peaks: assign **more** region sizes to have a unique feature. Blobs and corners are the ideal locations!

### Function:

Convolve image with kernel to identify sharp discontinuities:

$$f = \text{Kernel} * \text{Image} \tag{53}$$

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

$$\text{LoF} = \nabla^2 G(x,y) = \frac{\partial G(x,y)}{\partial x^2} + \frac{\partial G(x,y)}{\partial y^2}, \tag{54}$$

Then, the correct scale is found as local maxima across consecutive smoothed images. This should be done for **several**s region sizes (15).

## Feature Descriptors

We already know how to detect points, but how can we describe them for matching?

- **Simplest Descriptor**: **Intensity** values within a squared patch or gradient histogram.
- **Census transform** or **Histograms of Oriented Gradients**.

Then, descriptor mathicng can be done using **Hamming Distance (Census)** or **(Z)SSD,(Z)SAD, (Z)NCC**. We would like to fame the same features regardless of the **transformation** that is applied to them: most feature methods are designed to be invariant of

- 2D translation,
- 2D rotation,
- scale.

Some of them can also handle

- Small view point invariance.
- Linear illumination changes.

**How to achieve Invariance?**

**Step 1: Re-scaling and De-rotation:**

- Find the **correct scale** using LoG operator.
- **Rescale** the patch to a default size
- Find the **local orientation** (e.g. with dominant direction, Harris eigenvectors)
- **De-rotate** the patch.

In order to de-rotate the patch, one uses **patch-warping**:

### Patch Warping

1. Start with **empty** canonical patch (all pixels set to 0).
2. For each $(x, y)$ in the empty patch, apply the **warping function** $W(x, y)$ to compute the corresponding position in the detected image. It will be in floating point and will fall between the image pixels.
3. Interpolate the intensity values of the 4 closest pixels in the detected image with
   - Nearest neighbor,
   - Bilinear transform

### Example 1: Rotational Warping

Counterclockwise rotation:

$$\begin{aligned} x' &= x\cos(\theta) - y\sin(\theta) \\ y' &= x\sin(\theta) + y\cos(\theta) \end{aligned} \tag{55}$$

### Bilinear Interpolation

It is an extension of the linear interpolation, for interpolating functions of two variables on a rectilinear 2D grid. The key idea is to perform linear interpolation in one direction and then, again, in the other direction. We have that

$$I(x, y) = I(0, 0)\cdot(1-x)\cdot(1-y) + I(0, 1)\cdot(1-x)\cdot y + I(1, 0)\cdot x\cdot(1-y) + I(1, 1)\cdot xy \tag{56}$$

### Example 2: Affine Warping

To achieve **slight view-point invariance**:

- The second moment matrix $M$ can be used to identify the two directions of fastest and slowest change of intensity around the feature.
- Out of these two directions, an elliptic patch is extracted at the scale computed with the LoG operator.
- The region inside the ellipse is normalized to a circular one.

There are however disadvantages:

- If not warped patches, very small errors in rotation, scale and view-point will affect matching score significantly.
- Computationally expensive (need to unwarp every patch)

A better solution is HOGs

### Histogram of Oriented Gradients

- Compute a histogram of orientations of intensity gradients.
- Peaks in histogram are dominant orientations.
- **Keypoint orientation= histogram peak**. If there are multiple candidate peaks, construct a different keypoint for each such orientation.
- **Rotate patch** according to this angle: this puts the patches into a canonical form.

## Scale Invariant Feature Transform (SIFT) Descriptor

Descriptor computation:

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

### Intensity Normalization

The descriptor vector $v$ is then normalized such that its $l_2$ norm is 1:

$$\bar{v} = \frac{v}{\sqrt{\sum_i^n v_i^2}}. \tag{57}$$

*Remark.* This guarantees that the descriptor is invariant to linear illumination changes. This was already invariant to additive illumination because it is based on gradients.

**SIFT matching robustness**

- Can handle changes in viewpoint (up to 60 degree out-of-plane rotation).
- Can handle significant changes in illumination (low to bright scenes).
- Expensive: 10fps.

In order to reduce the computational cost, one can use difference of Gaussian instead of Lapiacian:

$$\text{LOG} \approx \text{DOG} = G_{k\sigma}(x,y) - G_{\sigma}(x,y) \tag{58}$$

**SIFT Detector**

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

- Detect maxima and minima of difference-of-Gaussian in scale space.
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below.
- For each max and min found, the output is the location and the scale.

**Implementation:**

1. 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).
2. Adjacent image scales are then subtracted to produce the difference-of-Gaussian (DoG) images.

**Summary**

- An approach to detect and describe regions of interest in an image.
- SIFT detector = DoG detector.
- SIFT features are reasonably invariant to changes in rotation, scaling, and changes in viewpoint (up to 60deg) and illumination.
- Real time but still **slow** (10Hz on an i7 laptop).

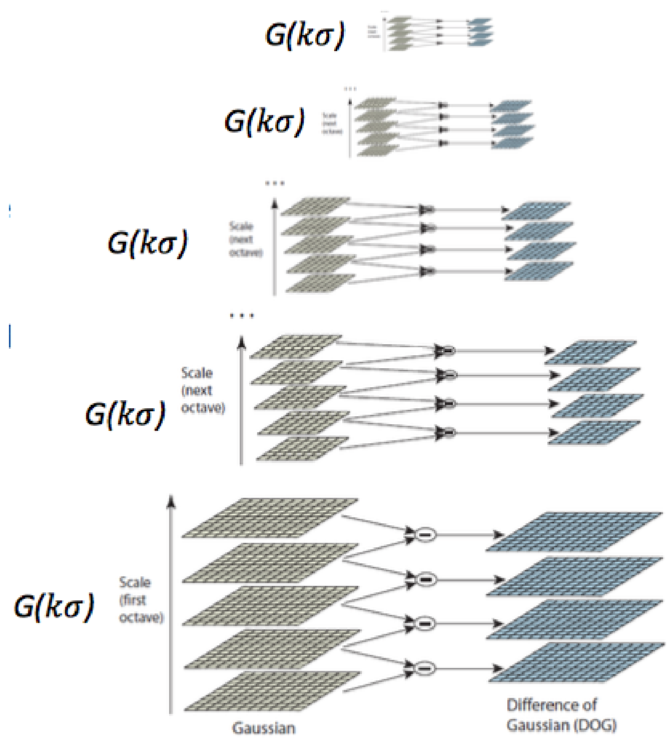


Figure 6: Difference of Gaussian.

The **repeatability** can be expressed as

$$\frac{\text{number of correspondences detected}}{\text{number correspondences present}} \tag{59}$$

The highest repeatability is obtained when sampling 3 scales per octave.
**Influence of Number of Orientation and NUmber of Sub-Patches**: Single orientation histogram is poor at discriminating, but the results continue to improve up to a 4x4 array of histograms with 8 orientations.

- **Descriptor**: 4x4x8 = 128-element 1D vector.
- **Location**: 2D vector.
- **Scale** of the patch. 1 scalar value.
- **Orientation** (angle of the patch). 1 scalar value.

### SIFT for object recognition

Can be simply implemented by returning as best object match the one with the largest number of correspondences with the template (object to detect). 4 or 5 point RANSAC can be used to remove outliers.

## Feature Matching

Given a feature $I_1$, how to find the best match in $I_2$?

1. Define distance function that compares two descriptors ((Z)SSD,SAD,NCC, or Hamming distance for binary descriptors (e.g. Census, BRIEF, BRISK).
2. **Brute-force matching**:
    - Test all the features in $I_2$.
    - Take the one at min distance.

**Issues with closest descriptor:** Can give good scores to very ambiguous (bad) matches (curse of dimensionality). A better approach would be to compute the ratio of distances between the *first*and *second*match:

$$\frac{d(f_1)}{d(f_2)} < \text{Threshold (usually 0.8)}, \tag{60}$$

where

$$\begin{aligned} &d(f_1) \text{ is the distance of the closest neighbor} \\ &d(f_2) \text{ is the distance of the second closest neighbor} \end{aligned} \tag{61}$$

**Explanation for distance ratio**: In SIFT, the nerest neighbor is defined as the keypoint with minimum euclidean distance. However, many features from an image 1 may not have any correct match in image 2 because they arise from background cluttere or were *not detected at all* in the image 1. An effective measure is otained by comparing the distance of the closest **neighbor** to that of the second closest neighbor. Why? 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.
**Why 0.8?**

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

**SURF (Speeded Up Robust Features)**

- Based on ideas similar to SIFT.
- Approximated computation for detection and descriptor.
- Results are comparable with SIFT but
    - Faster and
    - Shorter descriptors.

**FAST detector (Features 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 $intensity(C) + threshold$.
- Typically tests for 9 contiguous pixels in a 16 pixel circumference.
- **Very fast detector** (100 Mega pixel/second).

**BRIEF descriptor (Binary Robust Independent Elementary Features)**

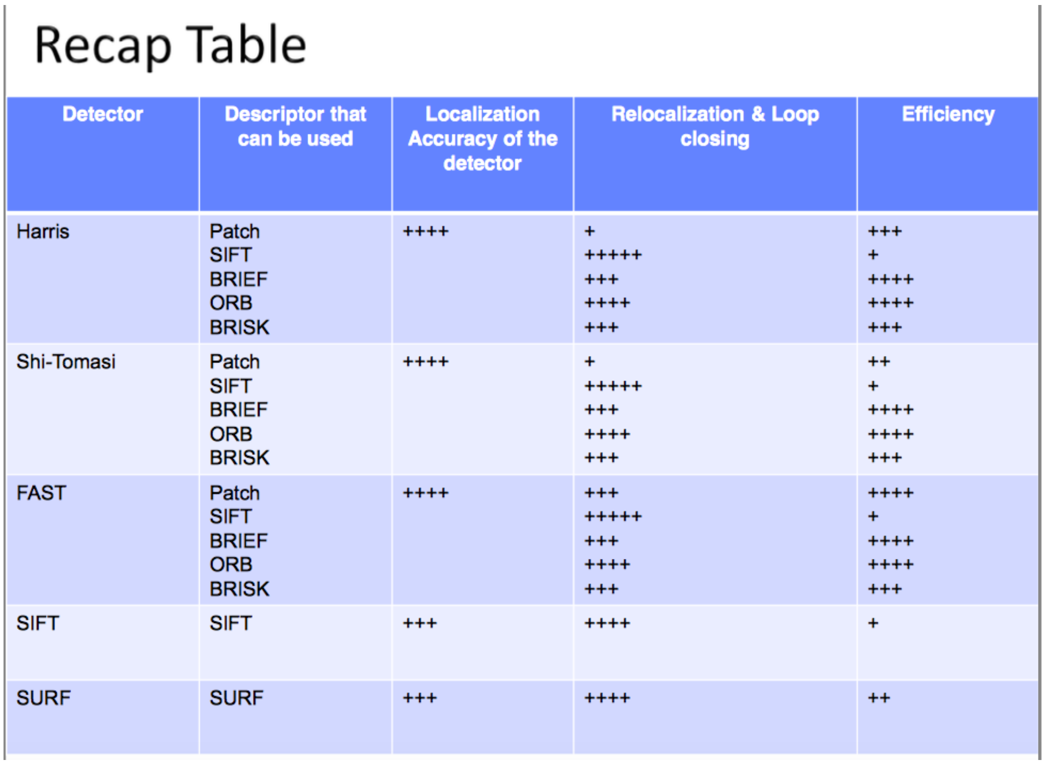
- Goal: high speed.
- **Binary** descriptor formation
  - Smooth image,
  - For each detected keypoint (e.g. with FAST) **sample** 256 intensity pairs $(p_1^i, p_2^i)$, $i = 1 - 256$ within a squared patch around keypoint.
  - Create an empty 256-element descriptor.
  - For each i-th pair:
    - if $I_{p_1^i} < I_{p_2^i}$, then set i-th bit of descriptor to 1.
    - else to 0.
- The **pattern is generated randomly** (or by ML) only once: then, same pattern is used for all patches.
- Pros: **Binary Descriptor** allows very fast Hamming distance matching: count the number of bits that are different in the descriptors matched.
- Cons: **Not scale/rotation invariant**.

**ORB descriptor (Oriented FAST and Rotated BRIEF)**

- Keypoint detector based on FAST.
- BRIEF descriptors are steered(?) according to keypoint orientation (to provide rotation invariance).
- Good binary features are learned by minimizing the correlation on a set of training patches.

**BRISK descriptor (Binary Robust Invariant Scalable Keypoints)**

- **Binary**: formed by pairwise intensity comparisons.
- **Pattern** defines intensity comparisons in the keypoint neighborhood.
- **Red circles**: size of the smoothing kernel applied.
- **Blue circles**: smoothed pixel values used.
- Compare short- and long-distance pairs for orientation assignment and descriptor formation.
- Detector and descriptor speed: circa **10 times faster than SURF**.
- Slower than BRIEF, but scale- and rotation- invariant.



## Recap Table

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

Figure 7: Recap for detectors and descriptors.

**Things to remember**

- Similarity metrics: NCC (ZNCC), SSD (ZSSD), SAD (ZSAD), Census Transform.
- Point feature detection:
  - Properties and invariance to transformations: **Challenges** are rotation, scale, view-point and illumination changes.
  - Extraction:

    - Moravec
    - Harris and Shi-Tomasi: rotation invariance.
  - Automatic Scale Detection
  - Descriptor
    - Intensity patches:
      - Canonical representation: how to make them invariant to transformations: rotation, scale, illumination and view point (affine).
    - Better solution: Histogram of oriented gradients: SIFT descriptor.
  - Matching:
    - (Z)SSD, SAD, NCC, Hamming distance (last one only for binary descriptors), ration first/second closest descriptor.
  - Depending on the task, you may want to trade off repeatability and robustness for speed: approximated solutions, combinations of efficient detectors and descriptors.
    - Fast corner detector,
    - Keypoint descriptors faster than SIFT: SURF, BRIEF, ORB, BRISK.

# Lecture 07: Multiple View Geometry 1

We have different problem statements:

- **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.

For a 2-view geometry, we can define the same problems as

- **Depth from stereo (stereo vision)**
  - **Assumption:** $K, T, R$ are known.
  - **Goal:** Recover the 3D structure from images.
- **2-view structure from motion**
  - **Assumption:** $K, T, R$ are unknown.
  - **Goal:** Recover simultaneously 3D scene structure and camera poses (up to scale) and intrinsic parameters from two different views of the scene.

## Depth from stereo

From a single camera, we can only compute the **ray** on which each image point lies. With a stereo camera (binocular), we can solve for the intersection of the rays and eventually recover the 3D structure.

### The human binocular system

**Stereopsys:** the brain allows us to see the left and right retinal images as a single 3D image. The images project on our retina up-side-down but our brains let us perceive them as straight. Radial distortion is removed. This process is also known as **rectification**. The distance of two seen images is called **disparity** (allows us to perceive the depth). An application of this concept are **stereograms**.

### Stereo Vision: basics

The basic principle behind stereo vision is **triangulation**.

- Gives reconstruction as intersection of two rays.
- Requires **camera pose (calibration) and point correspondence**.

There are basically two cases

1. Simplified case: identical cameras are **aligned**.
2. General case: different cameras are not **aligned**.

### Simplified Case

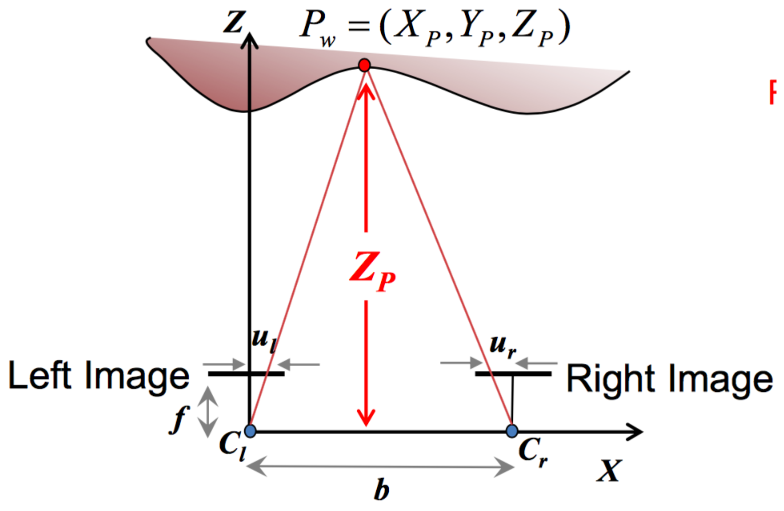


Figure 8: Simplified case for stereo vision.

The two cameras are identical, meaning that they have the same **focal length**, and are **aligned** with the $x$-axis. If we have a world point $P_w$, a distance from the axis to the point $Z_P$, a distance between the cameras $b$ and focal length $f$, we can use similar triangles from Figure 8 and get

$$\begin{aligned} \frac{f}{Z_P} &= \frac{u_l}{X_P} \\ \frac{f}{Z_P} &= \frac{-u_r}{b - X_P} \\ \Rightarrow Z_P &= \frac{b \cdot f}{u_l - u_r}. \end{aligned} \tag{62}$$

The difference $u_l - u_r$ is called **disparity**: difference in image location of the projection of a 3D point on two image planes. (QUESTIONS)
What is the optimal baseline?

- **Too small:**
  - Large depth error
- **Too large:**
  - Minimum measurable distance increases.
  - Difficult search problem for close objects.

### General Case: triangulation

In reality no cameras are identical and aligning both cameras on a horizontal axis is impossible. (WHY?). In order to be able to use a stereo camera, we need to compute

- The extrinsic parameters (relative rotation and translation)

- the intrinsic parameters (focal length, optical center, radial distortion of each camera).

In order to do this we use calibration methods (Tsai or homographies). How do we get the **relative pose?** Lines do not always interset in the 3D space: we want to minimize the error. For the two cameras we have

$$\tilde{p}_l = \lambda_l \cdot \begin{pmatrix} u_l \\ v_l \\ 1 \end{pmatrix} = K_l \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix}, \quad \tilde{p}_r = \lambda_r \cdot \begin{pmatrix} u_r \\ v_r \\ 1 \end{pmatrix} = K_r \cdot R \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + T. \tag{63}$$

In order to triangulate, we use **least-squares** approximation:

$$\begin{aligned} \lambda_1 \cdot \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} &= K \cdot [I|0] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \Rightarrow \lambda_1 p_1 = M_1 \cdot P \qquad \text{left camera.} \\ \lambda_2 \cdot \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} &= K \cdot [R|T] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \Rightarrow \lambda_2 p_2 = M_2 \cdot P \qquad \text{right camera.} \end{aligned} \tag{64}$$

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 $3 \times 2$). We use pseudoinverse approximation (least squares) and get

$$A^T \cdot A \cdot \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = A^T \cdot b \Rightarrow \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = (A^T \cdot A)^{-1} \cdot A^T \cdot b \tag{65}$$

**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.

### Triangulation: nonlinear approach

We want to find $P$ that minimizes the **sum of squared reprojection error**

$$SSRE = d^2(p_1, \pi_1(P)) + d^2(p_2, \pi_2(P)), \tag{66}$$

where

$$d(p_1, \pi_1(P)) = ||p_1 - \pi_1(P)|| \tag{67}$$

is called **reprojection error**. In practice, this is done by initializing $P$ using linear approach and then minimize SSRE using Gauss-Newton of Levenberg-Marquardt.

### Correspondence Problem

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

**Problem:** Exhaustive image search can be computationally very expensive! Can we do that in 1D?
$\rightarrow$ potential matches for $p$ have to lie on the corresponding epipolar line $l'$.

- The **epipolar line** is the projection of the infinite ray $\pi^{-1}(p)$ corresponding to $p$ in the other camera image.
- The **epipole** is the projection of the optical center on the other camera image.
- A **stereo camera** has two epipoles!

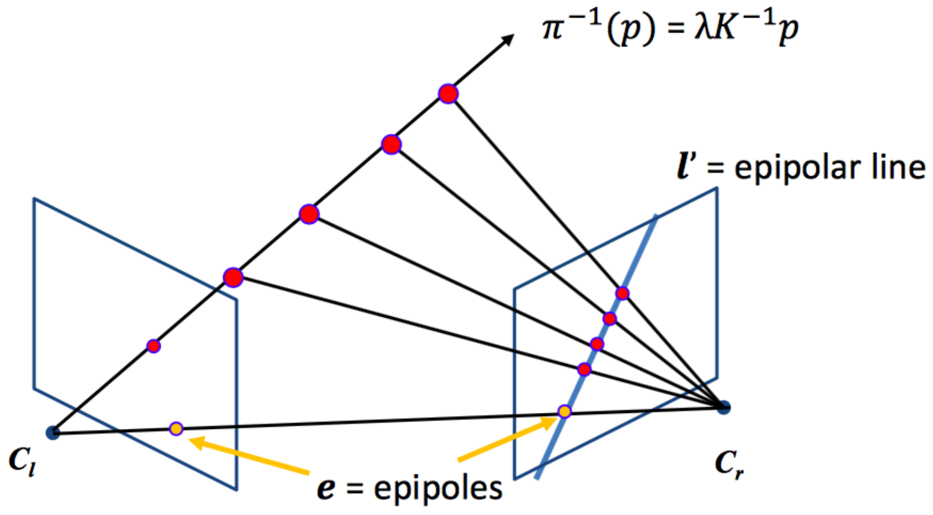


Figure 9: Epipolar lines and epipoles.

**The Epipolar Constraint**

- The epipolar plane is uniquely defined by the two optical centers $C_l, C_r$ and one image point $p$.
- The **epipolar constraint** constraints the location, in the second view, of the corresponding point to a given point in the first view.
- $\Rightarrow$ This reduces the search to 1D problem along conjugate epipolar lines.

**Example: converging cameras**

**Important:** All epipolar lines intersect at the epipole! As the position od the 3D point varies, the epipolar line *rotates* about the baseline.

**Example: forward motion**

Epipoles have the same coordinates in both images: points move along lines radiating from $e$: *Focus of expansion*

**Stereo Rectification**

- Even in commercial stereo cameras, left and right images are not aligned.
- It is convenient if image scanlines are the epipolar lines
- Stereo rectification warps left and right images into new *rectified* images, whose epipolar lines are algined to the baseline.

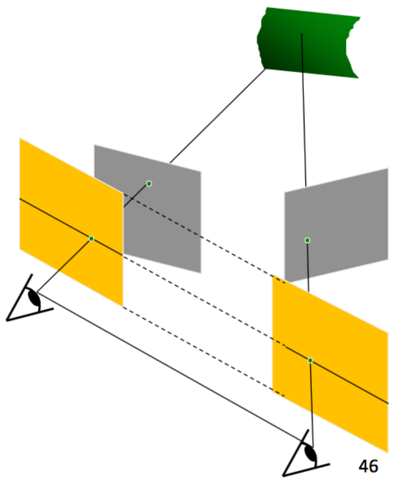


Figure 10: Stereo Rectification.

- Reprojects image planes onto a common plane parallel to the baseline.
- It works by computing two homographies, one for each input image reprojection.
- $\Rightarrow$ Then, scanlines are **aligned** and epipolar lines are **horizontal**.

**Idea:** we define two new Perspective Projection Matrices obtained by rotating the old ones around their optical centers, until focal planes become coplanar (containing the baseline). This ensures **epipoles at infinity**. In order to have horizontal epipolar lines, the baseline should be *parallel* to the new X axis of both cameras. Moreover, corresponding points should have **the same vertical coordinate**. This is obtained by having same intrinsic parameters for the new cameras. Since the the focal length is the same, the new image planes are coplanar. PPMs are the same as the old cameras, whereas the new orientation (the same for both cameras) differs from the old ones by suitable rotations. For both cameras, **intrinsic parameters are the same**.

**Implementation**

1. The perspective equation for a point in the world is

$$\text{image point} = \tilde{p} = \begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K[R|T] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}. \tag{68}$$

   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 $C$):

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot R^{-1} \cdot \left( \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C \right) \tag{69}$$

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

$$\begin{aligned} \lambda_L \cdot \begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} &= K_L \cdot R_L^{-1} \cdot \left( \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_L \right) \quad \text{left camera} \\ \lambda_R \cdot \begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} &= K_R \cdot R_R^{-1} \cdot \left( \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_R \right) \quad \text{right camera} \end{aligned} \tag{70}$$

3. 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

$$\begin{aligned} \lambda_L \cdot \begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} = K_L \cdot R_L^{-1} \cdot \left( \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_L \right) &\rightarrow \overline{\lambda}_L \cdot \begin{pmatrix} \overline{u}_L \\ \overline{v}_L \\ 1 \end{pmatrix} = \overline{K} \cdot \overline{R}^{-1} \cdot \left( \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_L \right) \\ \lambda_R \cdot \begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} = K_R \cdot R_R^{-1} \cdot \left( \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_R \right) &\rightarrow \overline{\lambda}_R \cdot \begin{pmatrix} \overline{u}_R \\ \overline{v}_R \\ 1 \end{pmatrix} = \overline{K} \cdot \overline{R}^{-1} \cdot \left( \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - C_R \right) \end{aligned} \tag{71}$$

4. 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:

$$\begin{aligned} \overline{\lambda}_L \cdot \begin{pmatrix} \overline{u}_L \\ \overline{v}_L \\ 1 \end{pmatrix} &= \lambda_L \cdot \underbrace{\overline{K} \cdot \overline{R}^{-1} \cdot R_L \cdot K_L^{-1}}_{\text{homography left camera}} \cdot \begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} \\ \overline{\lambda}_R \cdot \begin{pmatrix} \overline{u}_R \\ \overline{v}_R \\ 1 \end{pmatrix} &= \lambda_R \cdot \underbrace{\overline{K} \cdot \overline{R}^{-1} \cdot R_R \cdot K_R^{-1}}_{\text{homography right camera}} \cdot \begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} \end{aligned} \tag{72}$$

5. The new $\overline{K}, \overline{R}$ can be chosen as

$$\begin{aligned} \overline{K} &= \frac{K_L + K_R}{2} \\ \overline{R} &= [\overline{r}_1, \overline{r}_2, \overline{r}_3], \end{aligned} \tag{73}$$

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

$$\begin{aligned} &\overline{r}_1 = \frac{C_2 - C_1}{||C_2 - C_1||} \\ &\overline{r}_2 = r_3 \times \overline{r_1} \text{ where } r_3\text{is the third column of } R_L \\ &\overline{r}_3 = \overline{r}_1 \times \overline{r}_2. \end{aligned} \tag{74}$$

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

**Example:** First, you remove the radial distotion (use e.g. bilinear interpolation). Then, compute homographies and rectify (bilinear interpolation).
In order to solve the correspondence problem, one can use a window around the point of interest and use similarity measures to find neighborhoods.

### Correlation-based window matching

- Problem: textureless regions (**aperture problem**), high ambiguity! **Solution:** increase window size to distinguish!
  - Smaller window: **more detail but more noise**!
  - Larger window: **smoother disparity maps but less detail**

### Disparity Map

Input to dense 3D reconstruction

1. For each pixel in the left image, find its corresponding point in the right image.
2. Compute the disparity for each pair of correspondences.
3. Visualized in gray-scale or color coded image.

The depth $Z$ can be computed from the disparity by recalling that

$$Z_P = \frac{bf}{u_l - u_r} \tag{75}$$

Challenges include: occlusion, repetition, non-lambertian surfaces (specularities), textureless surfaces.

### Improvements:

Multiple match could satisfy the epipolar constraint. A good way to address the problem would be to have

- 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.

**Sparse Stereo Correspondence** restrict search to sparse set.

# Lecture 08 - Multiple View Geometry 2

## Two-view Structure from Motion

**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:

$$\begin{aligned} \lambda_1 \cdot \begin{pmatrix} u_1^i \\ v_1^i \\ 1 \end{pmatrix} &= K_1 \cdot [I|0] \cdot \begin{pmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{pmatrix}, \\ \lambda_2 \cdot \begin{pmatrix} u_2^i \\ v_2^i \\ 1 \end{pmatrix} &= K_2 \cdot [R|T] \cdot \begin{pmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{pmatrix} \end{aligned} \tag{76}$$

We have two cases then:

### Calibrated Cameras ($K_1, K_2$ known)

For convenience, we use *normalized image coordinates*

$$\begin{pmatrix} \bar{u} \\ \bar{v} \\ 1 \end{pmatrix} = K^{-1} \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}. \tag{77}$$

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

$$\begin{aligned} \lambda_1 \cdot \begin{pmatrix} \bar{u}_1^i \\ \bar{v}_1^i \\ 1 \end{pmatrix} &= K_1 \cdot [I|0] \cdot \begin{pmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{pmatrix}, \\ \lambda_2 \cdot \begin{pmatrix} \bar{u}_2^i \\ \bar{v}_2^i \\ 1 \end{pmatrix} &= K_2 \cdot [R|T] \cdot \begin{pmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{pmatrix}. \end{aligned} \tag{78}$$

**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**.

- 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

$$\begin{aligned} 4n &\geq 5 + 3n \\ \Rightarrow n &\geq 5. \end{aligned} \tag{79}$$

The first analytical solution for 5 points was given by Kruppa in 1913 (10 degree order polynomial, up to 10 solution with complex ones).
Let's define the **cross product** as a matrix multiplication

$$a \times b = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \cdot \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = [a]_x \cdot b \tag{80}$$

**Epipolar Geometry**

$$\bar{p}_1 = \begin{pmatrix} \bar{u}_1 \\ \bar{v}_1 \\ 1 \end{pmatrix}, \quad \bar{p}_2 = \begin{pmatrix} \bar{u}_2 \\ \bar{v}_2 \\ 1 \end{pmatrix} \tag{81}$$

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

$$\begin{aligned} p_2^T \cdot n &= 0 \\ p_2^T \cdot (T \times p_1') &= 0 \\ p_2^T \cdot (T \times (Rp_1)) &= 0 \\ p_2^T \cdot [T]_x \cdot Rp_1 &= 0 \\ p_2^T \cdot E \cdot p_1 &= 0 \text{ is the epipolar constraint, where } E = [T]_x \cdot R \text{ is the essential matrix.} \end{aligned} \tag{82}$$

Applying the constraints results in *four different* solutions for $R$ and $T$.

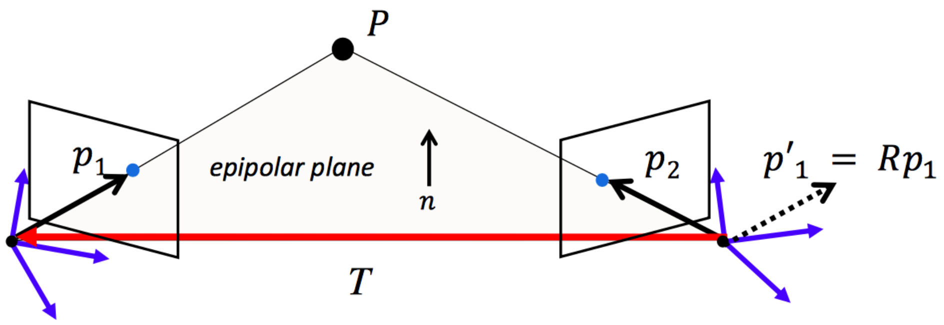


Figure 11: Epipolar constraint

**How to compute the Essential Matrix**

Kruppa's solution is not efficient. In 1996, Philipp proposed an iterative solution. In 2004, the first efficient non iterative solution was proposed. This uses **Groebner Decomposition**. The first popular solution uses 8 points and is called the **8 point algorithm or Longuet-Higgins algorithm** (still used in NASA rovers).

**The 8-point Algorithm**

The Essential matrix is defined by

$$\bar{p}_2^T \cdot E\bar{p}_1 = 0. \tag{83}$$

Each pair of point correspondences provides a linear equation. For $n$ points we can write

$$\underbrace{\begin{pmatrix} \bar{u}_2^1 \cdot \bar{u}_1^1 & \bar{u}_2^1 \cdot \bar{v}_1^1 & \bar{u}_2^1 & \bar{v}_2^1 \cdot \bar{u}_1^1 & \bar{v}_2^1 \cdot \bar{v}_1^1 & \bar{v}_2^1 & \bar{u}_1^1 & \bar{v}_1^1 & 1 \\ \bar{u}_2^2 \cdot \bar{u}_1^2 & \bar{u}_2^2 \cdot \bar{v}_1^2 & \bar{u}_2^2 & \bar{v}_2^2 \cdot \bar{u}_1^2 & \bar{v}_2^2 \cdot \bar{v}_1^2 & \bar{v}_2^2 & \bar{u}_1^2 & \bar{v}_1^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{u}_2^n \cdot \bar{u}_1^n & \bar{u}_2^n \cdot \bar{v}_1^n & \bar{u}_2^n & \bar{v}_2^n \cdot \bar{u}_1^n & \bar{v}_2^n \cdot \bar{v}_1^n & \bar{v}_2^n & \bar{u}_1^n & \bar{v}_1^n & 1 \end{pmatrix}}_{Q \text{ (known)}} \cdot \underbrace{\begin{pmatrix} e_{11} \\ e_{12} \\ e_{13} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{31} \\ e_{32} \\ e_{33} \end{pmatrix}}_{\bar{E} \text{ unknown}} = 0 \tag{84}$$

This problem can be written as

$$Q \cdot \bar{E} = 0. \tag{85}$$

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.
- **Degenerate Solution** if 3D points are coplanar. There is the 5 point algorithm which holds also for coplanar points.

**Interpretation**

With the algorithm we try to minimize the **algebraic error**

$$\sum_{i=1}^{N} (\bar{p}_2^{i^T} \cdot E \cdot \bar{p}_1^i)^2, \tag{86}$$

where

$$\bar{p}_2^T \cdot E \cdot \bar{p}_1 = ||\bar{p}_2|| \cdot ||E \cdot \bar{p}_1|| \cdot \cos(\theta) \tag{87}$$

which is not zero is $p_1, p_2, T$ are not coplanar. When extracting solutions, four results are available. We look only at results with points **in front of both cameras**.

**Uncalibrated Cameras ($K_1, K_2$ unknown)**

It holds

$$\bar{p}_2^T \cdot E\bar{p}_1 = 0, \tag{88}$$

where

$$\begin{pmatrix} \bar{u}_1^i \\ \bar{v}_1^i \\ 1 \end{pmatrix} = K_1^{-1} \cdot \begin{pmatrix} u_1^i \\ v_1^i \\ 1 \end{pmatrix}, \quad \begin{pmatrix} \bar{u}_2^i \\ \bar{v}_2^i \\ 1 \end{pmatrix} = K_2^{-1} \cdot \begin{pmatrix} u_2^i \\ v_2^i \\ 1 \end{pmatrix}. \tag{89}$$

By rewriting the constraint, one obtains

$$\begin{aligned} \begin{pmatrix} u_2^i \\ v_2^i \\ 1 \end{pmatrix}^T \cdot K_2^{-T} \cdot E \cdot K_1^{-1} \cdot \begin{pmatrix} u_1^i \\ v_1^i \\ 1 \end{pmatrix} &= 0 \\ \begin{pmatrix} u_2^i \\ v_2^i \\ 1 \end{pmatrix}^T \cdot F \cdot \begin{pmatrix} u_1^i \\ v_1^i \\ 1 \end{pmatrix} &= 0, \end{aligned} \tag{90}$$

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

$$F = K_2^{-T} \cdot E \cdot K_1^{-1} = K_2^{-T} \cdot [T]_x \cdot R \cdot K_1^{-1}. \tag{91}$$

The same 8-point algorithm can be used to compute the fundamental matrix:

$$\underbrace{\begin{pmatrix} \bar{u}_2^1 \cdot \bar{u}_1^1 & \bar{u}_2^1 \cdot \bar{v}_1^1 & \bar{u}_2^1 & \bar{v}_2^1 \cdot \bar{u}_1^1 & \bar{v}_2^1 \cdot \bar{v}_1^1 & \bar{v}_2^1 & \bar{u}_1^1 & \bar{v}_1^1 & 1 \\ \bar{u}_2^2 \cdot \bar{u}_1^2 & \bar{u}_2^2 \cdot \bar{v}_1^2 & \bar{u}_2^2 & \bar{v}_2^2 \cdot \bar{u}_1^2 & \bar{v}_2^2 \cdot \bar{v}_1^2 & \bar{v}_2^2 & \bar{u}_1^2 & \bar{v}_1^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{u}_2^n \cdot \bar{u}_1^n & \bar{u}_2^n \cdot \bar{v}_1^n & \bar{u}_2^n & \bar{v}_2^n \cdot \bar{u}_1^n & \bar{v}_2^n \cdot \bar{v}_1^n & \bar{v}_2^n & \bar{u}_1^n & \bar{v}_1^n & 1 \end{pmatrix}}_{Q \text{ (known)}} \cdot \underbrace{\begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix}}_{\bar{F} \text{ unknown}} = 0 \tag{92}$$

There are **orders of magnitude** of difference, which leads to poor results with least-squares. How to solve?

### Normalized 8-point algorithm

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 two

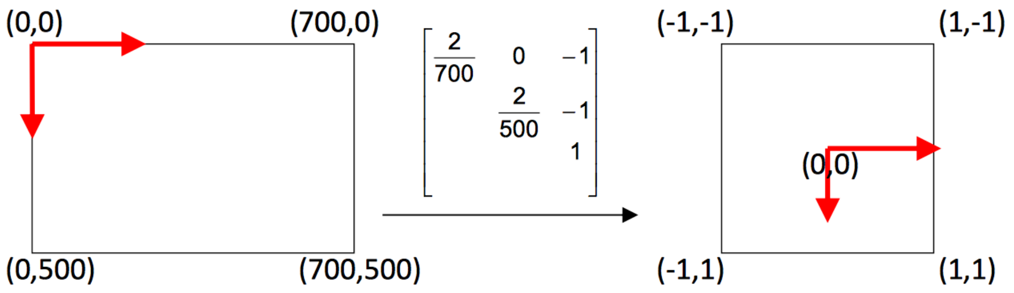

(0,0)
(700,0)
(0,500)
(700,500)
$\begin{bmatrix} \frac{2}{700} & 0 & -1 \\ & \frac{2}{500} & -1 \\ & & 1 \end{bmatrix}$
(-1,-1)
(1,-1)
(0,0)
(-1,1)
(1,1)

Figure 12: Shift for normalized algorithm.

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

$$\hat{p^i} = \frac{\sqrt{2}}{\sigma}\cdot(p^i-\mu), \tag{93}$$

where

$$\mu = \frac{1}{N}\sum_{i=1}^{n} p^i \tag{94}$$

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

$$\hat{p^i} = \begin{pmatrix} \frac{\sqrt{2}}{\sigma} & 0 & -\frac{\sqrt{2}}{\sigma}\mu^x \\ 0 & \frac{\sqrt{2}}{\sigma} & -\frac{\sqrt{2}}{\sigma}\mu^y \\ 0 & 0 & 1 \end{pmatrix}\cdot p^i. \tag{95}$$

The algoritm at the end reads

1. Normalize point correspondences: $\hat{p}_1 = B_1\cdot p_1$, $\hat{p}_2 = B_2\cdot p_2$.
2. Estimate $\hat{F}$ using normalized coordinates $\hat{p}_1, \hat{p}_2$.
3. Compute $F$ from $\hat{F}$ :

$$\begin{aligned} \hat{p}_2^T\cdot\hat{F}\cdot\hat{p}_1 &= 0 \\ B_2^T\cdot p_2^T\cdot\hat{F}\cdot\cdot B_1^T\cdot p_1^T &= 0 \\ \Rightarrow F &= B_2^T\cdot\hat{F}\cdot B_1 \end{aligned} \tag{96}$$

### Error Measures

The quality of the estimated fundamental matrix can be measured by looking at cost functions. The first is defined using the Epipolar Constraint

$$err = \sum_{i=1}^{N} (\bar{p}_2^{iT} \cdot E \cdot p_1^i)^2 \tag{97}$$

This error will exactly be 0 if computed from 8 points. For more points not 0 because of image noise/outliers. Better methods are

### Directional Error

Sum of the angular distances to the Epipolar plane: $err = \sum_i (\cos(\theta_i))^2$, where

$$cos(\theta) = \left( \frac{p_2^T \cdot E \cdot p_1}{||p_2^T|| \cdot ||E \cdot p_1||} \right) \tag{98}$$

### Epipolar Line Distance

### Squared Epipolar-Line-to-point Distances

$$err = \sum_{i=1}^{N} d^2(p_1^i, l_1^i) + d^2(p_2^i, l_2^i). \tag{99}$$

Cheaper than reprojection error: does not require point triangulation!

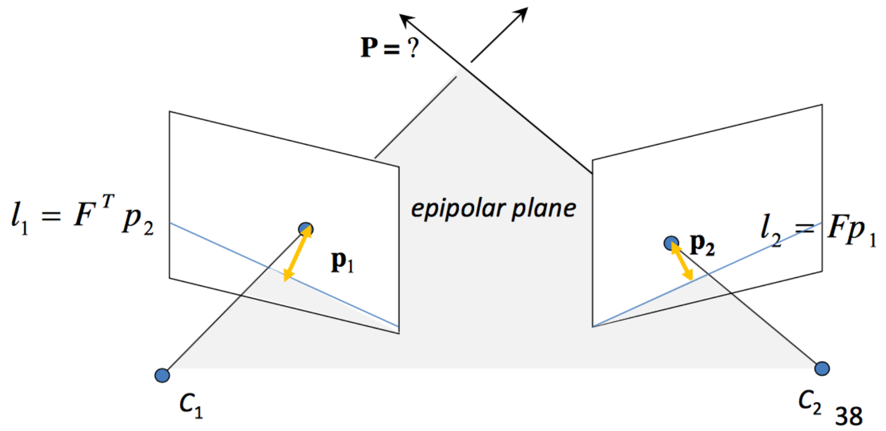


Figure 13: Epipolar Line Distance.

### Reprojection Error

Sum of the **Squared Reprojection Errors**

$$err = \sum_{i=1}^{N} ||p_1^i - \pi_1(P^i)||^2 + ||p_2^i - \pi_2(P^i, R, T)||^2 \tag{100}$$

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

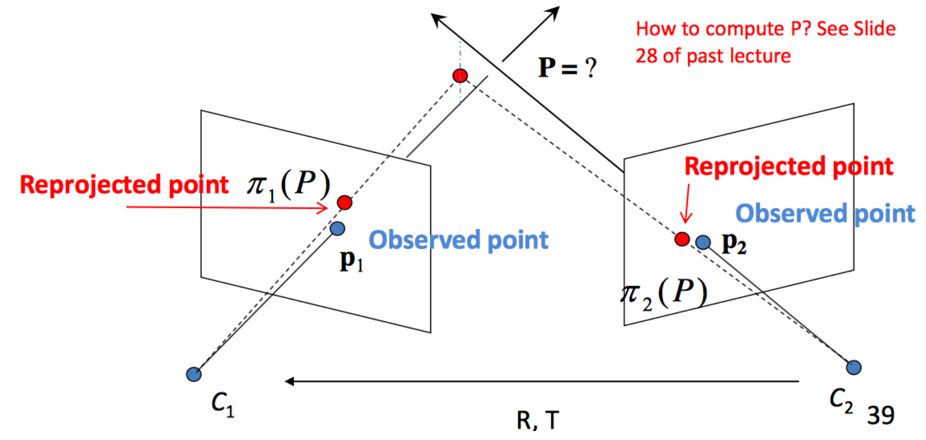


Figure 14: Reprojection Error.

## Robust Structure From Motion

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

- Change in view point
- Image noise
- Occlusions
- Blur

The task of removing them is for **Robust Estimation**. Since error is integrating over time, this increases and is really bad.

### RANSAC (Random Sample Consensus)

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:**

1. Select sample of 2 points at random.
2. Calculate model parameters that fit the data in the sample.
3. Calculate error function for each data point.
4. Select data that supports current hypothesis.
5. Repeat.
6. Select the set with the maximum number of inliers obtained within $k$ iterations.

How many iterations? All pairwise combinations : $\frac{N\cdot(N-1)}{2}$. This is computationally unfeasible is $N$ is too large. With a probabilistic approach, one can reduce this:
Let $w$ be the number of inliers/$N$, $N$ be the total number of data points. We can think of $w$ as

$$w = P(\text{selecting an inlier-point out of the dataset}). \tag{101}$$

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

$$\begin{aligned} w^2 &= P(\text{both selected points are inliers}) \\ 1 - w^2 &= P(\text{at least one of these two points is outlier}) \end{aligned} \tag{102}$$

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

$$(1 - w^2)^k = P(\text{RANSAC never selected two points both inliers}) \tag{103}$$

Let $p$ be the probability of success:

$$\begin{aligned} 1 - p &= (1 - w^2)^k \\ \Rightarrow k &= \frac{\log(1-p)}{\log(1-w^2)}. \end{aligned} \tag{104}$$

RANSAC applied to general model fitting is

1. Initial: let $A$ be a set of $N$ points.
2. Repeat.
3. Randomly select a sample of $s$ points from $A$.
4. Fir a model from the $s$ points.
5. Compute the distances of all other points from this model.
6. Construct the inlier set (i.e. count the number of points whose distance is $< d$).
7. Store these inliers.
8. Until maximum number of iterations $k$ is reached.
9. The set with the maximum number of inliers is chosen as solution to the problem.

$$k = \frac{\log(1-p)}{\log(1-w^s)}. \tag{105}$$

In order to implement RANSAC for Structure From Motion (SFM), we need three key ingredients

a) What's the **model** in SFM? $\rightarrow$ the Essential Matrix (for calibrated cameras) or the Fundamental Matrix (for uncalibrated cameras). Alternatively, $R$ and $T$.

b) What's the **minimum number of points** to estimate the model? $\rightarrow$ We know that 5 points is the theoretical minimum number of points. However, 8-point algorithm, then 8 is the minimum.

c) How do we compute the **distance** of a point from the model. $\rightarrow$ We can use the epipolar constraint to measure how well a point correspondence verifies the model E or F, respectively. However, the **Directional error**, the **Epipolar line distance**, or the **Reprojection error (even better)** are used.

With $s$ points

$$k = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} \tag{106}$$

*Remark.* No 6 DOF estimation for the 2-point RANSAC. $k$ increases exponentially with the fraction of outliers $\varepsilon$.

- As observed, $k$ is exponential in the number of points $s$ necessary to estimate the model.
- 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 $E$.

Can we use less than 5 points? With planar motion

### Planar Motion

Planar motion is described by three parameters $\vartheta, \varphi, \rho$

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

Let's compute the Epipolar Geometry

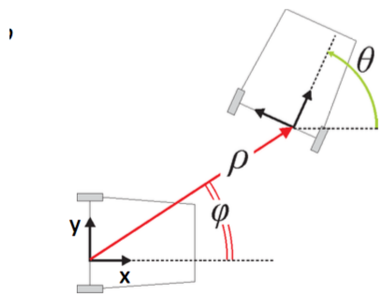


Figure 15: Planar motion.

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

$E$ has 2 DoF ($\theta$, $\varphi$), because $\rho$ is the scale factor. Thus, 2 correspondences are sufficient to estimate them. 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

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

Up to 1000 Hz, 1-point RANSAC in only used to find the inliers. Motion is then estimated from them in 6DOF.

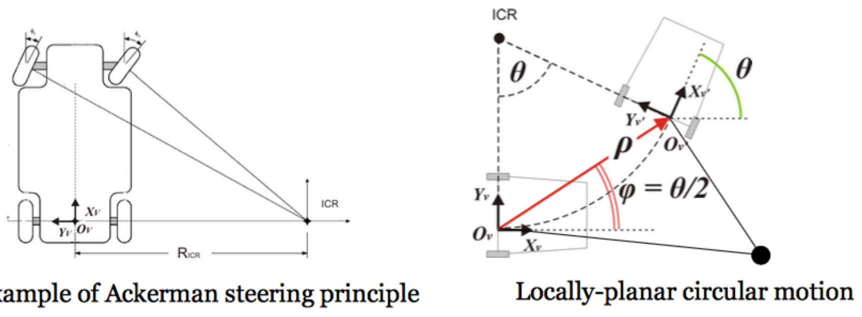


Figure 16: Non-holonomic.

# Lecture 09 - Multiple View Geometry 3

## Bundle Adjustement (BA)

**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:

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

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!

### Hierarchical SFM

1. Extract and match features between nearby frames.
2. Identify clusters consisting of 3 nearby frames:
3. 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.
4. 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).

1. Initialize structure and motion from 2 views (**bootstrapping**).
2. For each additional view:
   - Determine pose (localization).
   - Extend structure (i.e. extract and triangulate new features).
   - Refine both pose and structure (BA).

**2D to 2D** Motion from Image feature correspondences

- Both feature points $f_{k-1}$ and $f_k$ are specified in 2D.
- The minimal-case solution involves 5-point correspondences
- The solution is found by minimizing the reprojection error:

$$T_k = \begin{pmatrix} R_{k,k-1} & t_{k,k_1} \\ 0 & 1 \end{pmatrix} = \text{argmin}_{T_k} \sum_i ||p_k^i - \hat{p}_{k-1}^i||^2. \tag{113}$$

Popular algorithms: 8-/5-point.

**3D to 2D** Motion from 3D structure and Image correspondences

- $f_{k-1}$ is given in 3D, $f_k$ in 2D.
- This problem is known as *camera resection* or PnP (perspective from $n$ points).
- The minimal-case solution involves 3 **correspondences** (+1 for disambiguating the four solutions).
- The solution is found by minimizing the reprojection error:

$$T_k = \begin{pmatrix} R_{k,k-1} & t_{k,k_1} \\ 0 & 1 \end{pmatrix} = \text{argmin}_{X^i,C_k} \sum_{i,k} ||p_k^i - g(X^i, C_k)||^2. \tag{114}$$

Popular algorithms: P3P.

**3D to 3D** Motion from 3D-3D Point correspondences (point cloud registration).

- Both $f_{k-1}$ and $f_k$ are specified in 3D. To do this, it is necessary to triangulate 3D points (e.g. use a stereo camera).
- The minimal case-solution involves **3 non collinear correspondences**
- The solution is found by minimizing the 3D-3D euclidean distance

$$T_k = \begin{pmatrix} R_{k,k-1} & t_{k,k_1} \\ 0 & 1 \end{pmatrix} = \text{argmin}_{T_k} \sum_{i} ||\tilde{X}_k^i - T_k \cdot \tilde{X}_{k-1}^i||. \tag{115}$$

Popular algorithms: Arun 87, ICP, BA.

**Case Study: Monocular Visual Odometry (one camera!)**

**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.
- $\Rightarrow$ 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

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

**Localization**

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

$$\lambda \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot [R|T] \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{117}$$

- 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.

**Extend Structure**

- Extract and triangulate new features.

By denoting the relative motion between adjacent keyframes as

$$T_k = \begin{pmatrix} R_{k,k-1} & t_{k,k_1} \\ 0 & 1 \end{pmatrix}, \tag{118}$$

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

$$C_k = T_{k,k-1} \cdot C_{k-1} \tag{119}$$

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.

**Loop Closure Detection (i.e. Place Recognition)**

- 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)

**VO vs. Visual SLAM**

- VO: Focus on incremental estimation/**local consistency**. VO sacrifies consistency for real-time performance, without need to take into account all previous history of the camera (as SLAM does).
- Visual VSLAM: Simultaneous Localizazion and mapping. Focus on **globally consistent** estimation. Practically VO + loop detection + graph optimization.

**Feature-based Methods**

1. Extract and match featrues (+RANSAC)
2. Minimize Reprojection Error:

$$T_{k,k-1} = \text{argmin}_T \sum_i ||u_i' - \pi(p_i)||^2_\Sigma \tag{120}$$

**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)**

1. Minimize **photometric error**:

$$T_{k,k-1} = \text{argmin}_T \sum_i ||I_k(u_i') - I_{k-1}(u_i)||_\sigma^2, \tag{121}$$

where

$$u_i' = \pi(T \cdot (\pi^{-1}(u_i) \cdot d)) \tag{122}$$

**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.

**ORB-SLAM**

- Feature based:
  - Fast corner + Oriented Rotated Brief descriptor.
  - Binary descriptor.
  - Very fast to compute and compare.
  - Minimizes reprojection error.
- Includes:
  - Loop closing.
  - Relocalization.
  - Final optimization.
- Real time: 30Hz

**LSD-SLAM**

- Direct based + Semi-dense formulation:
  - 3D geometry represented as semi dense depth maps.
  - minimizes **photometric error**.
  - **Separately** optimizes poses and structures.
- Includes:
  - Loop closing.
  - Relocalization.
  - Final optimization.
- Real time: 30Hz

**DSO**

- Direct based + sparse formulation:
  - 3D geometry represented as sparse large gradients.
  - Minimizes **photometric error**.
  - **Jointly** optimizes poses and structures (sliding window).
  - Incorporate photometric correction to compensate exposure time change
- Real time: 30Hz

**SVO**

- Direct based :
  - Corners and edgelets.
  - Frame to frame motion estimation.
- Feature based :
  - Frame to Keyframe pose refinement.
- Mapping:
  - Probabilistic depth estimation.
  - Multi camera system
- 400 fps on i7 laptops, 100 fps on smartphone PC

# Lecture 10 - Dense 3D Reconstruction

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 the multi-view stereo, we have:
Input: calibrated images from several viewpoints.
Output: 3D object dense reconstruction.

## Sparse Reconstruction

We want to estimate the structure from a *dense* region of pixels (hence not only from corners). The workflow is:

1. **Local methods**: estimate depth for every pixel independently.
2. **Global methods**: refine the depth surface as a whole by enforcing smoothness constraint.

We use the **photometric error** (SSD): this 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.

### Aggregated Photometric Error

The Dense reconstruction requires establishing dense correspondences. These are computed basing on the photometric error (SSD between corresponding patches of intensity values (min patch size: $1 \times 1$ pixels). Pros and cons of large and small patches?

- Small window: Pro: more detail, Cons: more noise.
- Large window: Pro: smoother disparity maps, Cons: less detail.

Not all te pixels can be matched reliably, due to viewpoint changes, occlusions. We take advantage of *many* small baseline views, where *high quality* matching is possible. Important facts:

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

### Disparity Space Image (DSI)

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

$$C(u,v,d) = \sum_k \rho\left(\tilde{I}_k(u',v',d) - I_r(u,v)\right), \tag{123}$$

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 $d$. Furthermore $rho$ is the photometric error (SSD).

### Solution to depth estimation problem

Is a function $d(u, v)$ in the DSI that satisfies:

$$\begin{aligned}&\text{Minimum aggregated photometric error (i.e. } argmin_d C)\\ &\text{AND}\\ &\text{Piecewise smooth (global methods)}\end{aligned} \tag{124}$$

Interpolating while not overfitting!
**Global Methods:** We formulate them in terms of energy minimization. The objective is to fin a surgace $d(u, v)$ that minimizes a global energy

$$E(d) = \underbrace{E_d(d)}_{\text{data term}} + \underbrace{\lambda \cdot E_s(d)}_{\text{regularization term}}, \tag{125}$$

where

$$E_d(d) = \sum_{(u,v)} C(u,v,d(u,v)) \tag{126}$$

and

$$E_s(d) = \sum_{(u,v)} \rho_d(d(u,v) - d(u+1,v)) + \rho_d(d(u,v) - d(u,v+1)). \tag{127}$$

$\rho_d$ is a norm (e.g. the $L_1, 2$ or Huber norm). $\lambda$ controls the tradeoff data (regularization). What happens as $\lambda$ increases? Higher smoothing!

### Regularized depth maps

- **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)

$$\rho_d(d(u,v) - d(u+1,v)) \cdot \rho_I(||I(u,v) - I(u+1,v)||) \tag{128}$$

- $\rho_I$ is some monotically *decreasing* function of intensity differences: **lower** smoothness cost for **high intensity gradients**.

### Choosing the stereo baseline

What is the optimal baseline?

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

A possible approach is **depth map fusion**.

### GPGPU for Dense Reconstruction

General Purpose Computing on Graphics Processing Unit. Perform demanding calculations on the GPU instead of the CPU. We can run processes in **parallel** on thousands of cores (CPU is optimized for serial processing). More transistor for data processing.

- 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.

And

- **Image processing**
  - Filtering and feature extractions (e.g. convolutions)
  - Warping (e.g. epipolar rectification, homography).
- **Multiple-view geometry**
  - Search for dense correspondences (pixel wise operations, matrix and vector operations (epipolar geometry).
  - Aggregated photometric error.
- **Global Optimation**
- Variational methods (i.e. regularization (smoothing)) (divergence computation)

Typically on consumer hardware: 1024 threads per multiprocessor, 30 multiprocessors: 30000 threads. CPU with 4 cores which supports 32 threads. High arithmetic intensity. Have a look at Scaramuzza work!

# Lecture 11 - Tracking

## Point Tracking

**Problem:** Given two images, estimate the motion of a pixel from image $I_0$ to image $I_1$. Two approaches exist, depending on the amount of motion between the frames:

- **Block-based methods**
- **Differential methods**

### Block-based methods

- 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.

### Differential Methods

- 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)

$$\begin{aligned} SSD &= \sum (I_0(x,y) - I_1(x+u, y+v))^2 \\ &\approx \sum (I_0(x,y) - I_1(x,y) - I_x \cdot u - I_y \cdot v)^2 \\ &= \sum (\Delta I - I_x \cdot u - I_y \cdot v)^2, \end{aligned} \tag{129}$$

which is a simple quadratic function in two variables $(u, v)$.

### Motion Vector

To minimize the $E$, we differentiate with respect to $(u, v)$ and equate to 0.

$$\begin{aligned} \frac{\partial E}{\partial u} &= 0 \Rightarrow -2 \sum (\Delta I - I_x \cdot u - I_y \cdot v) = 0 \\ \frac{\partial E}{\partial v} &= 0 \Rightarrow -2 \sum (\Delta I - I_x \cdot u - I_y \cdot v) = 0 \end{aligned} \tag{130}$$

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

$$\begin{aligned} \begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} &= \begin{pmatrix} \sum I_x \cdot \Delta I \\ \sum I_y \cdot \Delta I \end{pmatrix} \\ \begin{pmatrix} u \\ v \end{pmatrix} &= \begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{pmatrix}^{-1} \cdot \begin{pmatrix} \sum I_x \cdot \Delta I \\ \sum I_y \cdot \Delta I \end{pmatrix} \end{aligned} \tag{131}$$

# Lecture 13 - Visual Inertial Fusion

## 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:

$$C_k = \text{argmin}_{c_k} \sum_i \sum_j ||C_i - C_j \cdot T_{ij}||^2 \tag{132}$$

- 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.

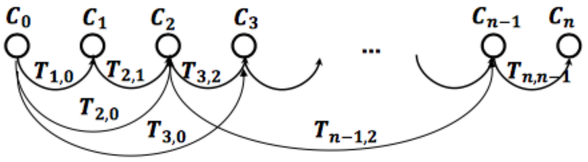


Figure 17: Pose graph optimization.

## Bundle Adjustment (BA)

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

$$X^i, C_k = \text{argmin}_{X^i, C_k} \sum_i \sum_k \rho\left(p_k^i - \pi(X^i, C_k)\right). \tag{133}$$

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

$$\begin{aligned} &\textbf{Huber} \quad \rho(x) = \begin{cases} x^2, & if\ |x| \leq k \\ k \cdot (2|x| - k) & if\ |x| \geq k \text{ linear} \end{cases} \\ &\textbf{Tukey} \quad \rho(x) = \begin{cases} \alpha^2 & if\ |x| \geq \alpha \\ \alpha^2 \cdot \left(1 - (1 - (\frac{x}{\alpha})^2)^3\right) & if\ |x| \leq \alpha. \end{cases} \end{aligned} \tag{134}$$

### 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. 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**

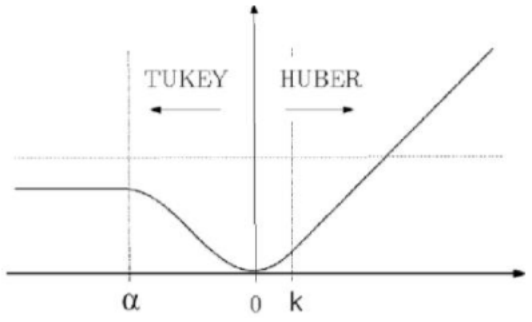


Figure 18: Tukey vs. Huber norm.

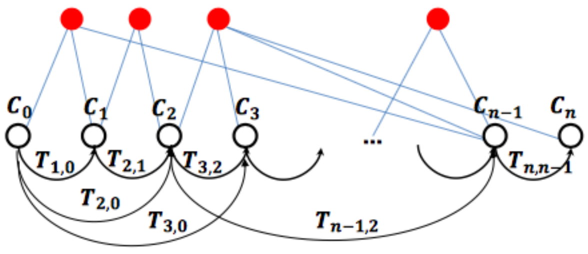


Figure 19: Bundle Adjustment.

## IMUs

Inertial Mesaurement Unit. Measures **angular velocity** and **linear accelerations**.

- 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. Have a look!

**Why IMU?**

- 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.

**Why visual inertial fusion?**

- **Cameras**

  + Precise in slow motion.
  + Rich information for other purposes

  - Limited output rate ($\sim 100Hz$)
  - Scale ambiguity in monocular setup.
  - Lack of robustness

- **IMU**

  + Robust.
  + High output rate ($\sim 1000Hz$).
  + Accurate at high acceleration.

  - Large relative uncertainty when at low acceleration/angular velocity.
  - Ambiguity in gravity / acceleration.

Together, they can work for state estimation: loop detection and loop closure.

**IMU: Measurement Model**

$$\begin{aligned}\tilde{\omega}_{WB}^B(t) &= \omega_{WB}^B(t) + b^g(t) + n^g(t)\\ \tilde{a}_{WB}^B(t) &= R_{BW}(t) \cdot \left(a_{WB}^W(t) - g^W\right) + b^a(t) + n^a(t)\end{aligned} \tag{135}$$