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 9×1 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:

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

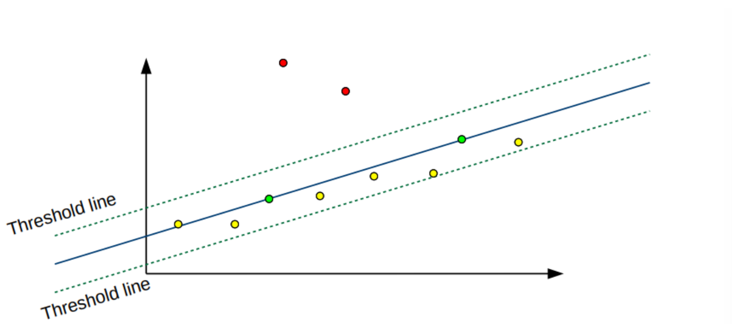
1.3.5 Image to Image Homography

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 ϵ is the fraction of outliers and s is the number of required correspondences to compute a solution. We have:

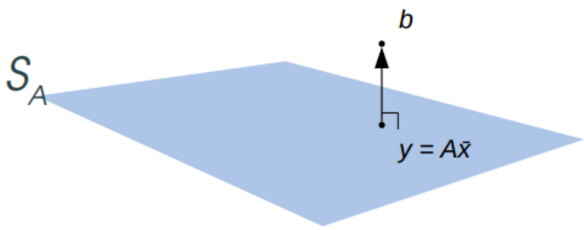
$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^S)}$$

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 δa 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 δa 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 λ 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:

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

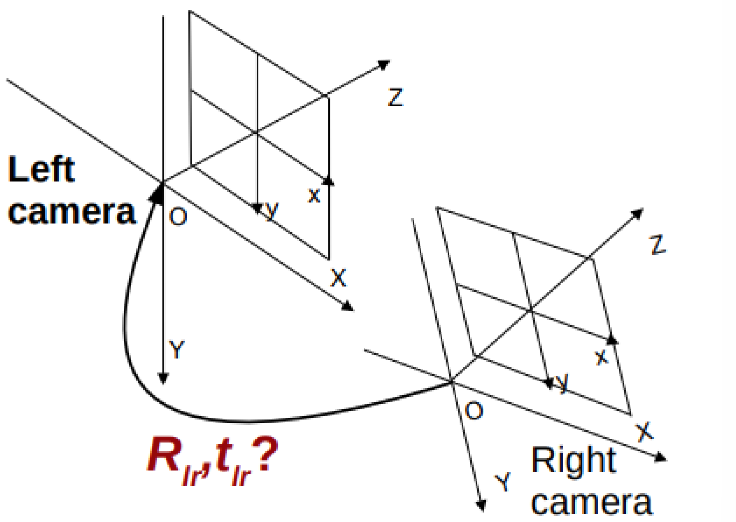
The ideas for the minimization: we use the LM algorithm to solve the iterative problem; we parametrize the rotations with **axis-angle representation**

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

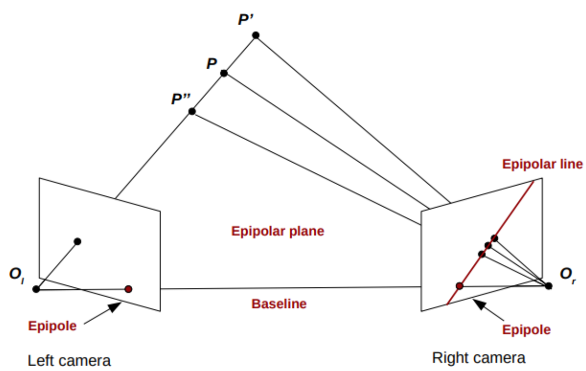
Epipolar Geometry



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:

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

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 $\tilde{p}_l(t_{lr} \times R_{lr} \tilde{p}_r) = 0$.



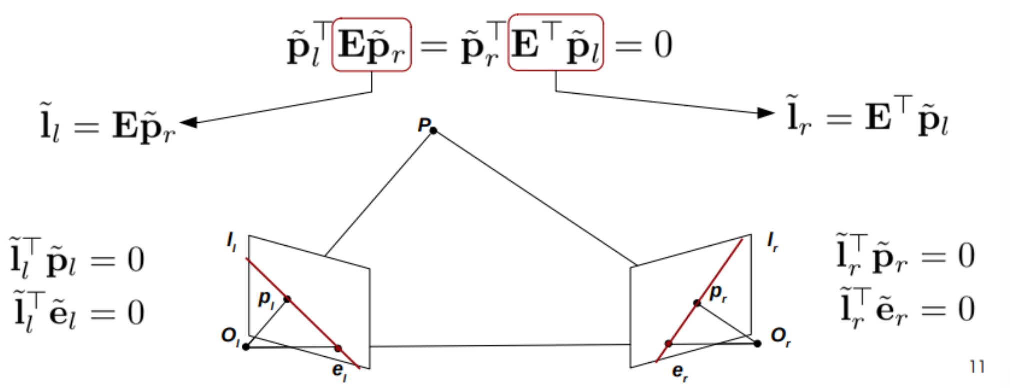
From there we can obtain the following constraint: $\tilde{p}_l^T \hat{t}_{lr} R_{lr} \tilde{p}_r = 0$, obtaining the essential constraint:

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

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**



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 3×3 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 σ 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:

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

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

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:

$$\begin{bmatrix} x'_1x_1, x'_1y_1, x'_1, y'_1x_1, y'_1y_1, y'_1, x_1, y_1, 1 \\ x'_nx_n, x'_ny_n, x'_n, y'_nx_n, y'_ny_n, y'_n, x_n, y_n, 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \Rightarrow Af = 0$$

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 α 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.

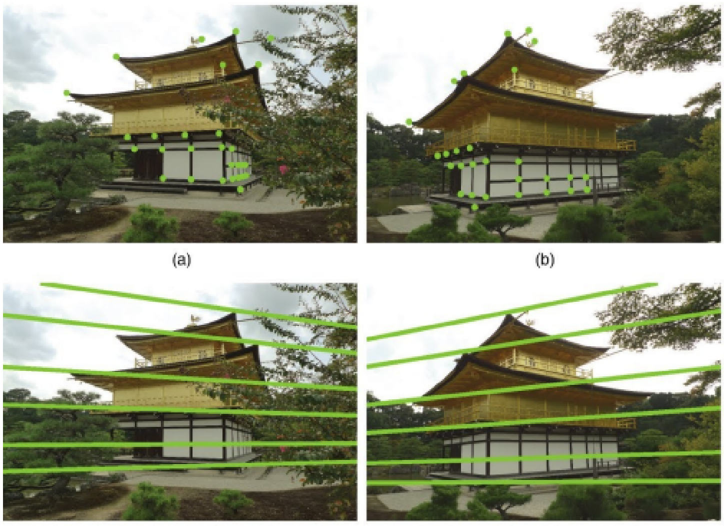
2.6 Essential Matrix Estimation

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



(a) (b) (c) (d)

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

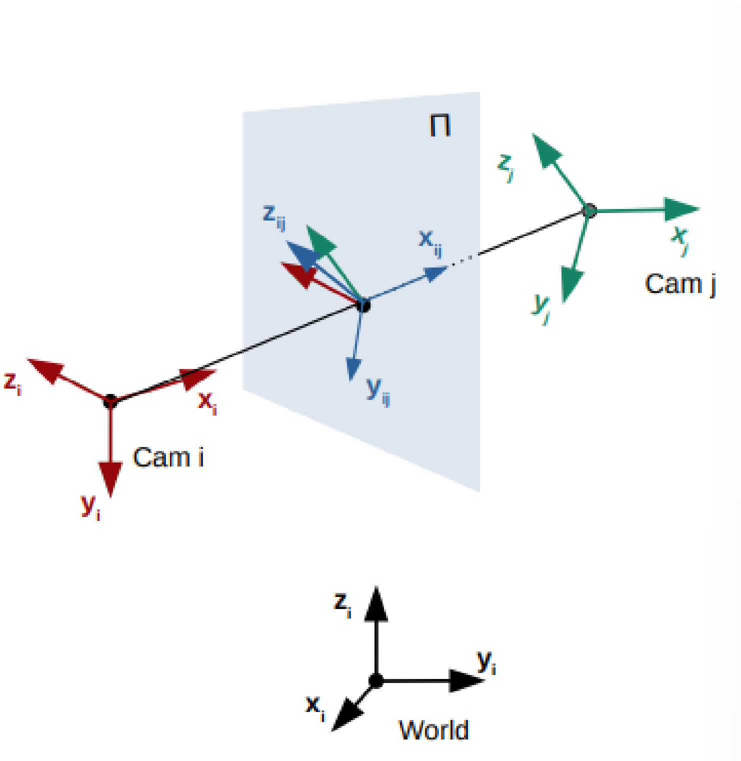
$$R_{ij} = (x_{ij} y_{ij} z_{ij})$$

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

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

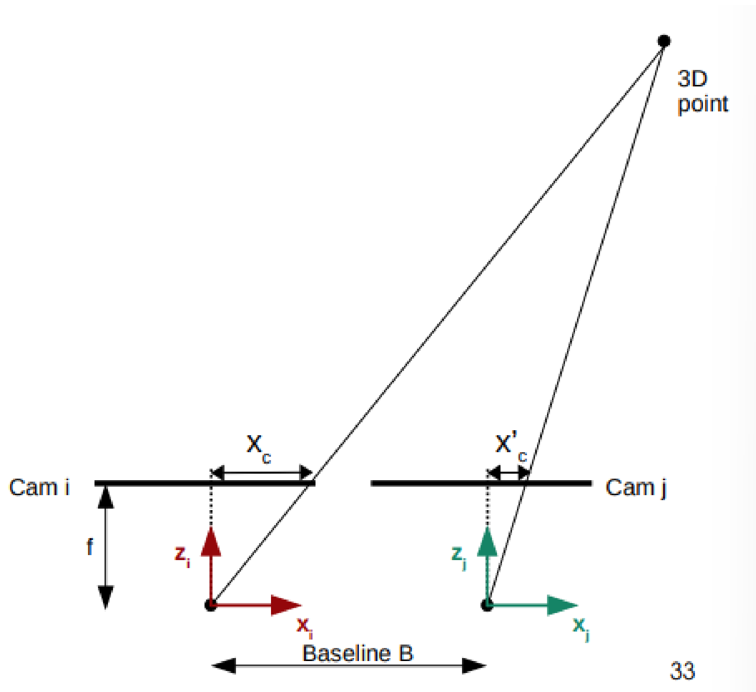
- Map to the original pixel are given by:

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



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

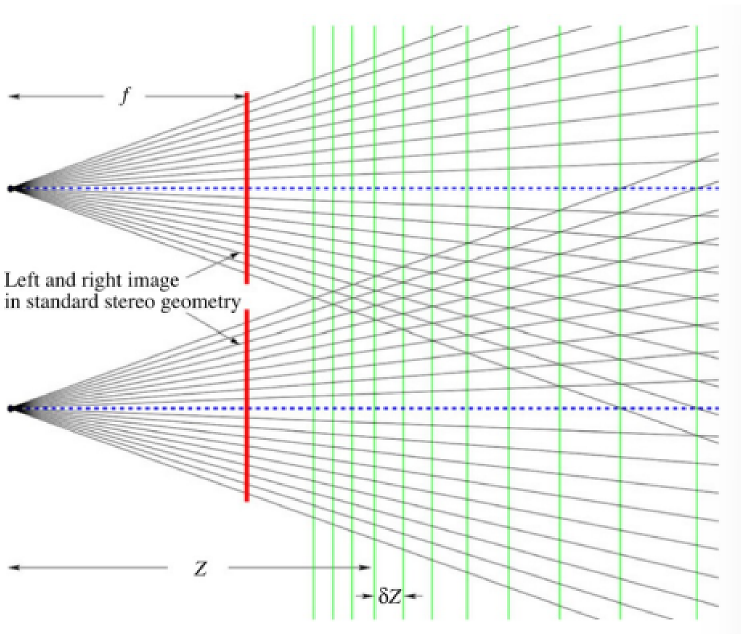
Search in a pair of stereo images for corresponding pixels that are projections of the same 3D point. Epipolar constraints reduces the search to one dimension. Epipolar constraints reduce the search to one dimension.
Sparse Matching: extract salient points and compute descriptors. Match according to 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:

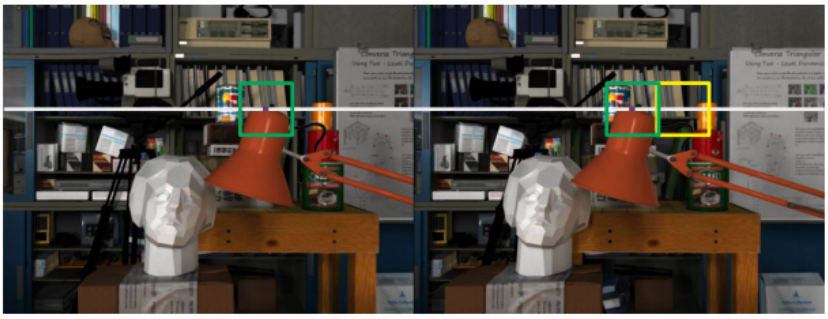
- Left base image: $L = \{+1, 0, -1, ..., -d_{max}\}$ $\quad 0 \leq -d \leq \min\{d_{max}, x - 1\}$
- Right base image: $L = \{-1, 0, +1, ..., d_{max}\}$ $\quad 0 \leq d \leq \min\{d_{max}, N_{cols} - x\}$

3.2 Depth uncertainty



A disparity of 0 between corresponding pixels $x_L - x_R = 0$ means 3D point at infinity. The distances δZ 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:

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

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

- 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.

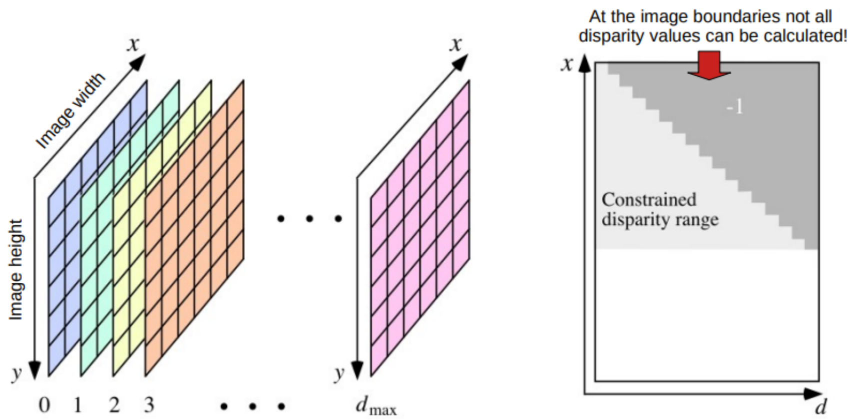


Figure 3.1: Cost volume

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:

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

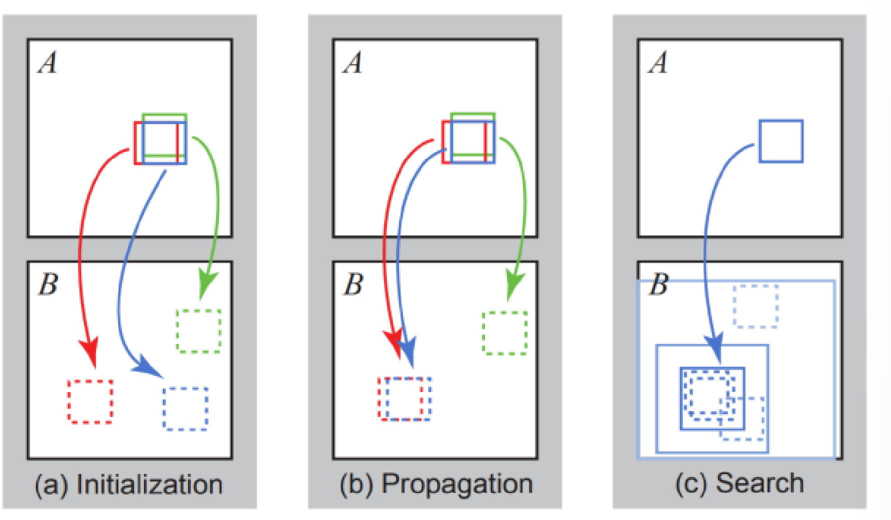
where

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

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.