

Introduction to 3D vision

Timothée Wintz

Sony CSL Paris

July 3, 2018

Camera model

Stereo vision

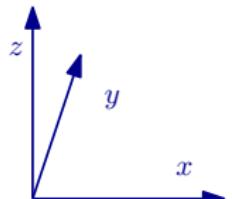
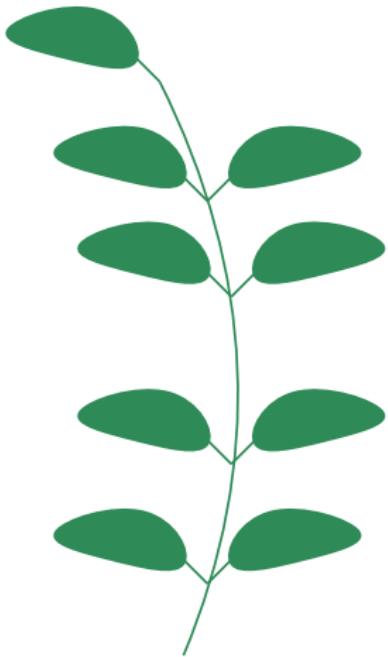
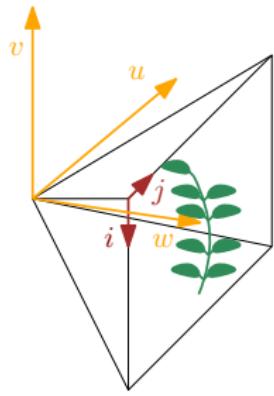
Photogrammetry

Camera model

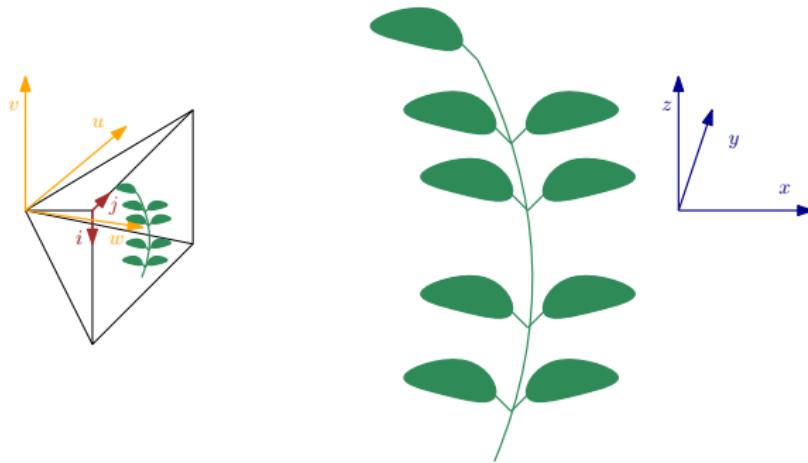
Stereo vision

Photogrammetry

Pinhole camera model



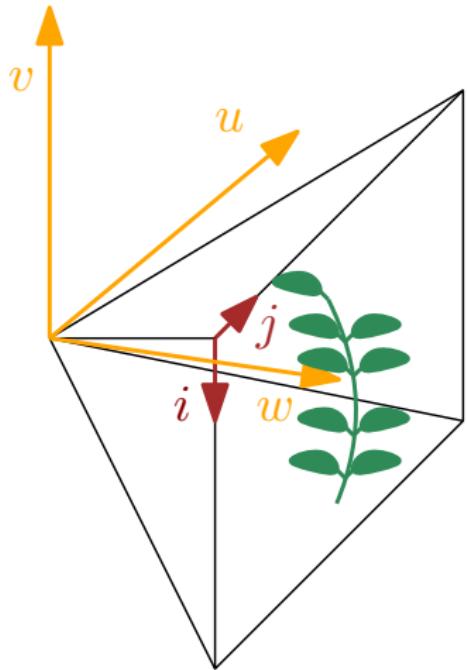
Camera pose



Extrinsics: rotation R and translation T .

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + T$$

Fundamental matrix



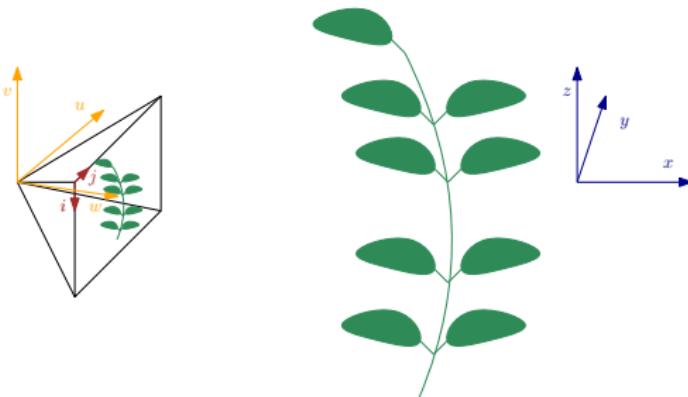
$$j = f_x \frac{u}{w} + c_x, i = f_y \frac{v}{w} + c_y$$

$$\begin{pmatrix} j \\ i \\ 1 \end{pmatrix} = F \begin{pmatrix} u/w \\ v/w \\ 1 \end{pmatrix}$$

where

$$F = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

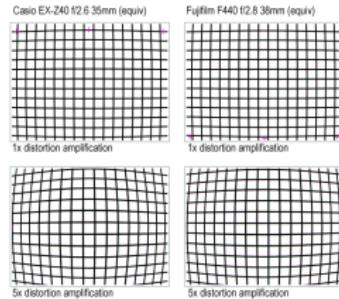
Pinhole camera model



Extrinsics:
$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + T$$

Intrinsics:
$$\begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Camera distortion



OpenCV model

$$r = \sqrt{x^2 + y^2}$$

$$x_{\text{corrected}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{\text{corrected}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Simplified radial: $k_2 = k_3 = 0.$

Summary

Camera parameters

5 to 7 parameters: $f_x, f_y, c_x, c_y, k_1, (k_2, k_3)$.

Pose parameters

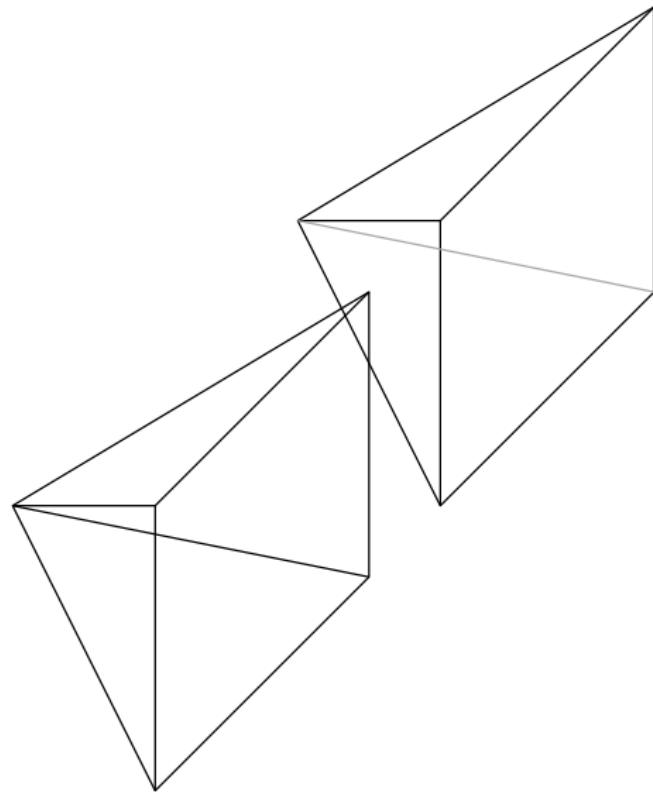
6 parameters: 3 for R , 3 for T .

Camera model

Stereo vision

Photogrammetry

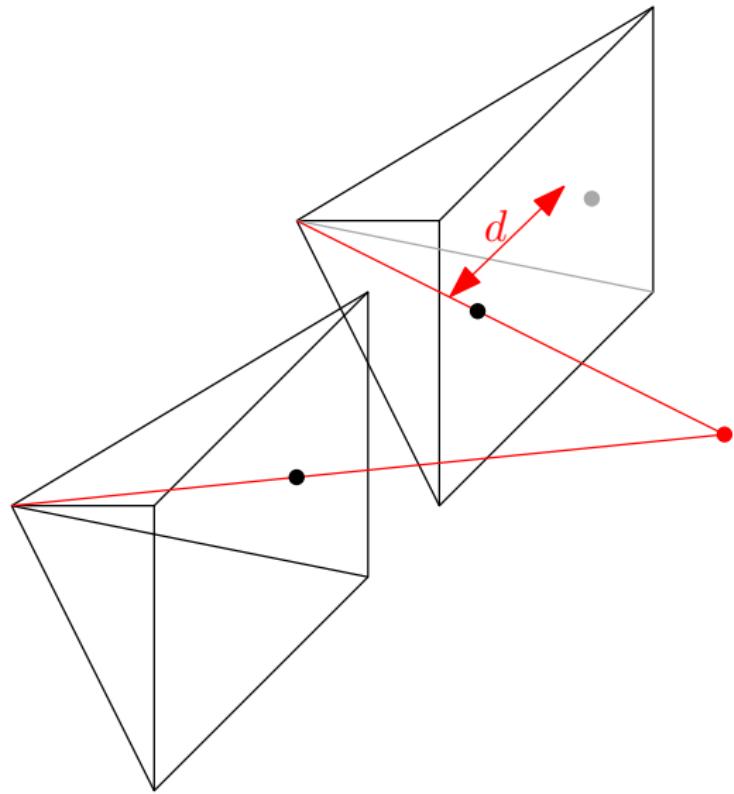
Basic principle



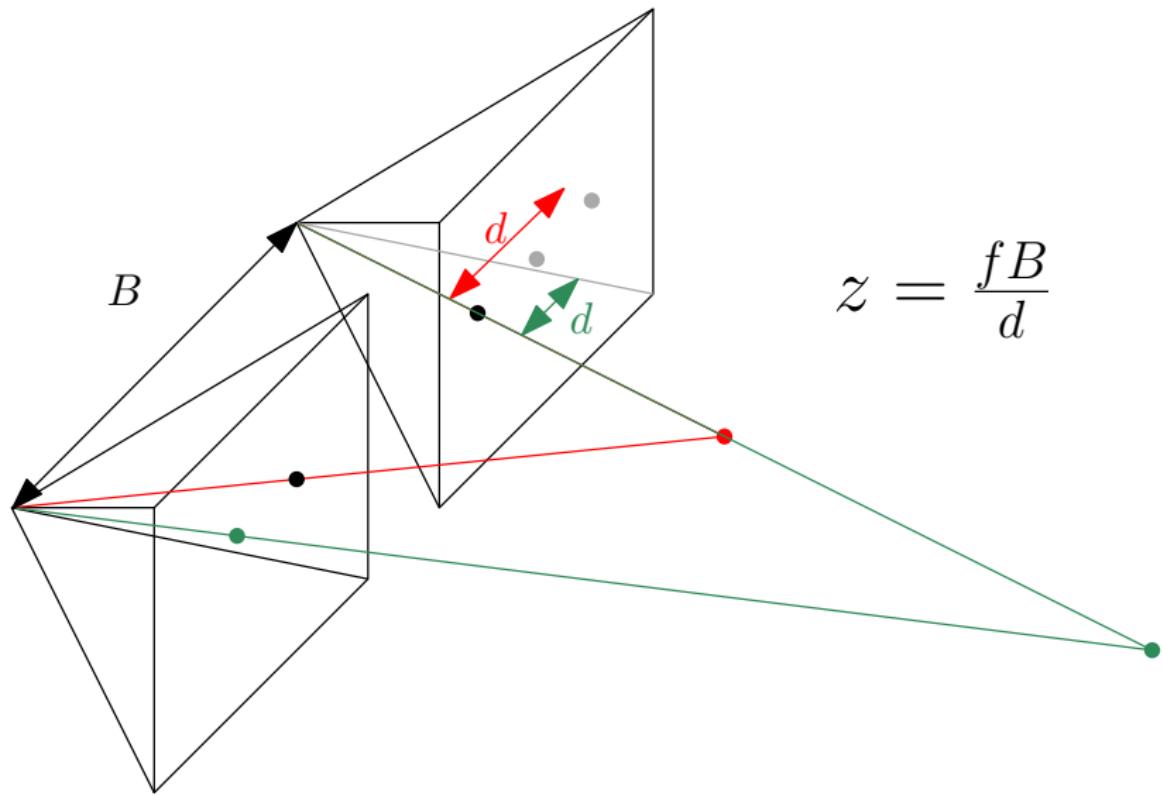
•

•

Basic principle



Basic principle



Disparity map

Goal

Compute depth at every point in the image.

First, we need to *rectify* images. I.e. align images such that points in 3D space are on the same line in both images.

Then for any pixel location i, j in the left image we have to find the corresponding point $i, j - d(i, j)$ in the right image. To do this, we minimize:

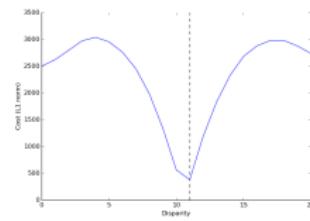
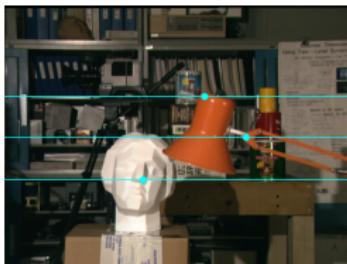
$$d(i, j) = \operatorname{argmin}_d(C(i, j, d)),$$

$$C(i, j, d) := \sum_{i', j' \in W(i, j)} |D_1(i', j') - D_2(i', j' - d)|.$$

Disparity map

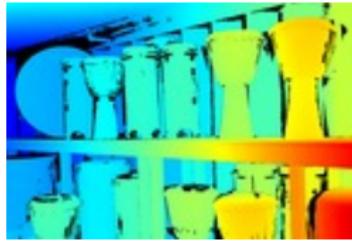
$$d(i,j) = \operatorname{argmin}_d(C(i,j,d)),$$

$$C(i,j,d) := \sum_{i'j' \in W(i,j)} |D_1(i',j') - D_2(i',j' - d)|.$$



<https://sourishghosh.com/2016/dense-disparity-maps-orb-descriptors/>

Disparity map



Camera model

Stereo vision

Photogrammetry

Principle

RGB Images



Point Cloud



Solutions

Free and open source: colmap, openMVG+openMVS, ...

Commercial: agisoft, ...

Pipeline

- ▶ Feature detection and computation
- ▶ Feature matching
- ▶ Pose estimation
- ▶ Bundle adjustment
- ▶ Dense point cloud generation

Feature detection and matching



Pose estimation

For every match of a real world point $\mathbf{x} = (x, y, z)$, between a camera pair k_1, k_2 . Let $x_1 = (i_1, j_1)$ be the pixel in view k_1 , and $x_2 = (i_2, j_2)$ be the corresponding pixel in view k_2 . We get the equations:

$$F(R_1\mathbf{x} + T_1) = \mathbf{x}_1$$

$$F(R_2\mathbf{x} + T_2) = \mathbf{x}_2$$

For every match, we add 4 equations and 3 unknowns. With enough matches, we can recover the relative pose of the two views up to a homotopy of the whole space.

Pose estimation

Iterating this over all pairs of images we obtain:

- ▶ The relative pose of all cameras.
- ▶ A sparse set of 3D points corresponding to the matches.

Bundle adjustment

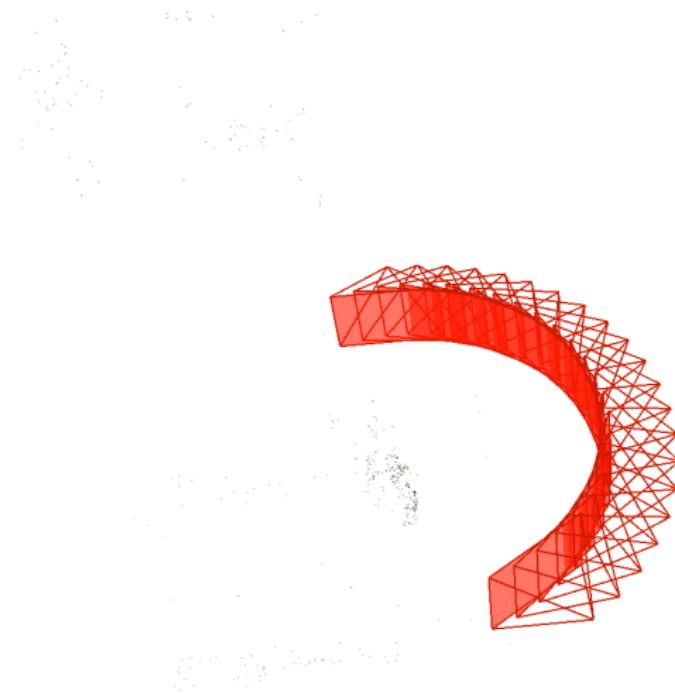
Refine the pose of all cameras and the position of all points simultaneously:

$$\min_{R_i, T_i, F, x_i} \sum_i \sum_j v_{i,j} d(F(R_i x_i + T_i), y_i)^2,$$

where:

- ▶ R_i, T_i, F : camera parameters,
- ▶ x_i : 3D points,
- ▶ d : euclidian distance.

Result (colmap)



Dense point cloud

We compute stereo disparity maps in each pair of images:

