Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

INFORMATION ENGINEERING DEPARTMENT

COMPUTER ENGINEERING - AI & ROBOTICS

# 3D Data Processing 2023/2024

Francesco Crisci 2076739

# Contents

Chapter 1

# Intoduction concepts

## 1.1 Sensors

A point cloud is a data structure used to represent a collection of **multi-dimensional points** and is commonly used to represent three-dimensional data. The points usually represent the X,Y and Z geometric coordinates of a sampled surface. Each point can hold additional information: RGB colors, intensity values and so on.

### 1.1.1 Proximity sensors

An emitter transmits a light wavelength. Light eventually reflected back from a nearby object, e.g. perceived by a photoresistor.

### 1.1.2 Sonar

Senses the range of an obstacle from the round-trip time of an ultrasound pulse. The narrower the cone, the higher the angular resolution of the sensor.

### 1.1.3 LiDAR

It stands for Light Detection And Ranging. We have two types: **Continuous wave**, and **Pulse-based**: it measures directly the time-of-flight, i.e. the round-trip-time of a pulse of light. Needs very short laser pulses and high temporal accuracy.
**Solid-state LiDAR** uses non-mechanical elements to deflect the laser beams. Usually an optical phase array (OPA) of closely spaced optical antennas.

### 1.1.4 Radar

Radio Detection and Ranging are similar to LiDARs, but emit radio waves. **FMCW**(Frequency-Modulated Continuous-Wave) radars emit a signal where the frequency varies linearly over time. The difference between the frequency of the signal sent and that received is linearly related to the distance from the object that generated the reflected signal.
Lower angular and range resolution/accuracy w.r.t. LiDARs, but more robust to environmental sonditions cush as snow, fog, ... and able to compute the relative velocity between the sensor and the objects. In **imaging radar**, multiple FMCW transceivers are cascaded together tpo generate a dense 3D point cloud of the surrounding environment.

### 1.1.5 Time-of-flight Cameras

Meeting point between LiDARs and digital cameras. IR light emitters illuminate the scene, sensed by a 2D imaging sensor sensible to the projected light.

### 1.1.6 Stereo Cameras

Two or more usually identical, rigidly mounted cameras, framing the same scene from different point of view. Knowing the rigid body transformation between the two cameras and the pixel onto which a 3D point is projected in both cameras, it is possible to estimate the depth of the 3D point by usinbg triangulation techniques.
**Active stereo cameras**: To create visual saliency also in homogeneous surfaces. Couple the digital cameras with a dense pattern projector that projects into the scene some visible textured pattern.
**Structured Light Camreas**: SL cameras employ the same operating principle of stereo cameras, but one of the two cameras is replaced with a light projector that illuminates the scene with a textured visual pattern.

The pattern is known and each pattern patch represents a unique binary matrix.
**Time-Multiplexing coding**: Projecting a series of light patterns so that every pixel is encoded with the codeword identified by the pattern sequence. Most common structure of the patterns is a sequence of stripes in(de)creasing their width by the time.

### 1.1.7 RGB-D Cameras

RGB-D cameras are sensor ensembles composed by a ToF or a SL camera rigidly coupled in the same chassis with a color camera. Each pixel of the RGB image is associated with a corresponding pixel in depth map.

## 1.2 Linear Algebra and Rigid-Body Transofrmationas

When dealing with 3D data, we often need to change the coordinate frame, move sensors, move objects in front of sensros, etc...
Many 3D sensors are based on ptojective geometry.
**Euclidean Space**: $E^N$, the set of n-dimensional points that satisfy the axioms of Euclidean geometry. Relationships can be expressed in terms of angles and distances. No special point, no additions between points, only differences (**vectors**).
**Linear Vector Space**: A set V of objects over the field F is a vector space if its elements are closed under **scalar multiplication** and **vector summation**. Given two vectors $u, v \in V$ and $\alpha, \beta \in \mathbb{F}$, we have $\alpha u + \beta v \in V$.
**Euclidean Vector Space**: a linear space denoted by $\mathbb{R}^N$ is called **real vector space**, each element defined by an ordered tuple. Any point in $E^N$ can be identified with a vecotr in $R^N$.
**Dot Product**: Given two vector u,v in $R^N$, we define the canonical inner product, or dot product, as:

$$< u, v >= u^T v = u_1 v_1 + ... + u_n v_n$$

**Distances and Angles**: dot product is used to measure distances: Euclidean norm or L2-norm is just the square root of the dot product. Dot product is also used to measure angles.
**Subspaces**: a subset $w \subseteq V$ of a vector space V is called a subspace if the zero vector belongs to W. given a set of vectors S, the spanned subspace is :

$$span(S) = \left\{ \sum_{i=1}^{n} \alpha_i v_i | v_i \in S, \alpha_i \in \mathbb{R} \right\}$$

S is linearly independent if $\sum_{i=1}^{n} \alpha_i b_i = 0 \Rightarrow \alpha_i = 0 \forall i = 1, ..., m$.
**Basis**: A seto of vectors S is said to be a basis for V if is linearly independent and it spans the whole space V. There are other notions from linear algebra, but it is the fifth year seeing this stuff, so i'm gonna skip them. Feel free to look at the slides.

### 1.2.1 Rigid Objects

A non-deformable objecy O in 3D can be modeled by a set of points in a Euclidean vector space. We assume that the coordinate frame of this space represents an inertial frame we call the **world** frame. If the objecy starts to move, its points move accordingly, i.e., they change their coordinates p(t) w.r.t. the world frame over time. Distancces between any of these two points u(t), q(t) must be preserved over time as the object moves:

$$||u(t_1) - q(t_1)|| = ||u(t_2) - q(t_2)||, \forall t_1, t_2 \in \mathbb{R}$$

A transformation that preserves distances is called a Euclidean transformation. Not enough to describe a rigid body movement. Consider reflection: it does not preserve, generally, the cross product.
Individual objects points cannot move relative to each other. Attach a coordinate frame to the object. Rigidbody motion w.r.t. the world frame can be described by the motion of the origin of the object frame and the rotation of the object frame.
**Object pose** is defined by translational and rotational part. The translational part is a vector t between the origin of the world frame and that of the object frame. The rotational part is a relative orientation R of the object frame axes w.r.t. fixed world frame axes. In practice, we apply R first, then add the translation t.
Suppose the object is performing a pure rotation around its origin o. We may always assume that the origin of the world frame is the center of rotation o.

### 1.2.2 Rotation Matrix

All objects coordinates $p_o$ are related to $R^3$ standard basis in the object frame:

$$p_o = [x_o \ y_o \ z_o]^T = x_o e_{o,1} + y_o eo, 2 + z_o eo, 3$$

To represent the positon of any point w.r.t. world, we just need to present $e_{o,1}, e_{o,2}, e_{o,3} : T(e_{o.,1} = r_1), T(e_{o.,2} = r_2), T(e_{o.,3} = r_3)$ More math on the slides, including the slides "FROM OBJECTS TO CAMERA".

## 1.3 Homographies

Some recap about distortion models, and how to correct the distortion. To fit the image, the input k matrix may be changed while performing image undistortion.

### 1.3.1 Back-Projections

Given a depth map how can we project back the rays into 3D points? For each pixel (u,v): we look at the pixel dept d=depth(u,v), we normalize the values:

$$u_n \frac{u-u_c}{\alpha_u}, \qquad v_n = \frac{v-v_c}{\alpha_v}$$

and we calculate back-projection computed as:

$$\begin{bmatrix} Z \\ Y \\ Z \end{bmatrix} = d \begin{bmatrix} u_n \\ v_n \\ 1 \end{bmatrix}$$

### 1.3.2 Scene to Image Homography

A **homography** is a transformation in GL(n)/R that maps points and lines to lines. Very useful model when dealing with planar scenes.
Let Q and q be a 3D point on a planar board and its 2D projection into the image plane, respectively:

$$\tilde{q} = AT\tilde{Q} = K[R|t]\tilde{Q}$$

Without loss of generality, we can choose to define the object plane so that Z=0:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K[r_1\ r_2\ r_3\ t] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = K[r_1\ r_2\ t] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = H \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

Due to the scale equivalence of homogeneous coordinates, H is defined up to a scalar facto (8 DOFs)

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = H \begin{pmatrix} X \\ y \\ 1 \end{pmatrix} \rightarrow \lambda H = H \forall \lambda \in \mathbb{R}$$

If we normalize the image coordinates, we have $H = [r_1\ r_2\ t]$
If the rigid body transformation is unknown, but the camera is calibrated and we are able to estiamte H from data:

$$H = [r_1\ r_2\ t]$$

we can extract R and t as:

- Scale H with $\lambda = 1/|h_1|$ to try to guarantee the condition of orthonormality
- t is just the third column of H
- R can be computed in closed form:

$$r_3 = r_1 \times r_2$$

from

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

In homogeneous coordinates we have:

$$x = \frac{h_{11}X + h_{12}Y + h_{13}}{h_{31}X + h_{32}Y + h_{33}} \Rightarrow x(h_{31}X + h_{32}Y + h_{33}) - h_{11}X + h_{12}Y + h_{13} = 0$$

$$y = \frac{h_{21}X + h_{22}Y + h_{23}}{h_{31}X + h_{32}Y + h_{33}} \Rightarrow y(h_{31}X + h_{32}Y + h_{33}) - h_{21}X + h_{12}Y + h_{23} = 0$$

From a set of four correspondences we obtain 8 equations with unknowns the elements of H. Stacking H as a $9 \times 1$ column vector h, we have to solve the following underdetermined system Ah=0.
The singular Value decomposition (SVD) of a $m \times n$ real matrix is a factorization of the form:

$$M = U\Sigma V^T$$

To solve the system Ax=0 we can use SVD, obtaining $U\Sigma V^T x = 0$ and set as solution the **last column of V**. The minimum number of correspondences required to estimate a homography is four. Better to use more than four correpsondences. A number of matches n greater than the minimum will not match exactly the same homography due to image noise: **use a least square solution for a over-determined system of 2n linear equations**. We can written the problem as:

$$\arg\min_x((Ax)^A x),\ \textit{subject to}\ ||x|| = 1$$

Let $y = V^T x$, we can rewrite the problem after SVD as:

$$\arg\min_y(y^T \Sigma^T \Sigma y),\ \textit{subject to}\ ||y|| = 1$$

The straightforward solution is $\hat{y} = (0, ..., 0, 1)^T$. From $\hat{x} = V\hat{Y}$ the solution is the **last column of V**.

### 1.3.3 Normalization

SVD-based homography estimation performs badly in presence of noice: **use data normalization prior to the application of SVD**. Translate and scale both image and world coordinates to avoid orders of magnitude difference between the columns of the data matrix:

$$x_n = N_i x,\ X_n = N_s X$$

Shift the coordinates so that the average coordinate is (0,0); Scale the coordinates so that the average distance from the origin is 1. The homography relative to the original coordinates can be recovered as:

$$H = N_i^{-1} \tilde{H} N_s$$

#### Quick sumary

To estimate H from a set of correpsondences Ah=0, normalize the data, compute the SVD and take as solution H' the last column of V. Denormalize H' to H.

### 1.3.4 Pure Rotation Homography

If two views are separated by a pure rotation around the camera center:

$$x = K[I|0]X, \qquad x' = K[R|0]X$$

Mapping between corresponding pixels in the two images are defined by a homography:

$$x' = KRK^{-1}x = Hx$$

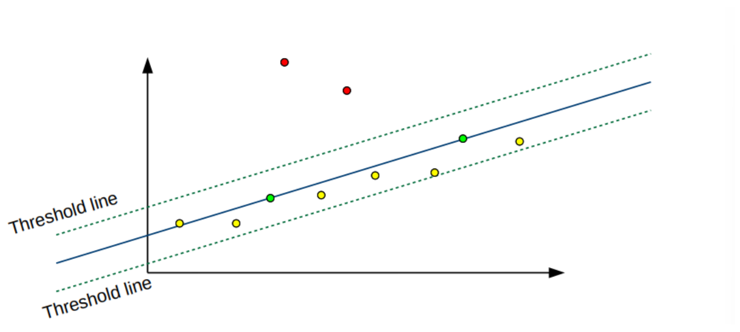
### 1.3.5 Image to Image Homography

If we have two view of a planar scene:

$$\lambda_x x = H_x x, \qquad \lambda_{x'} x' = H_{x'} X,\ H = H_{x'} H_x^{-1} \Rightarrow \lambda x' = Hx$$

From 2D-2D correspondences, it is possible to decompose H into the rigid body transformation R,t that relates the two cameras. Specifically, up to 4 solutions for such decomposition, with at most two physically possible.

### 1.3.6 Robust estimation: RaNSAC Algorithm



Random Sample consensus: repeat the follwoing steps N times: select two points at random (**putative points**), fit a line **l**, find the number of inliers that support **l**. The line with most inliers wins. Fit a new line with all the selected inliers.
For Homography Estimation, we need to repeat the following steps N times:

- Select a set of four correspondences at Random
- Estimate **H** solving the **undetermined** system Ah=0
- Find the number of inlier that support H:
  - For each correspondence $\{P, p\}$, project 3D point P with H, i.e.: p'=HP
  - If the distance between p and p' is less than a threshold, add the correspondence to the inliers set associated to H.
- Find the H with the highest number of inliers among the N trials
- Re-compute a least squares estimate of H using all m inliers solving an **overdetermined** system.

How many putative correspondences N we need to get at least one good sample with high probability? Probability p of getting at least a good sample after N trials:

$$p = 1 - (1 - (1 - \epsilon)^s)^N$$

where $\epsilon$ is the fraction of outliers and s is the number of required correspondences to compute a solution. We have:

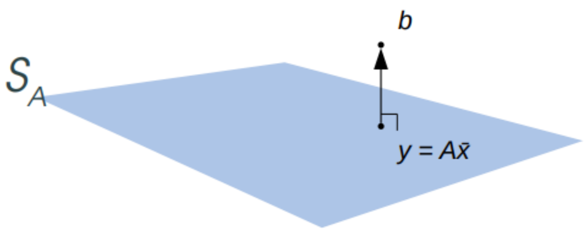
$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^S)}$$

## 1.4 Camera Calibration

Let the following overdetermined linear system Ax=b. we aim to find a solution for the following least square problem:

$$\arg\min_x ||Ax - b||^2$$

The n columns of A span a subspace $S_A \subseteq R^M$ with dimension up to n. Let $\bar{x} \in R^N$ a solution, with $y = A\bar{x} \in S_A$: y is the point in $S_A$ closest to b. y-b should be perpendicular to each of the column of A:



$$A^T(Ax - b) = 0 \Rightarrow A^T Ax = A^T b$$

### 1.4.1 Pseudoinverse

New linear system, with $A^T A$ $n \times n$ symmetric matrix with same soltion as Ax=b. It define the so-called normal equations of a least sqaure problem: if $A^T A$ os invertible, the solution is just:

$$x = (A^T A)^{-1} A^T b$$

defined as the **pseudoinverse of A**. Otherwise, solve the normal equations.

### 1.4.2 Nonlinear Least Squares Problems

Let f a nonlinear model function:

$$f : x \rightarrow f(x, a)$$

f depends on a set of parameters: $a = [aq_1, ..., a_m]^T$. Dataset composed by n pairs $\{X_i, Y_i\}$ ("data points"), where $x_i$ is an independent variable (input) and $y_i$ an observation given $x_i$ (output). "Goodness" of the model parameters with respect to the i-th data point is defined by:

$$\text{residual} \rightarrow e_i(a) = ||y_i - f(x_i, a||$$

Nonlinear least-square methods finds the parameters vector a that minimizes the sum of squared residuals:

$$E(a) = \frac{1}{2} \sum_{i=1}^{n} ||y_i - f(x_i, a||^2 = \frac{1}{2} \sum_{i=1}^{n} e_i(a)^2$$

We assume to have a "good" initial estimate $a_0$ of the parameters vector. the goal of an iterative least square method is to choose at each iteration an update $\delta a$ of the current state $a_k$ that reduces the value of the cost function.

$$a_{k+1} = a_k \delta a$$

We might stop after the number of iterations, step size or something else.

### 1.4.3 Gauss-Newton

Gauss-Newton is an efficient iterative method used to minimize a sum of squared function values:

$$E(a + \delta a) \approx E(a) + dE_a \delta a + \frac{1}{2!}(\delta a^T \ d^2 E_a \ \delta a)$$

We want to find an update $\delta a$ that minimizes the error function E. The Gauss-Newton update:

$$\delta a^{GN} = -(J^T J{-}1 J^T e(a)$$

The Pros are that it generally ensures fast convergence also for nonlinbear model function but the convergence is not guaranteed: an update may increase the cost function.
We have notions of gradient descent here, but i'm gonna omit them since we saw it litterally everywhere.

### 1.4.4 Leverberg-Marquardt

**Levenberg-Maruqrdt** algorithm combines Gauss Newton and Gradient Descent, trying to exploit the strength of both these approaches:

$$\delta a^{LM} = -(J^T J + \lambda I)^{-1} J^T e(a)$$

It tunes the damping factor $\lambda$ at each iteration to ensure rapid progress even where Gauss-Newton fails. If lambda increases we tend to converge toward the Gradient Descent, if it decreases we go toward the Gauss Newton algorithm.

## 1.5 Pinhole Camera Calibration

Camera calibration estimates intrinsic anhd extrinsic parameters of a given camera. **Intrinsic** parameters are the focal length, coordinates of the principal point, distance parameters, etc... **Extrinsic** parameters identify the pose of the camera, e.g. in a world coordinate system.
To calibrate the camera we do the following steps:

- Use a known planar pattern
- Colect a sequence of images of the pattern in several positions, and extract all corners from images
- Exploit the scene-to-image homography consttraints to extract an initial guess of the intrinsic paramteers matrix K' (we assume no distortion for now)
- Normalize all pixel projections with K' and for each image i extract the homography and recover an initial guess of the intrinsic parameters $(R_i, y_i)$
- Recover the final intrinsic and extrinsic parameters solving an iterative least square problem using K' and $(R_i, y_i)$ as initial guess. (We optimize also the distortion parameters here)

### 1.5.1 Data

Collect k images, each one framing a checkerboard with m internal corners from different point of views. For each image i, extract the m 2D corners projections: $p_{i,0}, ..., p_{i,m-1}$

### 1.5.2 Initial Guess for K

We need to estimate an initial guess for K, i.e.:

$$\theta_K = \alpha_u, \alpha_v, u_c, v_c, s$$

For each image i, use the m 3D-2D matches to estimate a scene-to-image homography $H_i$

$$H = [h_1 \ h_2 \ h_3]$$

Remembering that

$$\lambda_H h_1 = K r_1, \qquad \lambda_H h_2 = K r_2$$

From orthonormality of rotation matrix R:

$$r_1^T r_2 = h_1^T K^{-T} K^{-1} h_2 = 0$$

$$r_1^T r_1 = r_2^T r_2 \Rightarrow h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2$$

Exactly two constraints for each homography. We define:

$$B = K^{-T} K^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

Rearrange each constraint as:

$$h_i^T B h_j = v_{ij}^T b$$

where

$$v_{ij} = [h_{1i}h_{1j}, h_{1i}h_{2j} + h_{2i}h_{1j}, h_{2i}h_{2j}, h_{3i}h_{1j} + h_{1i}h_{3j}, h_{3i}h_{2j} + h_{2i}h_{3j}, h_{3i}h_{3j}]^T$$

$$b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$$

Two constraints can be rewritten as:

$$\begin{bmatrix} V_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0$$

that is a 2×6 system of equations.
Given k images arrange a $2k \times 6$ homogeneous linear system: Vb=0. Solve it by using SVD and get B. We applied the **Direct Linear Transformation (DLT) algorithm**, which solves a set of variables from a set of similarity relations. Extract the five intrinsic camera parameters plus one unknown scale factor from the 6 equations defined by:

$$B = \lambda_B K^{-T} K^{-1}$$

### 1.5.3 Initial Guesses for $R_i, t_i$

For each checkerboard position, we normalize all the image points using the initial guess camera intrinsics computed in the previous step. Remembering that $H = [r_1 \; r_2 \; t]$ we can extract R,t.

### 1.5.4 Refined Calibration

At each optimization iteration, given the current estimate of the transformations $T_i$ and the calibration parameters, we can project into each image the m 3D corners and compare them with the corresponding pixels, and optimize again looking for better $T_i$ and calibration parameters that minimize the cost function:

$$\arg \min_{\theta_K, \theta_d, T_0, \ldots, T_{k-1}} \sum_{i=0,j=0}^{k-1,m-1} ||K f_d(A_n T_i \tilde{P}_j) - \tilde{p}_{i,j}||^2$$

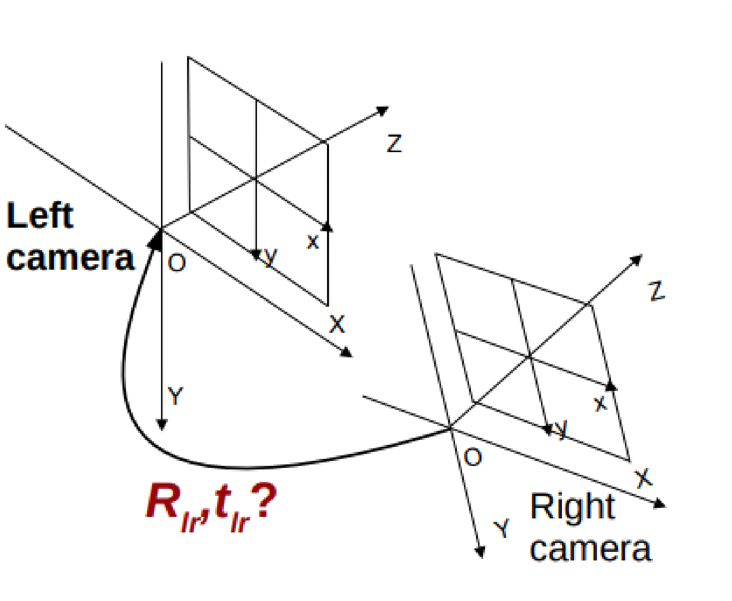
The ideas for the minimization: we use the LM algorithm to solve the iterative problem; we parametrize the rotations with **axis-angle representation**

### 1.5.5 PnP: Pwerspective-n-Point Problem

Given a rigid object and a set of 3D-2D correspondences, how to estiamte the rigid body transformation R,t with a calibrated camera? Find an initial guess for R,t, then solve with LM a nonlinear least-square problem with parameters R,t using as starting point the initial guess.

# Chapter 2

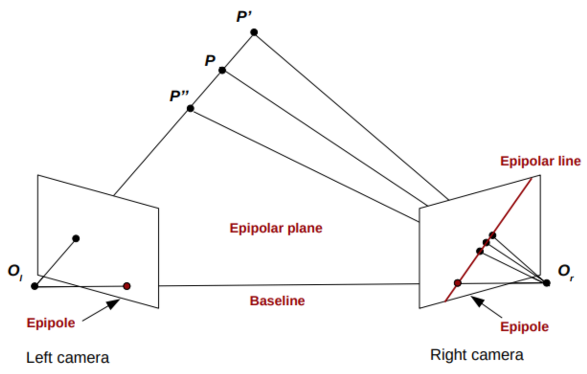
# Epipolar Geometry



Stereo pairs or stereo rig is composed by two calibrated cameras mounted on a rigid support. Common practice is to choose the optical center of one camera to be the origin of the stereo rig coordinate system. To calibrate it we use a checkerboard and, for each i-th view, normalize points and extract R,t for both cameras. For each view find an estimate of $R_{lr}, t_{lr}$. From all views, compute an average $R'_{lr}, t'_{lr}$ and set it as inital guess. Solve a non linear least square problem to compute a refined calibration $T_{lr} = R_{lr}, t_{lr}$ with reprojection errors computed in both cameras:

$$\arg \min_{T_{lr}, T_0, \dots, T_{k-1}} \left( \sum_{i=0,j=0}^{k-1,m-1} ||K^{(l)} f_d^{(l)}(A_n T_i \tilde{P}_j) - \tilde{p}_{i,j}^{(l)}||^2 + \sum_{i=0,j=0}^{k-1,m-1} ||K^{(r)} f_d^{(r)}(A_n T_{lr}^{-1} T_i \tilde{P}_j) - \tilde{p}_{i,j}^{(r)}||^2 \right)$$

3D reconstruction from stereo. Correspondence problem: For a point p in the left image, find the corresponding point p' in the right image. Reconstruction problem: Given two corresponding points p and p', find the 3D coordinates of corresponding scene point P. Since $p_l, p_r$ and $t_{lr}$ coplanar, we have that $\tilde{p}_l(t_{lr} \times R_{lr}\tilde{p}_r) = 0$.



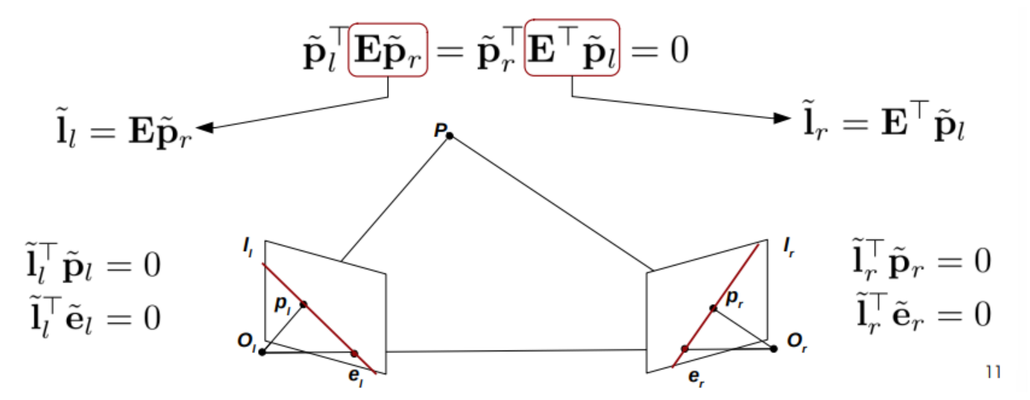
From there we can obtain the following constraint: $\tilde{p}_l^T \hat{t}_{lr} R_{lr} \tilde{p}_r = 0$, obtaining the essential constraint:

$$\tilde{p}_l^t E \tilde{p}_r = \tilde{p}_r^T E^T \tilde{p}_l = 0 \qquad E = \hat{t}_{lr} R_{lr}$$

The essential matrix E encapsulates the rotation and translation associated with the relative pose of the two cameras.

## 2.1 Epipoles and Epipolar Lines

Remembering that in homogeneous coordinates line-point incidence can be verified as: $I^T p = p^T I = 0$ **The**



$\tilde{\mathbf{p}}_l^\top \boxed{\mathbf{E}\tilde{\mathbf{p}}_r} = \tilde{\mathbf{p}}_r^\top \boxed{\mathbf{E}^\top \tilde{\mathbf{p}}_l} = 0$
$\tilde{\mathbf{l}}_l = \mathbf{E}\tilde{\mathbf{p}}_r$
$\tilde{\mathbf{l}}_r = \mathbf{E}^\top \tilde{\mathbf{p}}_l$
P
$\tilde{\mathbf{l}}_l^\top \tilde{\mathbf{p}}_l = 0$
$\tilde{\mathbf{l}}_l^\top \tilde{\mathbf{e}}_l = 0$
$l_l$
$p_l$
$O_l$
$e_l$
$l_r$
$p_r$
$O_r$
$e_r$
$\tilde{\mathbf{l}}_r^\top \tilde{\mathbf{p}}_r = 0$
$\tilde{\mathbf{l}}_r^\top \tilde{\mathbf{e}}_r = 0$
11

**essential matrix is a projective mapping that maps a point onto a line.**

### 2.1.1 Essential Test

To check if a match $p_l <-> p_r$ is correct in a stereo rig compute:

$$\tilde{I}_l = E\tilde{p}r$$

and verify if the distance between $I_l$ and $p_l$ is less than a certain threshold.
Essential matrices belong to a special set of matrices called Essential Space:

$$\epsilon = \{\hat{t}R | R \in S = (3), t \in \mathbb{R}^3\} \subset \mathbb{R}^{3\times3}$$

A nonzero $3 \times 3$ matrix E is an essential matrix if and only if E has the following singular value decomposition:

$$E = U\Sigma V^T \qquad \text{with} \qquad \Sigma = diag\{\sigma, sigma, 0\}$$

for some $\sigma$ inf $R_+$ and U,V in SO(3). E is rank 2, i.e., **is not invertible**.

## 2.2 Pose Recovery from E

From SVD it is possible to compute up to four possible decompositions of E. By defining:

$$w = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

the solution is given by:

$$R = UWV^T \qquad or \qquad UW^TV^T \qquad t = \pm u_3$$

To select a feasible solution, test with a single 3D point the cheirality constraint. When recovering positon from the Essential matrix we get 4 possible solutions: use cheirality constriant to select one.

## 2.3 Fundemantel Matrix

Map to a non distorted real camera:

$$\tilde{x}_l = K_l \tilde{p}_l \qquad \tilde{x}_r = K_r \tilde{p}_r$$

from: $\tilde{p}_l^T E \tilde{p}_r = 0$, we have: $\tilde{x}_l K_l^{-T} E K_r^{-1} \tilde{x}_r = 0$

$$\tilde{x}_l F \tilde{x}_r = 0 \qquad F = K_l^{-T} E K_r^{-1}$$

The fundamental matrix F encapsualtes the rotation, translation and the intrinsic parameters. As the essential matrix, F maps points (pixels) to lines:

$$I' = Fx \qquad I = F^T x'$$

Viceversa is not possible since F it has rank 2, i.e. is not invertible.
F has 7 DoF: like the homography is defined up to a scale factor but, being rank 2, is not invertible. The SVD is given as:

$$F = U diag(\sigma_1, \sigma_2, 0) V^T$$

$$x'^T F x = 0, \qquad x = [x, y, 1]^T \qquad \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \qquad x' = [x', y', 1]^T$$

We obtain:

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

One equation for each correspondence. Get n correspondences and arrange the following linear system:

$$\begin{bmatrix} x_1'x_1, x_1'y_1, x_1', y_1'x_1, y_1'y_1, y_1', x_1, y_1, 1 \\ x_n'x_n, x_n'y_n, x_n', y_n'x_n, y_n'y_n, y_n', x_n, y_n, 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \Rightarrow Af = 0$$

As for H, we have to solve a homogeneous linear system, withy F determined up to a scalar factor:

- n=8: minimal system, solution is the non trivial nullspace of A(**Eight-Point Algorithm**)
- $n > 8$: overdetermined system, solution with linear least squares.

In both case, solve with SVD taking the last column of V corresponding to the smallest singular value: again, **coordinates normalization is essential**.
In general the solutiojn for F will not have zero determinant (i.e., rank 2). solution:

- Take the estimated F', and compute it SVD, $F' = EDV^T$
- Defined $D_n$ as being equal to D but with the smallest singular value set to 0
- Recompute $F = ED_nV^T$

## 2.4 Robust Fundamental Matrix Estimation

- Normalize data
- Repeat the follwoing steps N times:
  - Select a set of eigth correspondences at random
  - Estimate F solving the undetermined system af=0
  - Find the number of inliers that support F
    - For each corresponding $\{p, p'\}$, map p to the corresponding epipolar line, i.e.:$I' = Hp$
    - If the distance between p' and the line I' is less than a threshold, add the correspondence to the iniers associated to F
- Find the F with the highest number of inliers among the N trials
- Re-compute a least square estimate of F using all m inliers solving an ovedetermined system (A $m \times 9$ matrix)
- Eventually enforce Rank 2 to F and denormalize F

## 2.5 Alternative Solutions

**Seven-Point algorithm:** Minimal solution: use seven correspondences and impose det(F)=0.

- Use SVD and take the last two columns of V that define a 2-dimensional null-space of solutions: $F = \alpha F_1 + (1-\alpha)F_2$
- The additional constraint $det(\alpha F_1 + (1-\alpha)F_2) = 0$ leads to a cubic polynomial equation in $\alpha$ which has 1 or 3 solutions: solve this equation, e.g. by exploiting some polynomial root-finding method

**Non-linear approaches**: often used for refinement. given an initial guess of F, setup a non-linear least squares problem where the squared distances between image points and epipolar lines are minimized.

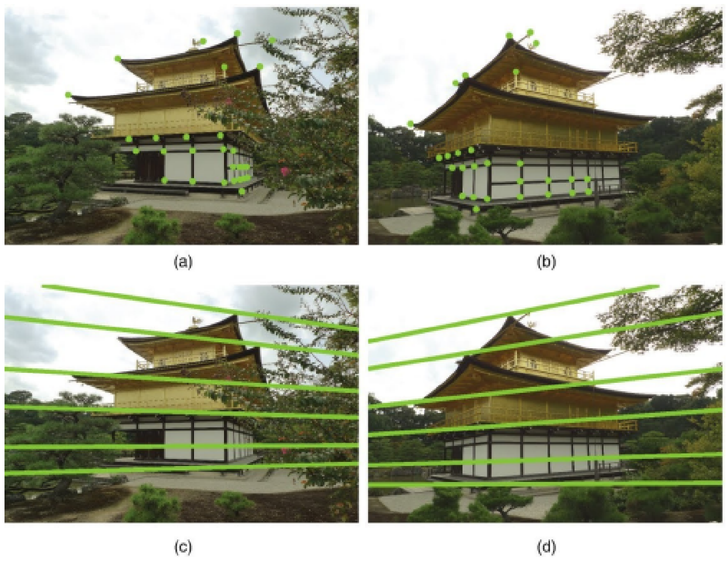
## 2.6 Essential Matrix Estimation

The **Eight-Point Algorithm** can be used also for computation of E, as long as:

- We use normalized coordinates in the correspondences;
- We enforce the affinity with the essential space:
  - Take the estimated E', and compute its SVD, $E' = UDV^T$, with $D = diag\{\sigma_1, \sigma_2, \sigma_3\}$
  - Define $D_n = \{\sigma, \sigma, 0\}$, with $\sigma = (\sigma_1 + \sigma_2)/2$
  - Recompute $E = UD_nV^T$

The minimal solver is represented by the **Five-Point Algorithm**, that uses only five correspondences.

## 2.7 Stereo Rectification



We assume a calibrated, non distorted rig with some vergence. We wish to map the image points onto a pair of image planes that are parallel to the baseline and in the same plane. we aim to perform only **pure rotations** around the cameras: each pixel in the rotated version of the camera is mapped to an original camera pixel by means of a homography.

- Take a plane $\Pi$ perpendicular to the baseline.
- project the unit vectors $z_i$ and $z_j$ of both optical axes into $\Pi$ and take the bisector $z_{ij}$
- compute $y_{ij}$ from $x_{ij}$ and $z_{ij}$ and compose:

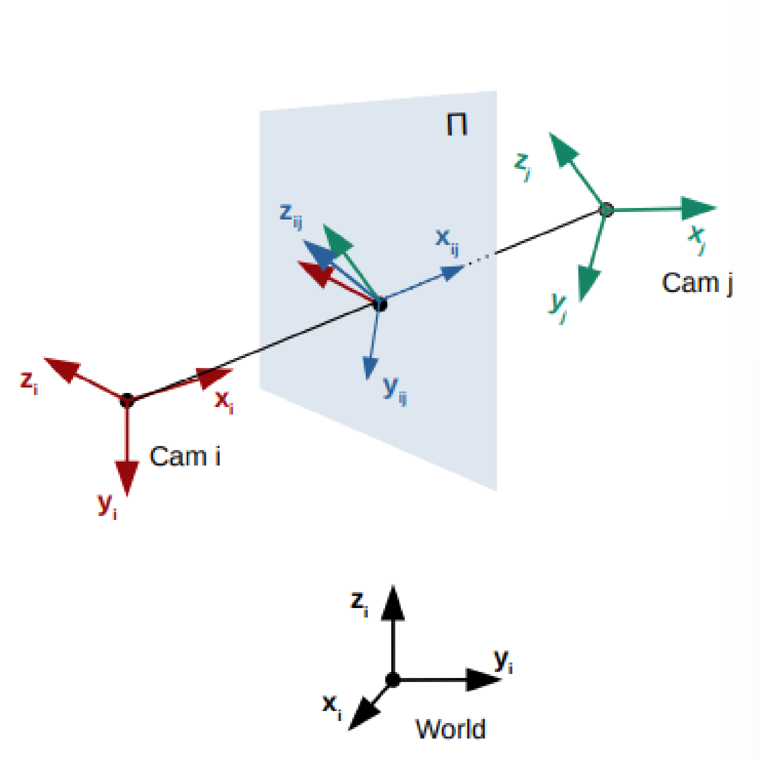
$$R_{ij} = (x_{ij} y_{ij} z_{ij})$$

- Rotate both cameras into their new viewing direction are as follows:

$$R_i^* = R_{ij} R_i^T \qquad \text{and} \qquad R_j^* = R_{ij} R_j^T$$

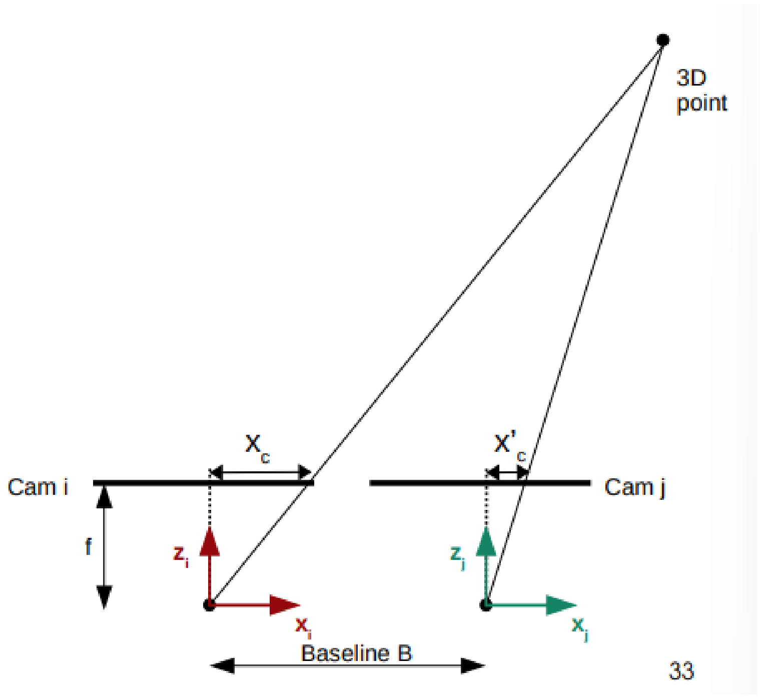
- Map to the original pixel are given by:

$$H = KRK^{-1}$$



### 2.7.1 Trinagulation in a Rectilinear Rig

The **Goal** is to get Z from stereo correspondences. For a rectilinear stereo rig, corresponding left and right image points lie in the same scanline. Two rays from the camera centers trhough the left and right image points can be back-projected in 3D. The two rays and the baseline lie on the epipolar plane.



$$x'_c = f\frac{X}{Z}\ x_c = f\frac{X+B}{Z},\ d = x_c - x'_c = \frac{fB}{Z},\ Z = \frac{fB}{d}$$

# Chapter 3

# Stereo Matching

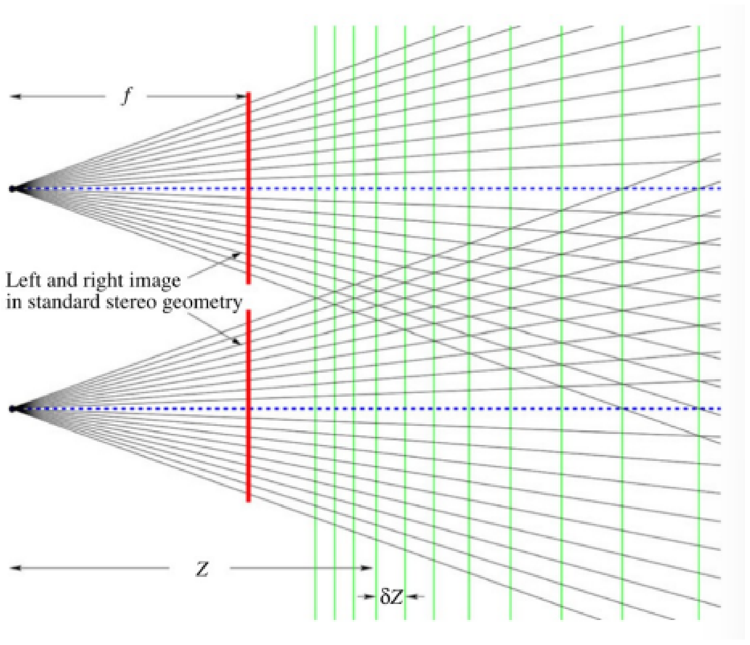
Search in a pair of stereo images for corresponding pixels that are projections of the same 3D point. Epipolar constraints reduces the search to one dimension. Epipolar constraints reduce the search to one dimension.
**Sparse Matching**: extract salient points and compute descriptors. Match according to the minimum distance between descriptors. It is suitable for wide baseline stereo rigs.
**Dense matching**: Find correspondences for all points that are projected in both images.
The **assumptions** are that most scene points are visible from both viewpoints. Corresponding image regions are similar. We consider rectified rectilinear stereo rigs: the epipolar lines are the row of the images (**scanlines**).
**Challenges: Occlusion:** some 3D points are projected in one image but not in the other; **Non-lambertian surfaces**: the appearence of the surfaces may depend on the point of view; **Perspective transformations**: a shape may appear different from a different point of view.

## 3.1 Matching Model

For each pixel (x,y) in the **base image B**, we search for a corresponding pixel in the **match image M**. We set a maximum disparity d and add a special value. Stereo matching as a labeling problem:

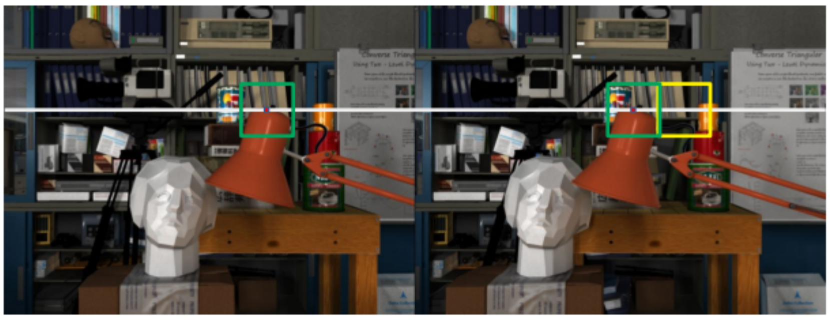
- Left base image: $L = \{+1, 0, -1, ..., -d_{max}\}$ $\quad 0 \leq -d \leq \min\{d_{max}, x-1\}$
- Right base image: $L = \{-1, 0, +1, ..., d_{max}\}$ $\quad 0 \leq d \leq \min\{d_{max}, N_{cols} - x\}$

## 3.2 Depth uncertainty



A disparity of 0 between corresponding pixels $x_L - x_R = 0$ means 3D point at infinity. The distances $\delta Z$ between subsequent depth layers increases nonlinearly.

## 3.3 Block Matching



For each pixel, slide a **support window** along the match scanline for the whole search interval and compare the pixels inside the window with corresponding pixels in the window of the base image.

### 3.3.1 Matching cost

- Sum of squared differences:

$$E_{SSD}(p,d) = \sum_{i=-l}^{l} \sum_{j=-k}^{k} [B(x+i,y+j) - M(x+d+i,y+j)]^2$$

- Sum of absolute differences:

$$E_{SAD}(p,d) = \sum_{i=-l}^{l} \sum_{j=-k}^{k} |B(x+i,y+j) - M(x+d+i,y+j)|$$

- Zero-mean normalized cross correlation:

$$E_{ZNCC}(x,d) = 1 - \frac{\sum_{i=-l}^{l} \sum_{j=-k}^{k} [B_{x+1,y+j} - \bar{B}_x][M_{x+d+i,y+j} - \bar{M}_{x+d}]}{\sqrt{\sigma^2_{B,x}\sigma^2_{M,x+d}}}$$

$$\sigma^2_{B,x} = \sum_{i=-l}^{l} \sum_{j=-k}^{k} [B_{x+i,y+j} - \bar{B}_x]^2 \qquad \sigma^2_{M,x+d} = \sum_{i=-l}^{l} \sum_{j=-k}^{k} [M_{x+d+i,y+j} - \bar{M}_{x+d}]^2$$

- Census tranform:
  - Binary number: one bit for each sample in the Census window
  - Bit value: comparison against the central value
  - Compare the descriptor using Hamming distance

  According to the window size we have different result: smaller window brings high accuracy but more noise, larger window bring higher reliability but less details.

**Local cost** calculation is generally ambiguous because of textureless surfaces, occlusions, repetitions and non-lambertian surfaces.
To improve the block mathcing we can use adaptive windows: start with a standard window size and then adapt the window according to the current disparity estimate and iterate. We could also use hierarchical stereo matchng, or exploit non-local correspondence constraints (**Global Methods**).

## 3.4 Global Methods

Enforce smoothness constraint, i.e. disparity is **piecewise smooth**. Enforce ordering constraints and greater computational complexity.

### 3.4.1 Belief-propagation matching

$$E(f) = \sum_{p\in\Omega}\left[E_{data}(p, f_p) + \sum_{q\in A(p)} E_{smooth}(f_p - f_q)\right]$$

Corresponding pixels should have a similar intensity. Neighboring pixels in the image should have a similar disparity space, e.g.:

$$E_{smooth}(l-h) = E_{smooth}(a) = \begin{cases} 0 & \text{if a} = 0 \\ c & \text{otherwise} \end{cases}$$

$$E_{smooth}(l-h) = E_{smooth}(a) = b|l-h| = b|a|$$

$$E_{smooth}(l-h) = E_{smooth}(a) = \min\{b|l-c|, c\} = \min\{b|a|, c\}$$

Usually 4-pixels adjacency set. the set of all possible labelings has a cardinality $(d_{MAX})^{w\times h}$: intractable exhaustive solution. Solve by using efficient message-passing algorithms. It is accurate but very slow.

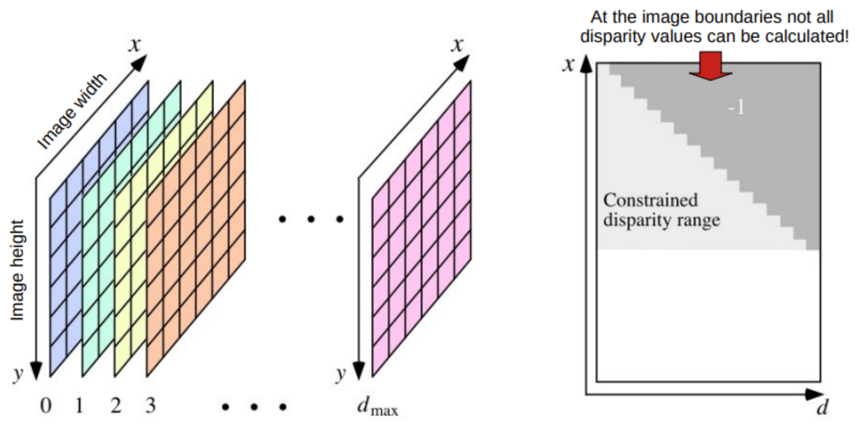

x
Image width
Image height
y
0 1 2 3 • • • $d_{max}$
At the image boundaries not all disparity values can be calculated!
-1
Constrained disparity range
d

Figure 3.1: Cost volume

**Area of influence**

Set used by the stereo matcher for each pixel to decide about its label via the smoothness constraint: in global methods we use the whole image. the idea is to reduce the area of influence, e.g. by considering only some directions.

### 3.4.2 Semi-Global Matching

Consider the following error function:

$$E(f) = \sum_{p\in\Omega}[E_{data}(p, f_p) + \sum_{q\in A(p)} E_{smooth}(f_p, f_q)]$$

with:

$$E_{smooth}(f_p, f_q) = \chi_1 f(f_p, f_q) + \chi_2(f_p, f_q)$$

with

$$\chi_1(f_p, f_q) = \begin{cases} c_1 & \text{if } |f_p - f_q| = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \chi_2(f_p, f_q) = \begin{cases} c_2(p,q) & \text{if } |f_p - f_q| > 1 \\ 0 & \text{otherwise} \end{cases}$$

This method approximate global methods by aggregating costs for a number of directions. the minimization along individual image rows can be performed efficiently in polynomial time using Dynamic programming.

- For each direction, start from one end point and go toward p
- For each pixel along the direction, update the following dynamic programming equation:

$$E(p_i, d) = E_{data}(p_i, d) + E_{smooth}(p_i, p_{i-1}) - \min_{0\le\Delta\le d_{max}} E(p_{i-1}, \Delta)$$

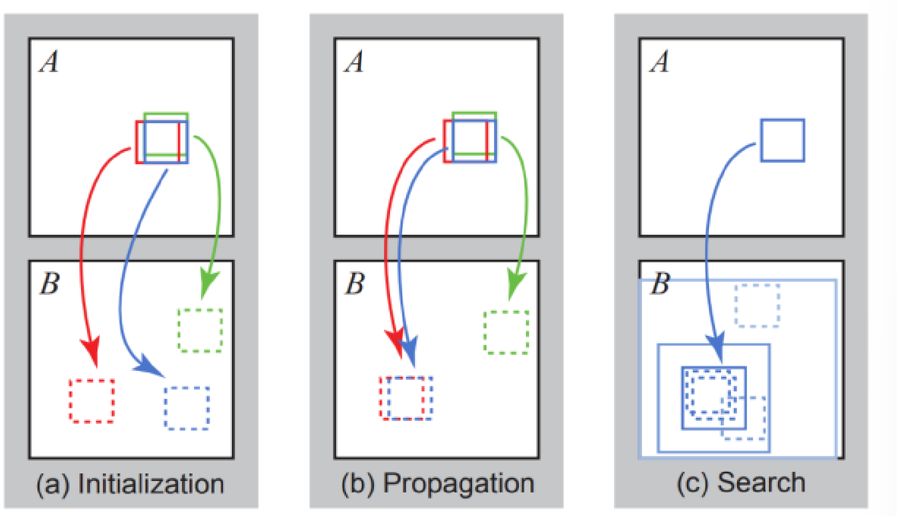
where

$$E_{smooth}(p,q) = \min \begin{cases} E(q, f_q) & \text{if } f_p = f_q \\ E(e, f_q) + c_1 & \text{if } |f_p - f_q| = 1 \\ \min_{0 \le \Delta \le d_{max}} E(q, \Delta) + c_2(p, q & \text{if } |f_p - f_q| > 1) \end{cases}$$

For each direction i (start with $E(p_0, d) = E_{data}(p_0, d)$). Add the last column of each integration matrix associated to a specific direction and **get the minimum**.
This method is really efficient, suitable for real-time applications. Basic algorithm performs badly in large untextured regions.

## 3.5 PatchMatch

Problem definition: given images A and b, find for every patch in A the nearest neighbor in B under some patch distance metric: Nearest-Neighbor Field (NNF): holds dx,dy **offsets**.
**PatchMatch**: a randomized algorithm for rapidly finding correspondences between image patches exploiting **spatial coherence** between images.



### 3.5.1 PatchMatch Propagation

Scan each patch in row major order, from top-left pixel:

- We attempt to improve f(x,y)=(dx,dy) using the known offset of f(x-1,y) and f(x,y-1), assuming that the patch offsets are likely to be the same
- Let D(p,f) denote the patch distance between the patch at p=(x,y) in A and patch p+f=(x,y)+(dx,dy) in B. We take the new value for f(x,y) to be:

$$\arg\min_f \{D(p, f(x,y), D(p, f(x-1,y)), D(p, f(x,y-1)))\}$$

Repeate the scan in inverse order, frombottom-right pixel
We can summarize the PatchMatch algorithm as: repeat until convergence: initialization, forward and backward propagation, then random search.

## 3.6 PatchMatch Stereo

A support window is typically centered on a pixel of the base image, but:

- The support window constrains pixels that lie on a different surface than the center pixel
- The window captures a surface that is slanted

PatchMatch Stereo tries to address the second prolem, by estiamting a 3D locla plane that approximates the slanted surface at each image point: Hence estimating a better support window in the match image.
For each pixel p of both frames, we search a plane $f_p$. Disparity of p is computed as:

$$d_p = a_{f_p} p_x + b_{f_p} p_y + c_{f_p}$$

where $a_f, b_f, c_f$ are the plane parameters being estiamted, $p_x, p_y$ are the pixel coordinates. The cost for matching pixel p with pixel q in the other image according to plane f is computed as:

$$m(p,f) = \sum_{q \in W_p} w(p,q)\rho(q, q - (a_f q_x + b_f q_y + c_f))$$

where:

- $w_p$ is the square support window centered on pixel p
- p(q,q') is the function that computes the pixel dissimilarity
- w(p,q) is the wighting function

The algorithm is the following:

- RRandom initialization
- Repeat until convergence:
- **spatial propagation**: we evaluate whether assigning to p the plane $f_q$ of spatial neighbor pixel q decrease the matching cost. If $m(p, f_q) < m(p, f_p)$, accept the new plane
- **View propagation**: we check all pixels p' of the second view that have our current pixel p as a matching point according to their current plane. if $m(p, f_p) < m(p, f_p)$ accept the new plane.
- **Plane refinement**: sample a new plane $f_{rand}$ around the current one. If $m(p, f_{rand}) < m(p, f_p)$ accept the new plane.

It is a good trade-off between accuracy and computational complexity.

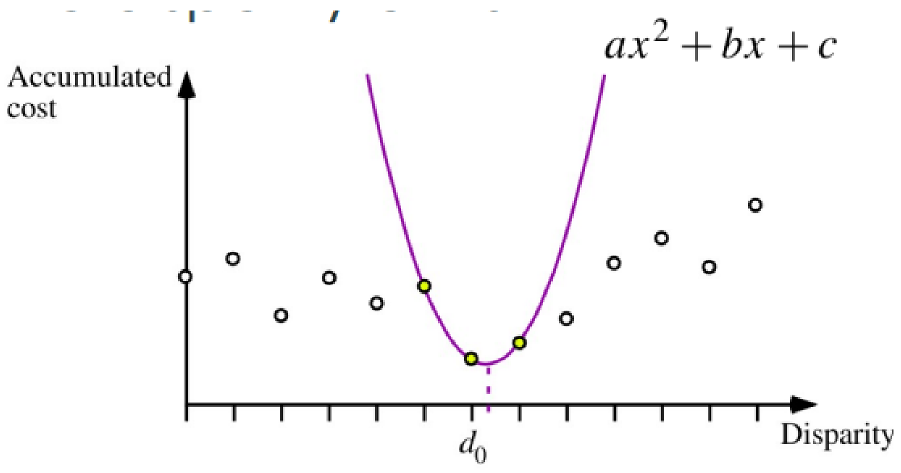
### 3.6.1 Left-Right Consistency

Left-to-right matching and then right-to-left matching, and comparte disparities. Used also as a confidence measure:

$$\Gamma_1(p)\frac{1}{|f_p^{(R)} - f_p^{(L)}| + 1}$$

### 3.6.2 Sub-pixel Accuracy

Fit a parabola to accumulate costs in a neighbourhood of the disparity $d_0$, and take the disparity at its minimum.



## 3.7 Beyond Classical Stereo Matching

### 3.7.1 Active Stereo

Add a projector that produces an highly textured light pattern, enhancing the texture of the scene. Projector-cameras extrinsic parameters are not required. Ausually a Near Infrared (NIR) projector is used: measurement cameras are sensitive to NIR and a "texture" camera equipped with IR-cut filter can be used to acquire RGB information.

### 3.7.2 Projectors

Can be of different types:

- Coherent laser light + diffractive optical elements
- Light projector + mask
- Digital Light Processing (DLP) projectors

### 3.7.3 Structured-Light Sensors

Conceptually a passive stereo system where one of the measurement cameras is replaced by a **calibrated projector**. Projector-cameras extrinsic parameters are required and the knowledge of the spatial and temproal structure of the pattern is required.

### 3.7.4 Temporal-Encoding

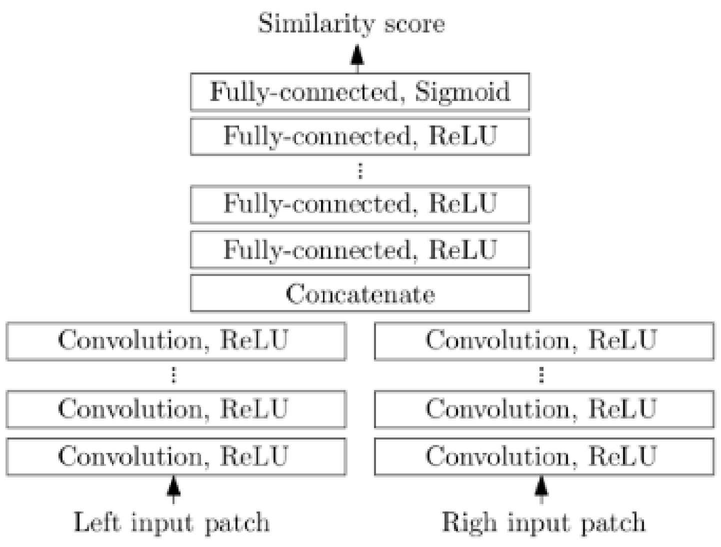
For reasons of error reduction, it is recommended that the projected slides follow the **Gray code** rather than the usual binary code: consecutive integers are represented by binary numbers that differ in one digit only.

| *Integers* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| *Usual code* | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| *Gray code* | 000 | 001 | 011 | 010 | 110 | 100 | 101 | 111 |

## 3.8 Stereo Matching with Deep Learning

### 3.8.1 Matching Cost with a CNN

**MC-CNN:** given two small patches, compute the matching cost with a Convolutional Neural Network. A set of examples is required.



### 3.8.2 MC-CNN Dataset

A set of positive and negative examples:

- Each example is a pair o **pair of patches,** one from the left and one from the right image:

$$< \mathcal{P}^L_{n\times n}, \mathcal{P}^R_{n\times n(\mathbf{q})} >$$

- A positive example is obained by setting the right patch close to the correct match ($o_{pos}$ "small")

$$\mathbf{q} = (x - d + o_{pos}, y)$$

- A negative example is obtained by randomly setting the right patch far from the correct match ($o_{neg}$ "big")

$$\mathbf{q} = (x - d + o_{neg}, y)$$

### 3.8.3 MC-CNN Loss Function

We use the binary cross-entropy:

$$t \log(s) + (1 - t) \log(1 - s)$$

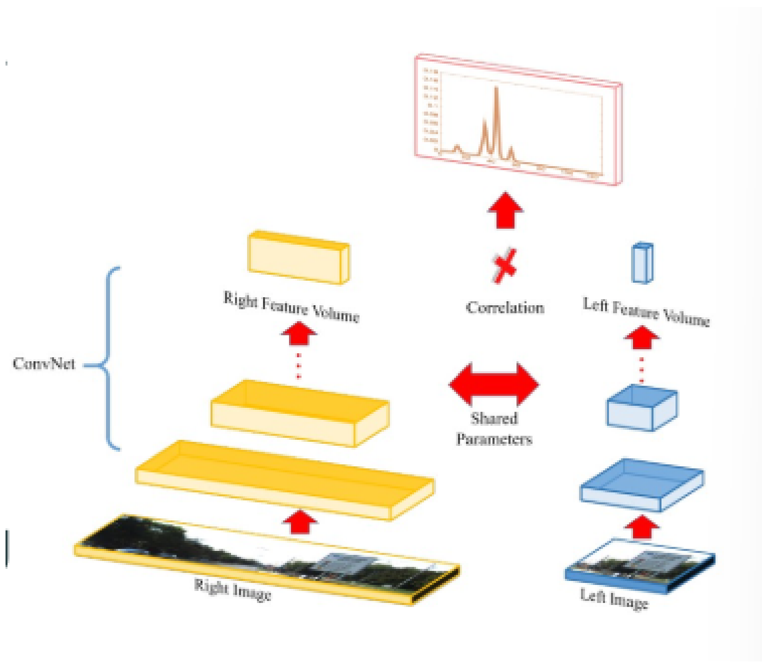
where s is the output of the network for one training example and t is the example label (1 if it is positive class, 0 otherwise).

### 3.8.4 MC-CNN Implementation Details

MC-CNN don't use the raw matching cost, but calculates the average of the costs over a fixed window of neighboring pixels. Use Semiglobal Matching to enforce smoothness constraints. Perform sub-pixel enhancement and smooth the disparity map by using bilateral filter: Smooth without blurring the edges.

## 3.9 Content-CNN

Inspiered by MC-CNN but without concatenation: just a dot product. Computes a 64-dimensional feature representation for every pixel of both images. For each possible disparity, match each pixel by calculating the dot product between corresponding features.



### 3.9.1 Content-CNN Loss Function

We train the network with cross entropy: for all possibloe patches (i) and all possible disparity values ($y_i$)

$$\min_{w} \sum_{i, y_i} p_{gt}(y_i) \log p_i(y_i, \mathbf{w})$$

$$p_{gt}(y_i) = \begin{cases} \lambda_1 & \text{if } y_i = y_i^{GT} \\ \lambda_2 & \text{if } |y_i - y_i^{GT}| = 1 \\ \lambda_3 & \text{if } |y_i - y_i^{GT}| = 2 \\ 0 & \text{otherwise} \end{cases}$$

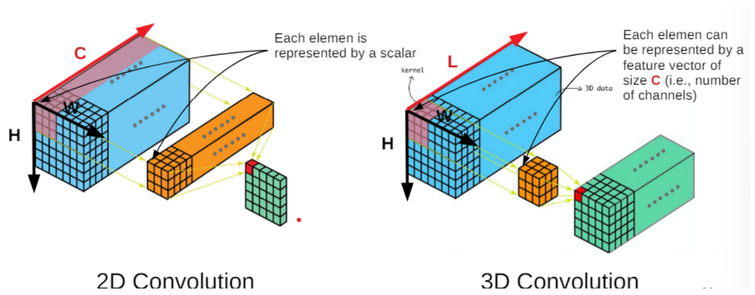
where $\lambda_1 > \lambda_2 > \lambda_3$

### 3.9.2 Content-CNN Implementation Details

The cost aggregation: average pooling over a 5x5 window. We use Semiglobal Matching to enforce smoothness constraints. Left-right consistency check and subpixel enhancement.

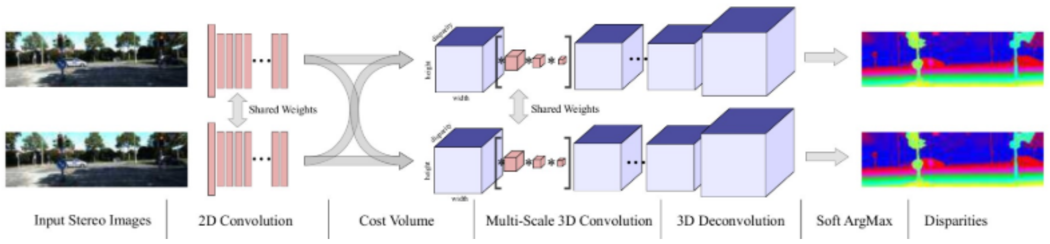
### 3.9.3 3D CNN on Volumetric Data

3D convolution uses possibly 4D kernels.


C
H
W
Each elemen is represented by a scalar
2D Convolution
L
H
W
kernel
3D data
Each elemen can be represented by a feature vector of size C (i.e., number of channels)
3D Convolution

## 3.10 GC-Net

GC-Net: **end-to-end** mapping from an image pair to disparity. It uses 3D convolutions to learn the cost volume. Regress sub-pixel disparity values from the disparity cost volume. Assemble a feature volume by



concatenating corresponding left and right features on each disparity.
We aim to compute a $W \times H \times D$ cost volume: we use 3-D convolutions to incorporate context and we extend the encoder-decoder idea to 3D, in order to increase feature's receptive field while reducing computation.

### 3.10.1 GC-Net: Estimate Disparity

Classical §Argmin operator over the cost volumes has two problem: no sub-pixel disparities and it is not differentiable. GC-Net propose to use a soft argmijn:

$$\text{soft arg min} := \sum_{d=0}^{D_{max}} d \times \delta(-c_d)$$

where

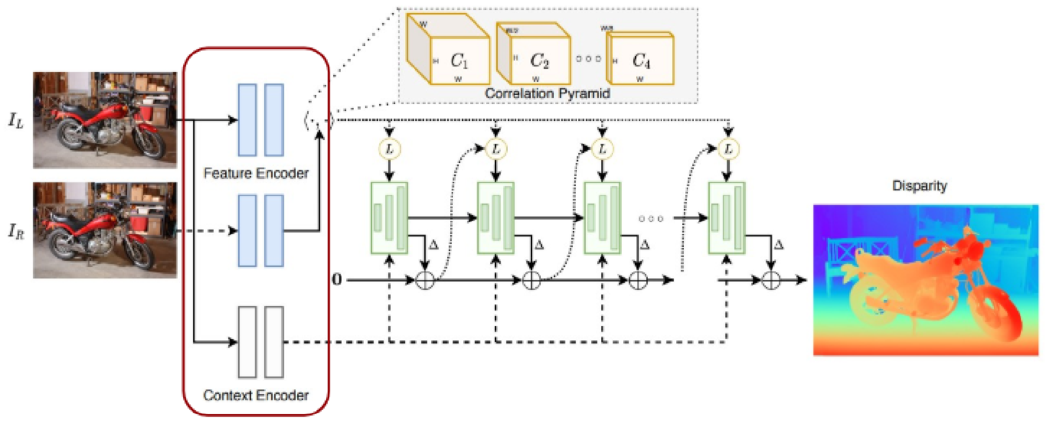
$$\delta(x_i) = \frac{\exp^{x_j}}{\sum_i \exp^{x_i}}$$

that is a weighted average of all possible softmax costs. The loss function is:

$$Loss = \frac{1}{N} \sum_{n=1}^{N} ||d_n - \hat{d}_n||_1$$

### 3.10.2 GC-Net Extensions

**SsSMNet (Self-Supervised Stereo Matching)** minimize the warping error without using ground truth disparities. It is online and has self-improving capabilities.



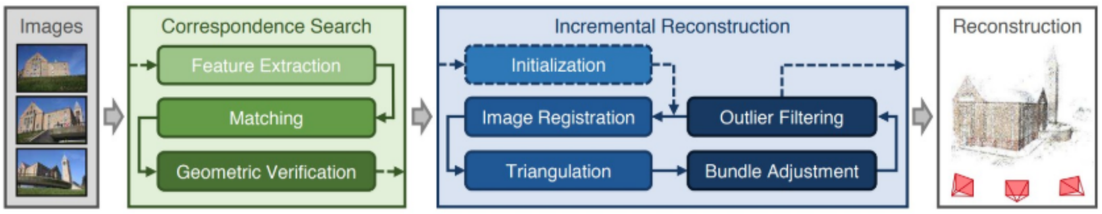
## 3.11 RAFT-STEREO

**RAFT-Stereo: Multilevel Recurrent Field Transform for Stereo Matching** uses only 2D convolutions and, also it is trained on synthetic data, and provides better generalization and accuracy. It is composed by a feature extractor, a feature correlation pyramid and a Gated Recurrent Unit (GRU) applied several times. Set of residual blocks and downsampling layers producing a feature per pixel. The cost volumes obtained by calculating the scalar products between the pairs of pixels corresponding to all disparities, "downscaled" in several pyramid levels.
The current estiamte of disparity are used to index the correlation volume, producing a set of correlation features. These features, the current dispairty map, and context features are then concatenated and injected into the GRU, that iteratively refine the disparity map.

# Chapter 4

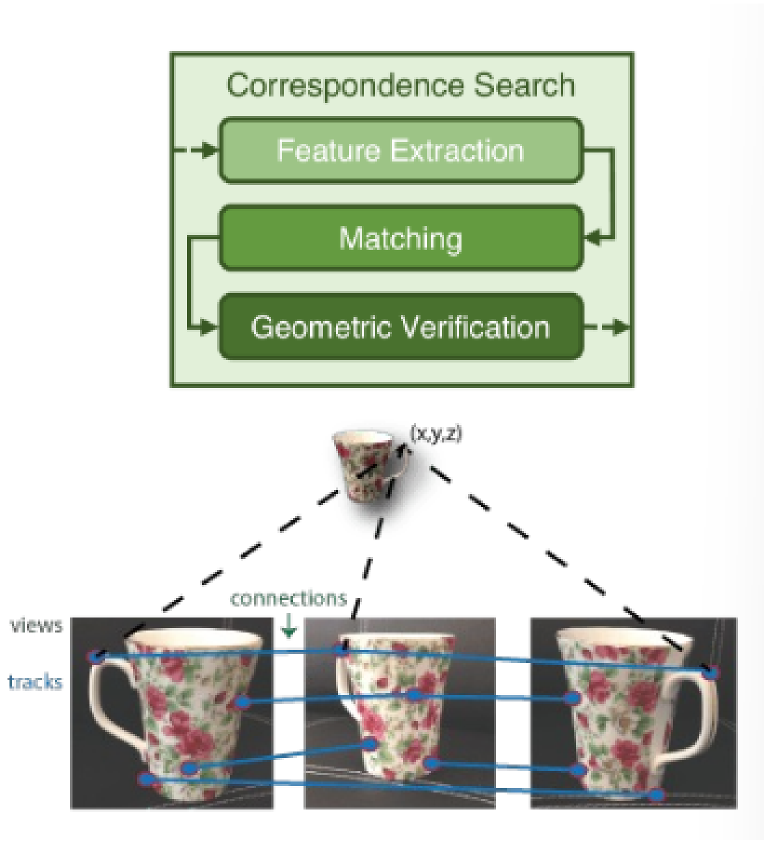
# Structure From Motion

Structure form motion is the process of reconstructing the 3D structure of a scene from its projections into a series of images taken from different viewpoints.
**Input:** a set of images that frame the same scene from different points of view. Each image should have some field of view overlap with other images: images can be totally unordered and taken with different cameras. The camera calibration parameters may not be available.
**Output:** camera positions, camera calibrations, and sparse 3D scene structure. Incremental SfM is a sequential



processing pipeline with an iterative reconstruction component.

## 4.1 Correspondence Search



For each image extract invariant features with associated descriptors. Match features between every image pair: this is just a naive approach since an exhaustive matching has computational complexiti $O(N_I^2 N_{F_i}^2)$. Many efficient approaches have been introduced to improve matching.

## 4.2 Geometric Verification

Try all models:

- Essentiam matrix E
- Fundamental matrix F
- Homography matrix H

For each model, use RANSAC to estimate the best estimate: Sample N minimal sets and find the maximum number of inliers that support the model. Select the best model E, F or H based on certain criteria: its inliers define the inlier matches between an image pair.

## 4.3 Scene Graph

After geometric verification, we can build a scene graph with images as nodes and verified pairs of images as edges. It provides correspondences between a pair of images, number of correspondences per image, number of observations in an image and the correspondence of an observation with all other images.

## 4.4 Incremental Mapping

Iterative try and error process. Reconstruction may never recover from a bad initialization and also from a bad image registration.

## 4.5 Initialization

Find a goot initial pair, possible basic algorithm:

- Sort images by decreasing number of correspondences, i.e., images with more correspondences appear in the front of the list.
- Select the first image in the list as the first seed image.
- Complete the seed pair with an image that is connected to the first seed image and has not been selected before.

Try to recover a rigid body transformation between the seed pair images, from the best model selected during geometric verification: in the uncalibrated case, this most likely leads to a ill-defined reconstruction. If failed, restart the initialization process. If the motion is a pure rotation estimated from H, restart the initialization process: No triangulation can be obtained from a pure rotation.

## 4.6 Initial Traingulation

Using point correspondences and the initial R,t traingulate point to obtain their 3D coordinates.
For a non rectilinear stereo rigs, no single solution due to noise. We have R,t and the calibrations parameters: given a 3D point, we know how to project it in both views! To use our points coordinates x, we need to normalize the homogeneous points:

$$x = \alpha P \tilde{X}$$

Cross product of two vectors of same direction is zero:

$$x \times PX = 0$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix} \Rightarrow x \times PX = 0$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix} = \begin{bmatrix} y p_3^T X - p_2^T X \\ p_1^T X - x p_3^T X \\ x p_2^T X - y p_1^T X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

The third line is a linear combination of the first and second lines.

$$\begin{bmatrix} y p_3^T - p_2^t \\ p_1^T - x p_3^T \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Use both points:

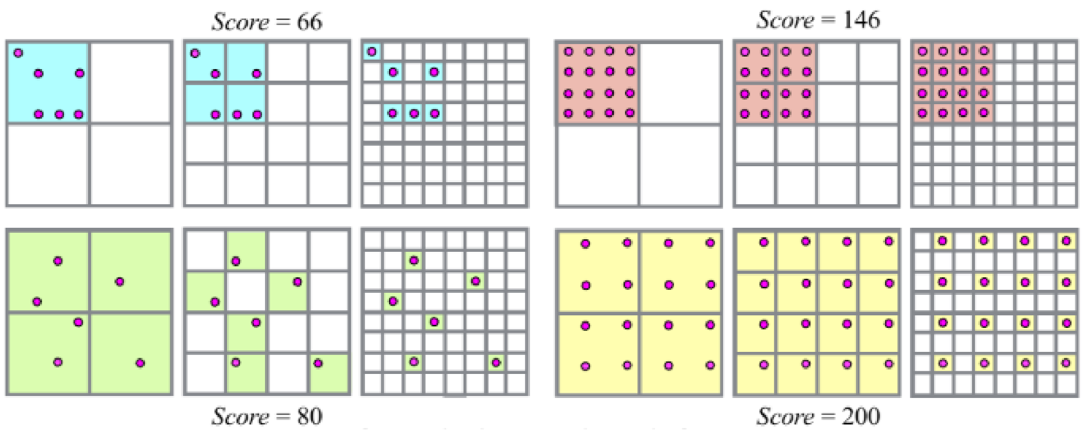
$$\begin{bmatrix} y p_3^T - p_2^t \\ p_1^T - x p_3^T \\ y' p_3'^T - p_2'^T \\ p_1'^T - x' p_3'^T \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow AX = 0$$

Is a homogeneous linear system. Use SVD and take the last column of V.

## 4.7 PnP-based Image Registration

A newly registered image must observe existing scene points. Use Perspective-n-Point using feature correspondences to triangulated points in already registered iamges. Estiamte both R',t' and intrinsic parameters. Use a modified version of PnP that exploits RANSAC and a minimal pose solver.
The newly registered image may also increase scene coverage by extending the set of points X trhough triangulation.

## 4.8 Next Best View Selection



Several criteria to select a still not registered image, among others:

- Sees the maximum of already registered points, or;
- Use visivility score.

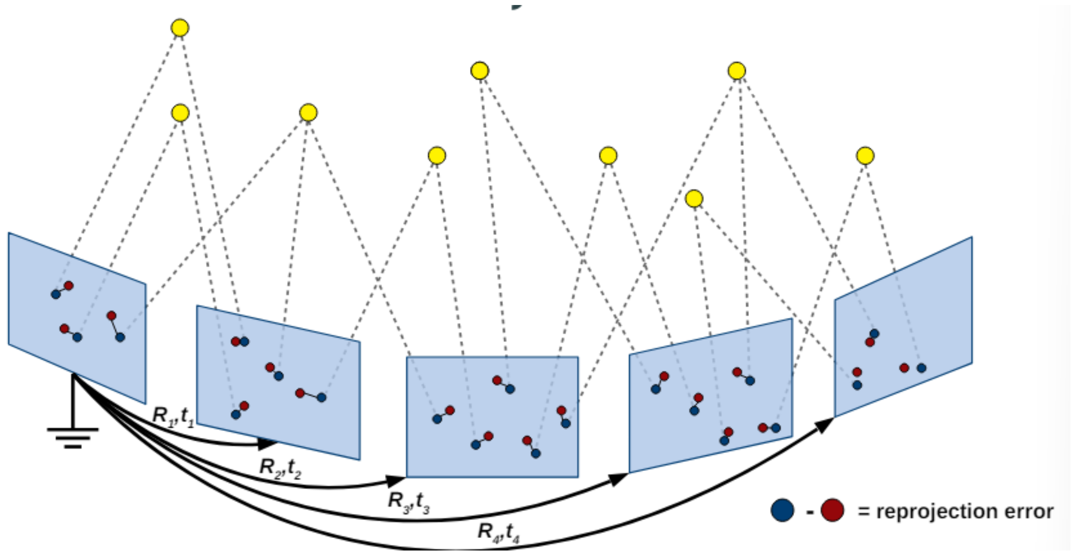
## 4.9 Bundle Adjustment

We may iterate with PnP and triangulation steps to register new images and new points but they are very correlated procedures: uncertainties in the camera pose propagate to triangulated points and vice-versa. **Bundle Adjustment (BA): joint non-linear refinement of camera parameters and point parameters.** BA goal: find a suitable parameters set (transformations $T_i = R_i, t_i$, calibration parameters $\theta_i$, 3D points positions $P_j, i = 0, ..., N_c - 1, j = 0, ..., N_p - 1$) that minimizes the sum of squares of reprojection errors (one for each observation pf each camera):

$$\arg \min_{\theta_0,...,\theta_{N_c}, T_1,...,T_{N_c-1}, P_0,...,P_{N_p-1}} \sum_{i=0,j=0}^{N_c-1, N_0(i)-1} \rho\big(||K_i f_i(A_n T_i \tilde{P}_{Obs(i,j)}) - \tilde{p}_{i,j}||^2\big)$$

Camera positions ($\# = 6(N_c - 1)$):

- Set the coordinate system of one of the two seed cameras as reference world frame



- Ese a suitable rotation representation

Calibration parameters (if not given)($\# = 3N_c$):

- Fix the principal point at the image center (principal point calibration is an ill-posed problem)
- Optimize for each camera the focal length in pixel and some distortion parameters

Scene points ($\# = 3N_p$): simply parametrized as 3D points.

### 4.9.1 BA Complexity

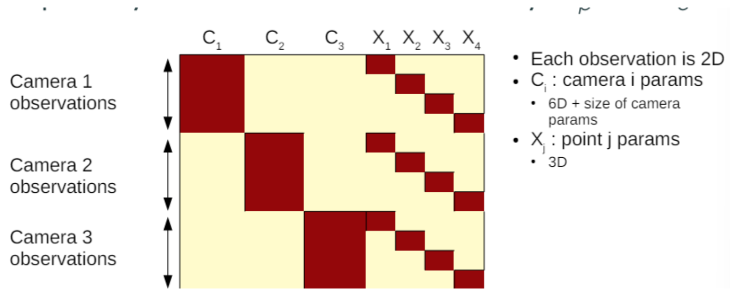
LM update:

$$\delta a^{LM} = -(J^T J + \lambda I)^{-1} J^T e(a)$$

with

$$J = \begin{bmatrix} \frac{\partial e_1}{\partial a_1} & \cdots & \frac{\partial e_1}{\partial a_m} \\ \vdots & \vdots & \vdots \\ \frac{\partial e_n}{\partial a_1} & \cdots & \frac{\partial e_n}{\partial a_m} \end{bmatrix} \qquad e_k = K_i f_i\big(A_n T_i \tilde{P}_{Obs(i,j)}\big) - \tilde{p}_{i,j}$$

## 4.10 Schur Complement Trick



The idea is to efficiently solve BA: exploit J sparsity and the fact that usually $N_p >> N_c$. We can rewrite the LM step as:

$$(J^T J + \lambda I)\delta a^{LM} = -J^T e(a)$$

and let defince $H = J^T J + \lambda I \Rightarrow H\delta a^{LM} = -g$.
Partitioning the step update between cameras and structure:

$$\begin{bmatrix} C & E \\ E^T & P \end{bmatrix} \begin{bmatrix} \delta a_c \\ \delta a_p \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix}$$

Multuply both sides by:

$$\begin{bmatrix} I & -EP^{-1} \\ 0 & I \end{bmatrix} \Rightarrow \underbrace{C - EP^{-1}E^T}_{\text{Schur complement of P}} \quad \delta a_c = v - EP^{-1}w$$

P block diagonal matrix: caluclating the inverse of P by inveerting each of its 3x3 blocks is cheap. Solve for $\delta a_c$ then backsubstituting it to obtain the value of $\delta a_p$. It has complexity of $O(N_c^3)$.

## 4.11 Robust Cost Function

Standard lest square assumes gaussian errors: high influence from high errors. To account for potential outliers, it is better to choose a proper loss function $\rho$.

# Chapter 5

# Visual SLAM

Visual Simultaneous Localization and Mapping has the goal of estimating the camera trajectory while reconstructing the environment in real-time. Viusla odometry focus on incremental estimation/local consistency, while Visual SLAM focus on globally consistent estimation.
SfM is more feneral than Visual SLAM and VO and tackles the problem of 3D reconstruction of both the structure and camera poses from possibly **unordered images** sets taken from possibly different and/or **uncalibrated cameras**. Visual SLAM is a particular case of SfM:

- it focuses on hte **sequential, real-time estimation** of the 3D motion and the 3D structures as a new frame arrives, by using a typically calibrated camera.
- Used for **robot navigation**.

## 5.1 Direct Vs Indirect Methods

**Indirect methods:**

- Images are pre-processed to generate an intermediate representation by means of point correspondences
- Correspondences are interpreted as noisy measurements inside an estimation problem.

**Direct methods:** Skips the pre-processing step and directly use the actual sensor values, i.e. light received from a certain direction over a certain time period.

## 5.2 Matching Vs Tracking

**Matching:** extract features for evvery image, compute a descriptor for each feature, and match them between images looking for similar descriptors.
**Tracking:** extract features in the first image; for each feature, look for a similar patch in a neighborhood on the next image; update the feature location and repeat from the previous step.

## 5.3 Dense Vs Sparse

Sparse methods use and reconstruct only a seected set of independent, salient points. Dense methods attempt to use and recconstruct all pixels in the 2D image domain. Semi-dense methods use and reconstruct largely connected and well-constructed subset of pixels.

## 5.4 Visual SLAM Pipeline Modules

The visual SLAM front-end is composed by position tracking and Local Mapping, while Visaul SLAM back-end is composed by Loop closure detection and Global Optimization.

## 5.5 Visual SLAM Front-end Approaches

We can use Epipolar geometry, filtering, Local bundle adjustment and direct approaches.

### 5.5.1 Epipolar Geometry

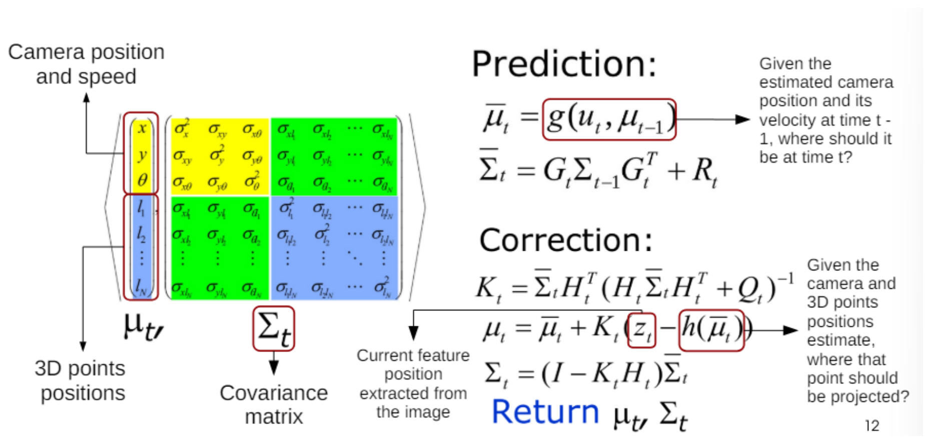
Extracct and match a set of salient features. It infers ego-motion using the epipolar geometry (essential, fundamental or homography matrix) inside a sample consensus framework. We estimate 3D features positions using tirangulation. We track consecutive motions with PnP and/or by expoloiting epipolar geometry constraints + scale registration.

### 5.5.2 Probabilistic Filtering

Define a state that includes camera position and features locations, along with other components used in the kinematic equations. For each image:

- Track a set of features;
- Update the state and **its uncertainty** using the kinematic equations;
- Predic the position of the tracked features after this motion update and compute the re-projection errors.
- ”Correcct” the state, e.g. using probabilistic tools.

#### Visual SLAM with Kalman Filter


Camera position and speed
3D points positions
$\mu_t$,
$\Sigma_t$
Covariance matrix
Prediction:
$\bar{\mu}_t = g(u_t, \mu_{t-1})$
$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
Given the estimated camera position and its velocity at time t - 1, where should it be at time t?
Correction:
$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
Return $\mu_t$, $\Sigma_t$
Current feature position extracted from the image
Given the camera and 3D points positions estimate, where that point should be projected?
12

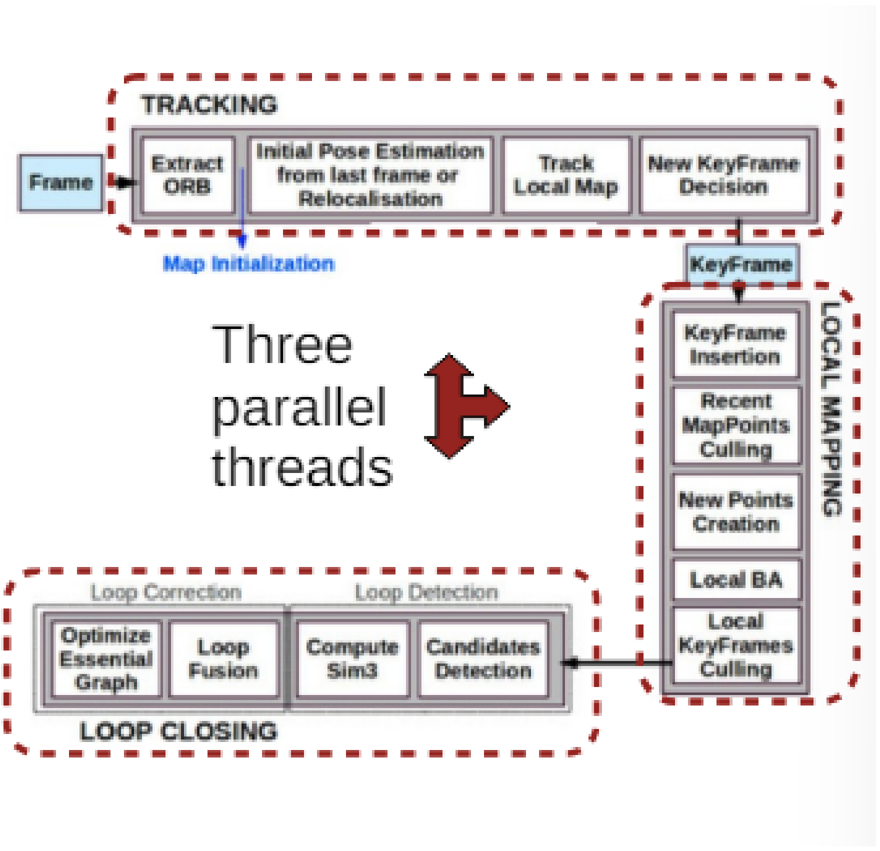
- Initialize first motion and first 3D points;
- Repeat:
  - **Predict** the camerra position at time t given the position and velocity estimates at time t-1
  - **Correct** the estimation by integrating features tracks;
  - Newly observed points should be integgrated in the state, points that are no longer visible should be removed.

The challenges of visual SLAM with Kalman Filters are that incorrect tracks lead the filter to divergence. We can track only up to a few hundred points. With a single camera the estimated motions are defined up to a scale factor, in addition we may have cumulative scale drift: a possivle solution is to use stereo vision or to add an IMU.

### 5.5.3 Local Bundle Adjustment

Initialize firsst otion and first 3D points. For each image:

- **Matching:** estimate feature positions and descriptors and match them with a previous **keyframe** (i.e., an image whose descriptors are stored for mapping and closure tasks)
- **Motion tracing:** estimate the current camera positon by solving a simplified bundle adjustment problem that only optimizes the camera positon (i.e., 3D points positions are fixed)
- **Local Mapping:** when required, select a new keyframe and estimate the camera positon along the 3D points positoons by solving a bundle adjustment pproblem over a fixed sequence of k newest keyframes.



### 5.5.4 ORB SLAM

ORB SLAM uses ORB features for position tracking, local mapping and loop-closure detection. ORB features for each keyframe are kept in memory for mapping andd loop-closure. Motion tracking is run at each frame andd local mapping is run when a new keyframe has been selected. Loop closure detection is tested at each keyframe. We can define two problems: Motion-Only Bundle Adjustment:

$$\{R, t\} = \arg\min_{R,t} \sum_{i \in \mathcal{X}} \rho\left(\left|\left|x^i_{(.)} - \pi_{(.)}(RX^i + t)\right|\right|^2_{\Sigma}\right)$$

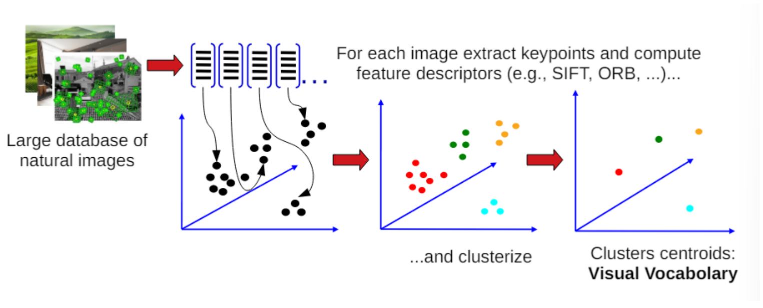
or Local Bundle Adjustment:

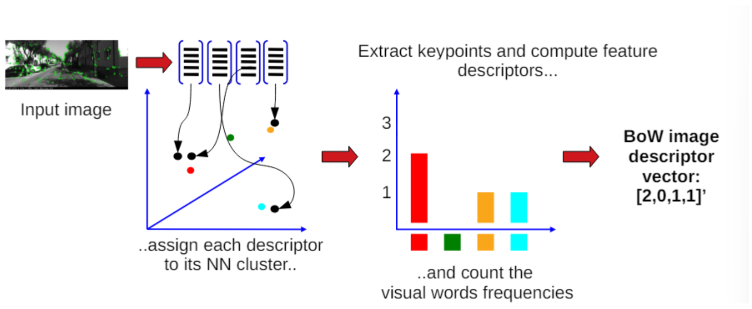
$$\{X^i, R_l, t_l | i \in \mathcal{P}_L, l \in \mathcal{K}_L\} = \arg \min_{X^i, R_l, t_l} \sum_{k \in \mathcal{K}_L \cup \mathcal{K}_F} \sum_{j \in \mathcal{X}_k} \rho(E_{kj})$$

$$E_{kj} = \left(\left|\left|x^j_{(.)} - \pi_{(.)}(R_k X^j + t_k)\right|\right|^2_{\Sigma}\right)$$

#### Loop-Closure Detection

A robust recognition is an essential requirement fo long-term Visual SLAM. Loop closures detections provide correct data association to obtain consistent maps. ORB-SLAM, as several other visual SLAM approaches, performs loop closure detection with Bag of Words (BoW) techniques. BoW summarizes the content of an image by the visual words it contains. visual words crrespond to a discretization of the descriptor space, known as the visual vocabulary.


Large database of natural images
For each image extract keypoints and compute feature descriptors (e.g., SIFT, ORB, ...)...
...and clusterize
Clusters centroids: Visual Vocabulary


Input image
Extract keypoints and compute feature descriptors...
..assign each descriptor to its NN cluster..
..and count the visual words frequencies
BoW image descriptor vector: [2,0,1,1]'
1
2
3

### 5.5.5 BoW-based Place Recognition

**Online place recognition:** compare the bag of words vector of the current image against the bag of words vectors of the images in the database (DB holds the BoWs of the **keyframes** collected so far). The image with the maximum individual score is considered as a loop candidate for the current image. A loop candidate is accepted if it is consistent with the previous k queries and it passes a geometrical check. ORB SLAM uses a vocabulary tree to speed-up images search.

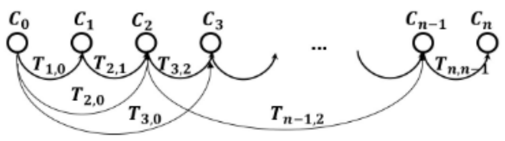
### 5.5.6 Loop Correction

Once a loop closure has been detected, the map should be correccted accordingly:

- Match the descriptos belonging to the current image $I_{cur}$ with the descriptors off the matched keyframe $I_{loop}$ representing the loop closure.
- Recover from the map the 3D points associated with the $I_{loop}$ descriptors.
- Recover the rigid body transformation **T=(R,T)** between $I_{cur}$ and $I_{loop}$ e.g., by registering $I_{curr}$ in the map with PnP.
- Use this transformation to correcct the full path: ORB SLAM like many other approaches achieves this by optimizing a **pose graph**.

### 5.5.7 Pose Graph Optimization

A pose graph optimizes the camera positions $C_0, C_1, ...$ associated to each collected keyframe. It only con-



siders the estiamtes of the rigid body transformations T between the nodes: minimize the sum off squared transformation errors in place of SfM reprojection errors:

$$\sum_{e_{ij}} ||C_i - T_{e_{ij}} C_j||^2$$

### 5.5.8 Direct Approaches

Camera motion and the scene structure are computed directly from **image intensity diiscrepancies using optimization techniques**. For each involved pixel:

- take note of its previous intensity $y_i$
- Un-project it back to 3D using the current depth guess $d_i$
- Re-project ontto the next image using the current transformation T guess
- Take note of its current intensity $y_i'$
- Compute the per-point residual $y_i - y_i'$
- Iteratively optimize T, $d_1, ..., d_n$ using least-square over such residuals

## 5.6 DSO: Direc Sparse Odometry

Direct method that jointly optimizes camera poses, sparse point positons, **camera intrinsics**, **photometric calibration**(such lens attenuation, gamma correction and exposure times). Optimization is performed in a sliding window of keyframes. Fast real-time CPU implementation.

### 5.6.1 DSO Photometric Error

Photometric error of a point p in a reference image $I_i$ observed in a terget image $I_j$ as weighted SSD over small neighborhood of pixel $N_p$:

$$E_{p_j} := \sum_{p \in \mathcal{N}_p} \left|\left| I'_j[p'] - \frac{t_j}{t_i} I'_i[p] \right|\right|$$

$$p' = \Pi_c(R\Pi^{-1}(p, d_p) + t) \qquad \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} := T_j T_i^{-1}$$

The full photometric error over all considered frames and points is given by:

$$E_{photo} := \sum_{i \in \mathcal{F}} \sum_{p \in \mathcal{P}_i} \sum_{j \in obs(p)} E_{pj}$$

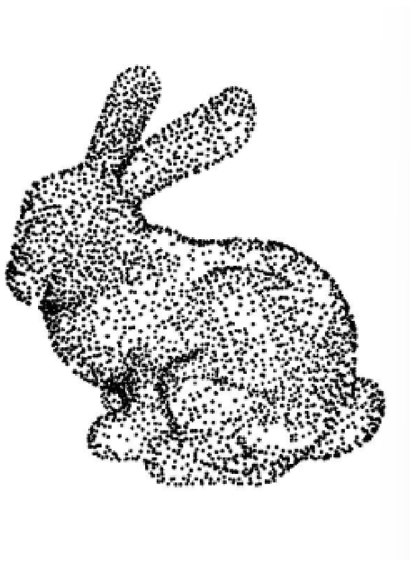
Chapter 6

# 3D Data Representation

## 6.1 Depth maps and Range Images

A depth map contains the distances of points in the scene to a plane. a range image contains the distance of points in the scene to another point.

## 6.2 Point Clouds



Set of 3D vertices with coordinates (x,y,z). Common raw output of several 3D scanners. Generally unstructured, arbitrary points order. we have no information about vertex connectivity. No clue about the underlying surfacce from which the points are sampled. Any permutation of the points yields the same point cloud.
The connectivvity of a given point can be approximated with its k-nearest neighbors. Surfacce normals can be estimated by assuming piecewise planar surfaces and ffitting a plane at one point and some of its neighbors.

### 6.2.1 Organized Point Clouds

Also called structured point clouds are in 2-D matrix of 3D points: mxnx3 array and, for each i=1,...,m and j=1,...,n, there is a corresponding 3D vertex $v_{ij}$. Points are organized according to some spatial relationships between them. Very efficient memory layout. We can use an additional 2D arrays of size mxn: presence or absence oof 3D data, confidencce values, point clors and so on.
There is an implicit mesh connectivity between neighbouring points, and single non-booundary points are always degree 6.

## 6.3 Meshes

Represent or approximate a 2D manifold in 3D space. A set of vertices connected to each other in order to form some polygons. **triangula mesh:** very common data structure usually employed by 3D renders: each triangle has a single surface normal. for eaach surface point it is possible to get a normal vector e.g., from the vertices normals or directly from the face normal. Normals can then be used by rendering algorithms.

### 6.3.1 Manifolds

In mathematics, a manifold is a geometrical space not necessarily equipped with a metric to measure distances that locally resemples a Euclidean space. Each point of n-dimensional manifold has a neighborhood that is homeomorphic to an open subset of n-dimensional Euclidean space. one-dimensional manifolds includes lines and cirlces. Two-dimensional manifolds also called surfaces, i.e. planes, spheres and the torus.

### 6.3.2 Triangular Mesh

**Closed:** the surface coompletely encloses a volume. **Open:** the mesh contains bboundary edges which are used by only one triangle. **Genus** of a surface: number of cuts we can make by means of closed, non-intersecting curves without disconnecting the manifold.
It is possible to associate a color to vertices of a mesh. The points that are inside a traingle can be painted with a color obtained by interpolating the colors of the vertices of the triangle.
Often, meshes are augmented with texture coordinates used to map from the surface to a 2D planar image. Texture ccoordinates range from 0 to 1 in the x and y axis.

### 6.3.3 Implicit Surfaces

The set of all points which satisy the function f(x,y,z)=0. Typically, function values greater than zero indicate a point which is outside the object, whiile negative values indicate a point which is inside; this is known as a **Signed Distance Function (SDF)**. In general, obtaining a function f() which exactly describes a general surfacce is difficult.

## 6.4 Voxels

Voxels are the 3D analogue of pixels. At each voxel, a booleean or a scalar is stored which indicates the rpesence or absence of the volume at that location. Voxels grids are used to represent properties at discrete spatial locations over a 3D volume. Precision depends on resolution. Data storage requirements are high. Can be easily used wwithin deep learning models.

## 6.5 Octrees

A space-efficien representation for voxel grids. It uses adaptive resolution depending on the complexity of the surface throughout the volume. Idea; subdivide voxels which are not fully occupied by the volume or fully empty. Represented by trees: the root represents the complete volume. Nodes have eight children. Leaves are fully occupied r empty voxels.

## 6.6 Volumetric signed Distance Function

The idea is to represent a 3D environment as a voxel gridd in which each voxel stores the signed distance to the nearest surfacce. The surface can be recovered by interpolation. In order to represent non-convex or multiple, disconnected surfaces, the SDF can be truncated between a minimum and maximum value (TSDF).

## 6.7 Surfacce Normals

Surface normals are important properties of a geometric surfacce used in several problems: 3D shape recognition; 3D ego-motion estiamtion/reconstruction from point-clouds; Object pose estiamtion; Surface classification; In computer graphcs to apply the correct light sources that generate shadings and other visual effects.
If the surface is truly piecewise planar associate a face normale $n_f$ to each traingular facet.

$$n_f = \frac{(v_2 - v_1) \times (v_3 - v_1)}{||(v_2 - v_1) \times (v_3 - v_1)||}$$

If the surfacce is assumed smooth the surface normal is approximated at each vertex by means of weighted averages of the adjacent facet normals.

### 6.7.1 Surface Normals from Point Clouds

No information about vertex connectivity. The idea is that for each point $p_i$ we select k-nearest neighbors and fit a plane. Given a point cloud $P = \{p_1, ...p_n\}, p_i \in R^3$, for each $p_i$ we would like to estimate a normal vector

$n_i$ from a set of k points in its neighborhood plus the point $p_i$ itself:

$$Q_i^+ = [p_1, q_{i1}, ..., q_{ik}]^T$$

Idea 1: fit a local plane to $Q_i^+$:

$$S_i = n_{ix}x + n_{iy}y + n_{iz}z + d$$

i.e, solve:

$$\min_{b_i} \left|\left|[Q_i^+ \; 1_{k+1}]b_i\right|\right|_2$$

with

$$b_i = [n_i^T, d]^T$$

Solved by computing SVD for $Q_i^+$ and taking the last column of V.
Idea 2: **minimize the variance** by removing the empirical mean from the data matrix $Q_i^+$:

$$\min_{n_i} \left|\left|[Q_i^+ \; \bar{Q}_i^+]n_i\right|\right|_2$$

we may use SVD to solve this system.

### 6.7.2 PCA-Recup

Orthogonal projection of data onto a lower dimension linear space that maxmizes varaince of projected data and minimizes mean squared distance between original data points and their projecction.

## 6.8 K-D trees

Brute force k-nearest neighbor search has $= (Kn^2)$: **K-dimensional trees** can be used for organising points to speed up this search to $O(kn \log n)$. A K-D tree is a binary tree in which every leaf node is represented by one or more K-dimensional point. K for 3D point clouds is just 3. A K-D tree is a space partitioning data structure for organizing points. Partitioning is done by setting planes orthogonal to one of the axes.
This simplified implementation proveides an **approximated NN**, since the actual NN may not be included in the reached leaf. The algorithm should be modified by adding a backtracking step to find the exact nearest neighbour. The tree should be as balanced as possible. Select the better dimension where operate the split.

### 6.8.1 Example of K-D Tree Construction

Construction for 1-NN: built the K-D tree from all 3D points pick a random dimension, find the median, split the data, repeat.

### 6.8.2 Example: K-D Tree NN Query

For each query point, find the NN by navigating the whole tree up to a leaf.

# Chapter 7

# 3D Local Descriptors

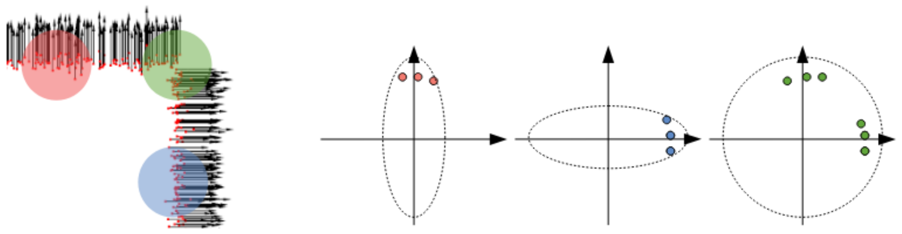
A 3D local descrptor (or 3D **local feature**) is a compact representation oof the geometric properties of a point in its neighborhood: just a **n-dimensional vector**. Extension to 3D data f the image features paradigm. Typically a subset of salient points (**keypoints**) is selected on which to calculate descriptors, e.g. a set of ineteresting, stable points and a regularly downsampled subset of the point cloud. The goals are **Descriptiveness**, i.e. capture predominant traits of the surface; and **Robustness**, i.e. obtain a similar descriptor also in the presence of nusances like noise, variations of point density and viewpoint, small occlusions, small deformations, etc. . .

## 7.1 3D keypoint Detection

Keypoints in point clouds are slightly less stable than keypoints in images, hence often they are just regularly subsampled from the original cloud. Simple 3D point extracction methods have been introduced by drawing inspiration from keypoint extracion in images.

### 7.1.1 3D Harris Keypoints

Uses the idea of 2D harris keypoints, but instead of using image gradients, it uses **surface normals**. Harris corner detector applies Taylor's expansion to approximate E with first-order derivatives. In 3D, for every possible small displacement $\Delta x, \Delta y, \Delta z$, the normals of the points enclosed by a sphere should clearly change. Given a sphere of radius R surrounding the tested point p and given $n_i$, the normals of the k points enclosed in



the sphere, i=1,...,k, calculate the autocorrelation matrix A:

$$A = \frac{1}{k}\sum_{i=1}^{k} n_i \cdot c_i^T = \frac{1}{k}\begin{bmatrix} \sum_{i=1}^{k} n_{i,x}n_{i,x} & \sum_{i=1}^{k} n_{i,x}n_{i,y} & \sum_{i=1}^{k} n_{i,x}n_{i,z} \\ \sum_{i=1}^{k} n_{i,y}n_{i,x} & \sum_{i=1}^{k} n_{i,y}n_{i,y} & \sum_{i=1}^{k} n_{i,y}n_{i,z} \\ \sum_{i=1}^{k} n_{i,z}n_{i,x} & \sum_{i=1}^{k} n_{i,z}n_{i,y} & \sum_{i=1}^{k} n_{i,z}n_{i,z} \end{bmatrix}$$

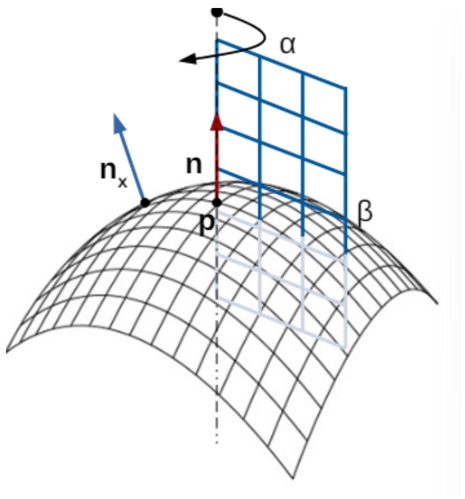
Calculate some response from A. Remembering that $det(A) = \lambda_1\lambda_2\lambda_3$ and $tr(A) = \lambda_1 + \lambda_2 + \lambda_3$, possible responses are:

$$R = \min\{\lambda_1, \lambda_2, \lambda_3\}, \qquad R = \frac{det(A)}{tr(A)}, \qquad R = det(A) - ktr(A)^2$$

## 7.2 Spin Images

**Image:** a 3D point is described by a 2D grid; **Spin:** the grid is rotated around the keypoint normal, counting the number of points that fall into each cell. The parameters are the bin size (descriptiveness), the image size (locality) an the support angle ($\text{acos}(nn_x I) < th$)(outlier filtering.)
Given an oriented keypoint (p,n), map 3D neighbors points x to 2D points in the $(\alpha, \beta)$ of spind images:

$$S_o(x) \rightarrow (\alpha, \beta) = (\sqrt{||(x-p||^2 - (n(x-p))^2}, n(x-p))$$



Accumulate 2D points $(\alpha, \beta)$ in discrete bins. The discretization of the $\alpha - \beta$ domain makes spin images very sensitive to noise. The solution is tat each point $(\alpha, \beta)$ contributes to the our surrounding bins in the 2D array by using bilinear interpolation.

## 7.3 3D Shape Context

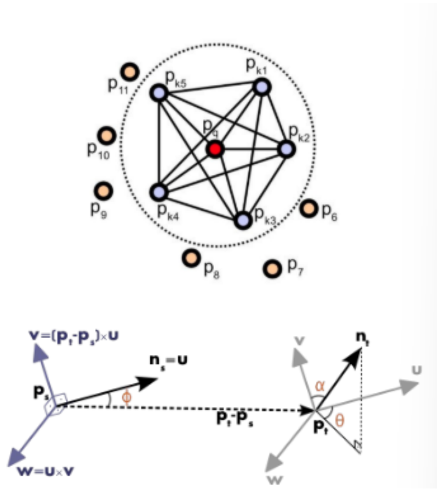
Extension of the 2D shape context descriptor to point clouds. At each keypoint, organize a 3D grid divided into bins along the log-raadial, azimuth, and elevation dimensions. the grid size is $d_r \times d_a \times d_e$. Counts the number of points falling in each bin and normalize. The north pole is aligned with the point normal. 3DSP invariant up to a rotation around the z-axis: multiple descriptors for a single feature point required to match across different views of a surface. 3DSP improved by fixing a repeatable and unambiguous local reference frame.

## 7.4 Point Feature Histogram

PFH encodes directional relationships between points in a local neighborhood and their surface normals. for each keypoint p, consider the k neighbors in a sphere of radius r. For each pair of points of the neighborhood define one point $p_s$ as source and the other $p_t$ as target, the source being the one having the smaller angle between the associated nromal and the segment $p_t - p_s$ connecting the points. Define a local ccoordinate system centered in $p_s$ with:

$$u = n_s \qquad v = u \times \frac{p_t - p_s}{||p_t - p_s||}, \qquad w = u \times v$$

Using the above uvw frame, the difference between the two normals $n_s$ and $n_t$ can be expressed as a set of

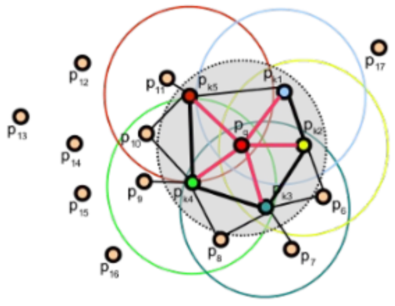


angular features as follows:

$$\alpha = < v, n_t >, \qquad \phi = < u, p_t - p_s > /d, \qquad \theta = \arctan(< w, n_t >, < u, n_t >)$$

Dkiscretize each feature in b intervals and count the occurences for each bin. Normalize the histogram. Typically 4 intervals. Distances between points are sometimes used as features. The complexity is $O(k^2)$.

## 7.5 Fast Point Feature Histogram



PFH is computationally expensive and not suitable for real-time applications. FPFH is a simplified variant of PFH that computes the PFH features as $\alpha, \phi, \theta$ only for pairs that share a reference point. This is called Simplified Pint Feature Histogram (SPF).

- For each keypoint $p_q$ comute the SPF,i.e. the PFH just considering the pairs $(p_q, p_i)$ for each $p_i$ belonging to the k neighbors of $p_q$
- For each point $p_i$, find its k neighbors and compute its SPF.
- The FPFH descriptors is obtained by summing hte SPF of $p_q$ with a distance-weighted sum of the SPFs of its k neighbors:
$$FPFH(p) = SPF(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k}SPF(p_k)$$

The default FPFH implementation uses 11 binning subdivisions. FPFH does not fully interconnect all neighbors of $p_q$. The PFH models a precisely determined surface around the keypoint, while the FPFH includes additional point pairs outside the r radius sphere. Compleity of O(k), suitable for real-time applications.

## 7.6 SHOT Descriptor

SHOT: Unique Signatures of Histograms for Local Surface Description. For each keypoint we extract a unique and repeatable local reference frame and we comose a descriptor by concatenating a set of local histograms that report angularr differences statistics between points normals.

### 7.6.1 SHOT: Local Reference Frame

Compute the 3x3 covariance matrix M from the k-nearest neighbors $p_i$ of the keypoint p:

$$M = \frac{1}{k}\sum_{i=0}^{k}(p_i - \hat{p})(p_i - \hat{p})^T = \frac{1}{k}\sum_{i=0}^{k}$$

In order to increase repeatability in presence of clutter, one can assign to distant points smaller weights, i.e.:

$$M = \frac{1}{\sum_{i:d_1 \le R}(R - d_i)}\sum_{i:d_i \le R}(R - d_i)(p_i - p)(p_i - p)^T, \qquad d_i = ||p_i - p||_2$$

The 3 eigevenctors of M define, respectively, the **directions** of the x,y,z axes of the local reference fframe. Their sign is not defined unambiguously. SHOT reorients the sign of each eigenvector of M so that its sign is coherent with the majority of the vectors it is representing, i.e.:

$$S_x^+ = \{i : d_i \le R \wedge (p_i - p)x^+ \ge 0\}, \qquad S_x^- = \{i : d_i \le R \wedge (p_i - p)x^- > 0\}$$

$$x = \begin{cases} x^+, & |S_x^+| \ge |S_x^-| \\ x^-, & \text{otherwise} \end{cases}$$

### 7.6.2 SHOT: Descriptor computation

SHOT divides the spherical support around the keypoint with a 3D spherical grid, partitioned uniformly along the radial, azimuth, and elevation axes. The grid is aligned with the axes given by the estimated local reference frame.
For each grid's bin, compute a local histogram that cosiders only points falling into such bin. Each histogram bin defines a discretized set of angles $\theta_i$, i=1,...,n between the normal n at eacch point and the z-axis of the local reference frame: just counts the number of points that have normal that forms the angle $\theta_i$ with the z-axis.
Compose the descriptor by concatenating all the histograms: the deffault implementation partitions 3D spherical grid into 32 bins and the local histograms into 11 bins: 352-dimensional descriptors. To increase the robustness against small displacements, the contribution of the points is interpolated between neighboring bins.

## 7.7 3DMatch

One of the fiirst methods that learns a 3D local feature descriptor with deep learning. The input point cloud or mesh is locally voxellized around keypoints. Each voxel stores the Truncated Signed Distance Function **(TSDF)** value. A **Siamese 3D CNN** is trained with a **contrastive loss function** that minimizes the L2 distance between 512-dimensional descriptors generated from matching points and maximizes the L2 distance between non-corresponding points.

## 7.8 CGF

Compact Geometric Features uses a handcrafted, very high-dimensional descriptor to represent the rawlocal geometry around points: 3D Shape Context aligned with SHOT local reference frame. Train a depp network for dimensinality reduction by using standard triplet loss.

## Chapter 8

# 3D Shape Registration

3D shape registration is the process of finding a rigid body transformation that aligns two or more 3D shapes. Registration can also be done for deformable, non-rigid shapes or for different but similar objects, finding for example some extra transformation parameters. Given two or more 3D shapes we want:
If a set of correct correspondences between them is given, extract the transformation which best aligns the shapes. If an initial alignment between them is given, perform a registration refinement to find a more accurate alignment. Usually the above two tasks are performed in ssequence, to mitigate small misalignments caused by non-ideal matches.
**Problem statement:** given two point clouds D and M

$$\mathcal{D} = \{d_1, ..., d_{n_\mathcal{D}}\}, \mathcal{M} = \{m_1, ..., m_{n_\mathcal{M}}\}$$

We want to find R,t that minimize:

$$E(R,t) = \sum_{i=1}^{n_\mathcal{D}} \left|\left|m_{j(i)} - (Rd_i + t)\right|\right|^2$$

where $d_i$ and $m_{j(i)}$ are corresponding points, i.e. $d_i \leftrightarrow m_{j(i)}$.

## 8.1 Correspondence-based Registration

If the **correct** correspondences are known (i.e., correcct **data association**), the optimal transformation can be calculated in closed form. We assume here for simplicity $n_\mathcal{D} = n_\mathcal{M} = n$ with the cloud points associated in order.

## 8.2 Decoupling R,t

Point clouds centroids:

$$d_c = \frac{1}{n}\sum_{i=1}^{n} d_i, \qquad m_c = \frac{1}{n}\sum_{i=1}^{n} m_i$$

Given an optimal solution of the minimization problem:

$$\{\hat{R}, \hat{t}\} = \arg\min_{R,t} sum_{i=1}^{n} \left|\left|m_i - (Rd_i + t)\right|\right|^2$$

The centroid of the transformed cloud is:

$$\hat{d}_c = \frac{1}{n}\sum_{i=1}^{n}(\hat{R}d_i + \hat{t}) = \hat{R}\frac{1}{n}\sum_{i=1}^{n} d_i + \hat{t} = \hat{R}d_c + \hat{t}$$

**Centroid Coinciidence Theorem:** $m_c = \hat{d}_c$.

$$m_c = \hat{R}d_c + \hat{t} \Rightarrow \hat{t} = m_c - \hat{R}d_c$$

Replace t in the residual and you'll obtain:

$$m_i - (\hat{R}d_i + \hat{t}) = m_i - (\hat{R}d_i + m_c - \hat{R}d_c) = (m_i - m_c) - \hat{R}(d_i - d_c)$$

Subtracting the centroids from each point in th etwo point clouds leads to a reduced minimization problem:

$$\hat{R} = \arg\min_{R} \sum_{i=1}^{n} ||m_i' - Rd_i'||^2$$

To sum up:

- Compute the centrooids and subtract them from each point in the two point clouds:

$$d'_i = d_i - d_c, \qquad m'_i = m_i - m_c$$

- Find the rotation that minimizes:

$$\hat{R} = \arg\min_R \sum_{i=1}^{n} ||m'_i - Rd'_i||^2$$

- Find the translation in closed form:

$$\hat{t} = m_c - \hat{R}d_c$$

## 8.3 Finding R

**Theorem:** let

$$W = \sum_{i=1}^{n} m'_i d_i^{T'}$$

From its SVD $W = U\Sigma V^T$. It is possible to find an ptimal solution for R as: $\hat{R} = UV^T$. If $det(UV^T) = -1$, then the solution is:

$$\hat{R} = U\text{diag}(1, 1, -1)V^T$$

## 8.4 Correspondence-free registration

If the correct correspondences are not known, it is required to jointly estimate both the correspondences and the transformation.

## 8.5 Iterative Closest Point Algorithm

**ICP** idea: iterate to find good correspondneces and a good alignment. Correspondences not required. Requires a good initial guess for R,t. ICP is guaranteed to converge monotonically to a local minimum of E(R,t).

### 8.5.1 Basic ICP Algorithm

I Using the current R,t estimate, determine the correspondences $d_i \leftrightarrow m_{j(i)}$ as:

$$j(i) = \arg \min_{k=1,\ldots,n_{\mathcal{M}}} ||m_k - (Rd_i + t||^2$$

II Compute R,t using the new correspondences

III Compute the new error:

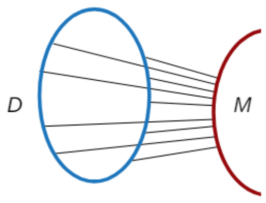
$$E(R, t) = \sum_{i=1}^{n_{\mathcal{D}}} ||m_{j(i)} - (Rd_i + t)||^2$$

IV If error E decreased and $E >$ threshold, return to point I

V Otherwise stop and return the final transformation.

### 8.5.2 Assumpitons

**A1:** the two veiws must be close to each other. **A2:** the two clouds must fully overlap, or D must be a subset of M, otherwise we'd have spurious correspondences.



#### A1 Prealignment

Often 3D descriptors computed on a subset of points are used to find matches between clouds and hence estimate an initial, coarse traansformation: correspondece-based registration and RANSAC can be used to reject outlier matches. Then we can forget the matches and use ICP to refine the transformation: Correspondence-free registration.

#### A2: Outlier Rejection and Segmentation

Reject correspondences with point-to-point distance higher than a given threshold. Reject correspondences that are not consistent with their neighboring.
Segment the clouds in clusters using some criteria, align pairs of clusters, take the alignment that gets the lower error, try to merge subset of clusters.

### 8.5.3 Speed-up ICP

#### Normal-Space Subsampling

Choose points such that the distribution of normals among the selected points is as spread as possible.

#### Distance Formulation

With point-to-plane distance in place of point-to-point distance ICP converges quickly

#### Additional Information

ICP accuracy can be improved by considering additional information of points and verifying the compatibility of the associated points: accept only matches between compatible points. The compatibility can be based on normals, curvature, other local features or colors.

### 8.5.4 ICP with LM

The registration error:

$$E(R,t) = \sum_{i=1}^{n_{\mathcal{D}}} ||m_{j(i)} - (Rd_i + t)||^2 = \sum_{i=1}^{n_{\mathcal{D}}} e_i(R,t)^T e_i(R,t)$$

with:

$$e_i(R,t) = m_{j(i)} - (Rd_i + t)$$

can be directly minimized using general-purpose nonlinear optimization, such as Levenberg-Marquardt algorithm.
It has the following properties:

- Wider convergence basin
- Usually, more accurate solutions
- Can handle general non-rigid transformations
- It is possible to adopt robbust loss function for outlier rejection/wider basin of convergence
- Can handle wieghted residuals
- Comparable in speed to special-purpose ICPs sch as SVD-based ICP.

### 8.5.5 Weighted Residuals

Using LM, each residual can be weighted as 3x3 positibe definite matrix $\Omega_i$:

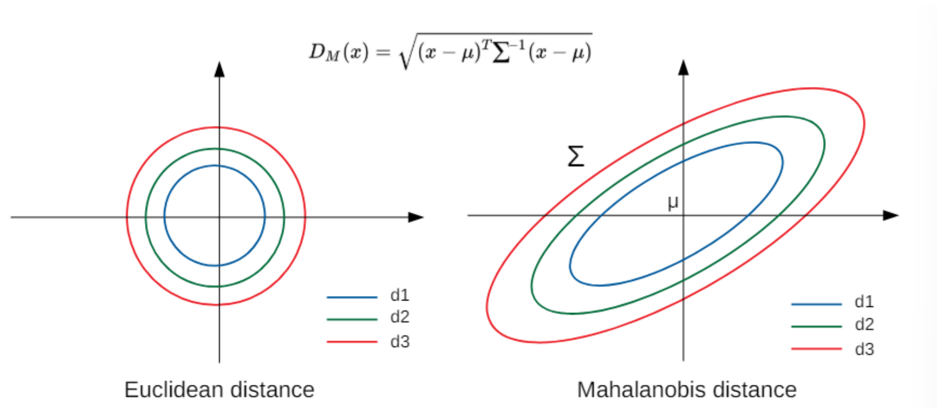
$$E(R,t) = \sum_{i=1}^{n_{\mathcal{D}}} ||m_{j(i)} - (Rd_i + t)||^2_{\Omega_i} = \sum_{i=1}^{n_{\mathcal{D}}} e_i(R,t)^T \Omega_i e_i(R,t)$$

E.g.m a diagonal $\Omega_i = diag(W_i, W_i, W_i)$ matrix can be used to assign lower wieghts for points with higher point-to-point distances. More generally $\Omega_i$ is defined as the information matrix and can be set to the inverse of some measurement error covariance matrix $\Sigma_i^{-1} = \Omega_i$. The distance:

$$||m_{j(i)} - (Rd_i + t)||_{\Omega_i} = \sqrt{e_i(R,t)^T \Omega_i e_i(R,t)} = \sqrt{e_i(R,t)^T \Sigma_i^{-1} e_i(R,t)}$$

reminds the **Mahalonobis distance**

$$\sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)}$$


$D_M(x) = \sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)}$
Σ
μ
d1
d2
d3
d1
d2
d3
Euclidean distance
Mahalanobis distance