Chapter 3. Uncalibrated stereo

Prof. Slimane LARABI, USTHB

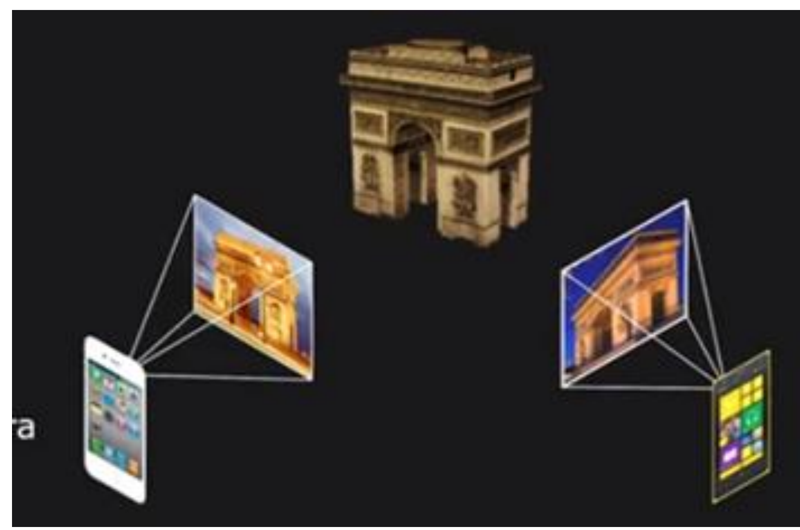
Chapitre 3. Uncalibrated stereo

Chapitre 3. Uncalibrated stereo

3.1 Overview

Two individuals taking photographs of a monument from different vantage points will produce two distinct images.

If we know the internal parameters of the two cameras, then from these two views we can compute the translation and rotation of one camera with respect to another camera. And then we can then compute a 3D model of the monument.

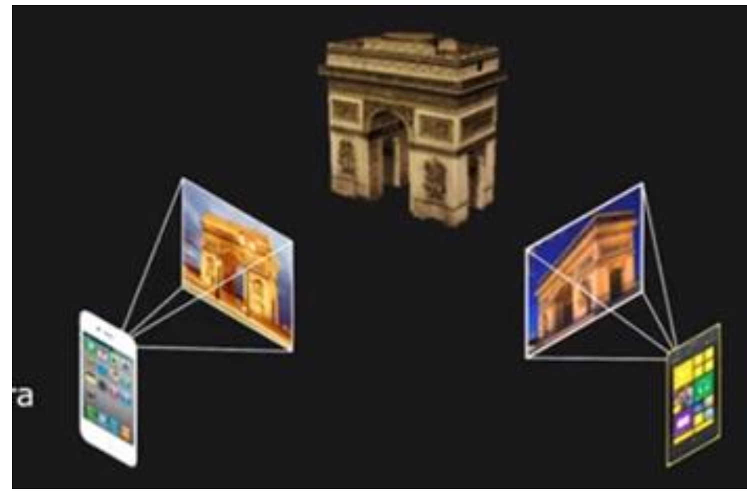


Chapitre 3. Uncalibrated stereo

3.1 Overview

We present a method to estimate 3D structure of a static scene from two arbitrary views. We will study :

- The problem of uncalibrated stereo
- The epipolar geometry
- Estimating Fundamental matrix
- Finding dense correspondences
- Computing depth



Note that:
-intrinsic parameters of cameras are known.
- extrinsic parameters (relative position, orientation of cameras) are unknown.

Chapitre 3. Uncalibrated stereo

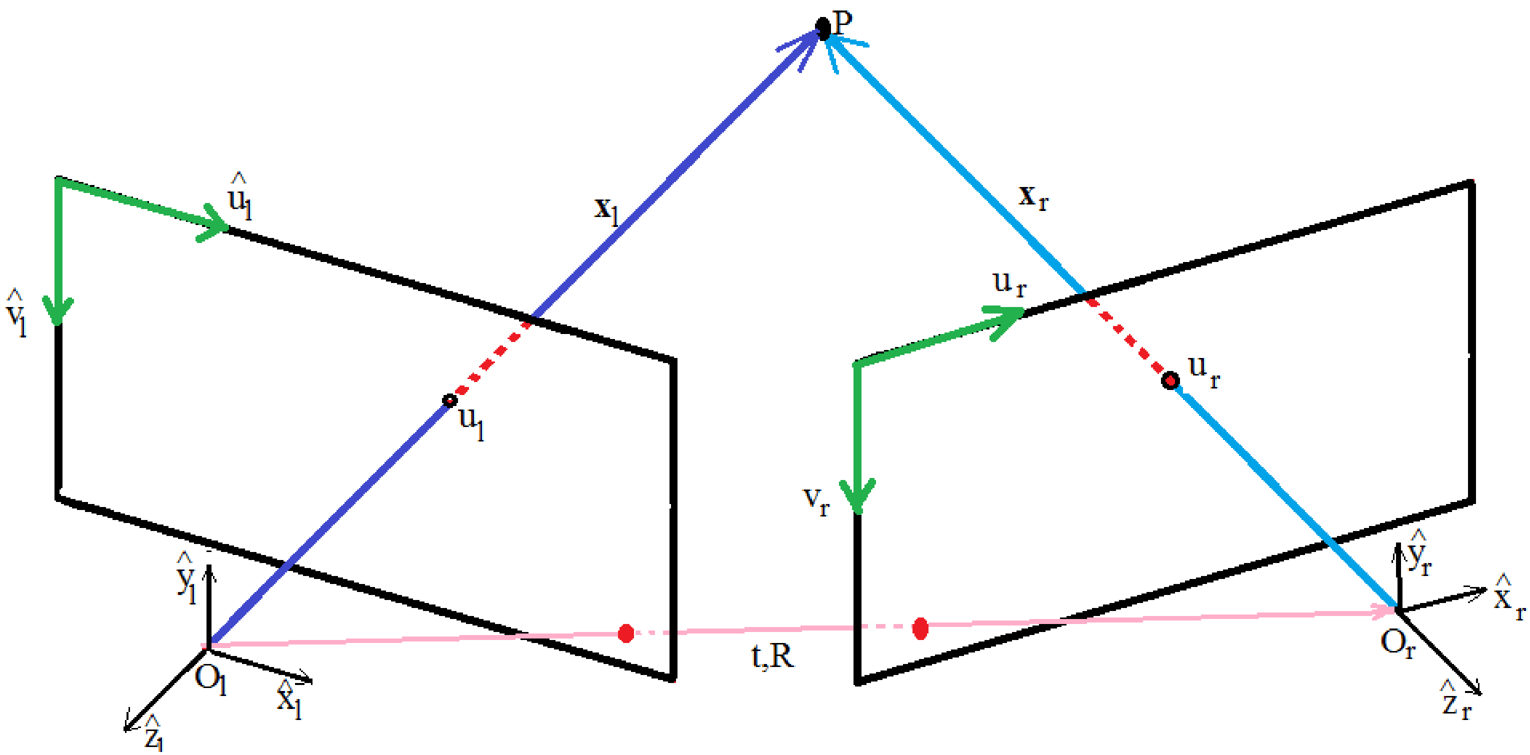
3.2 Problem of uncalibrated stereo

What is known:

- The camera matrix K of each camera is available.

What we need:

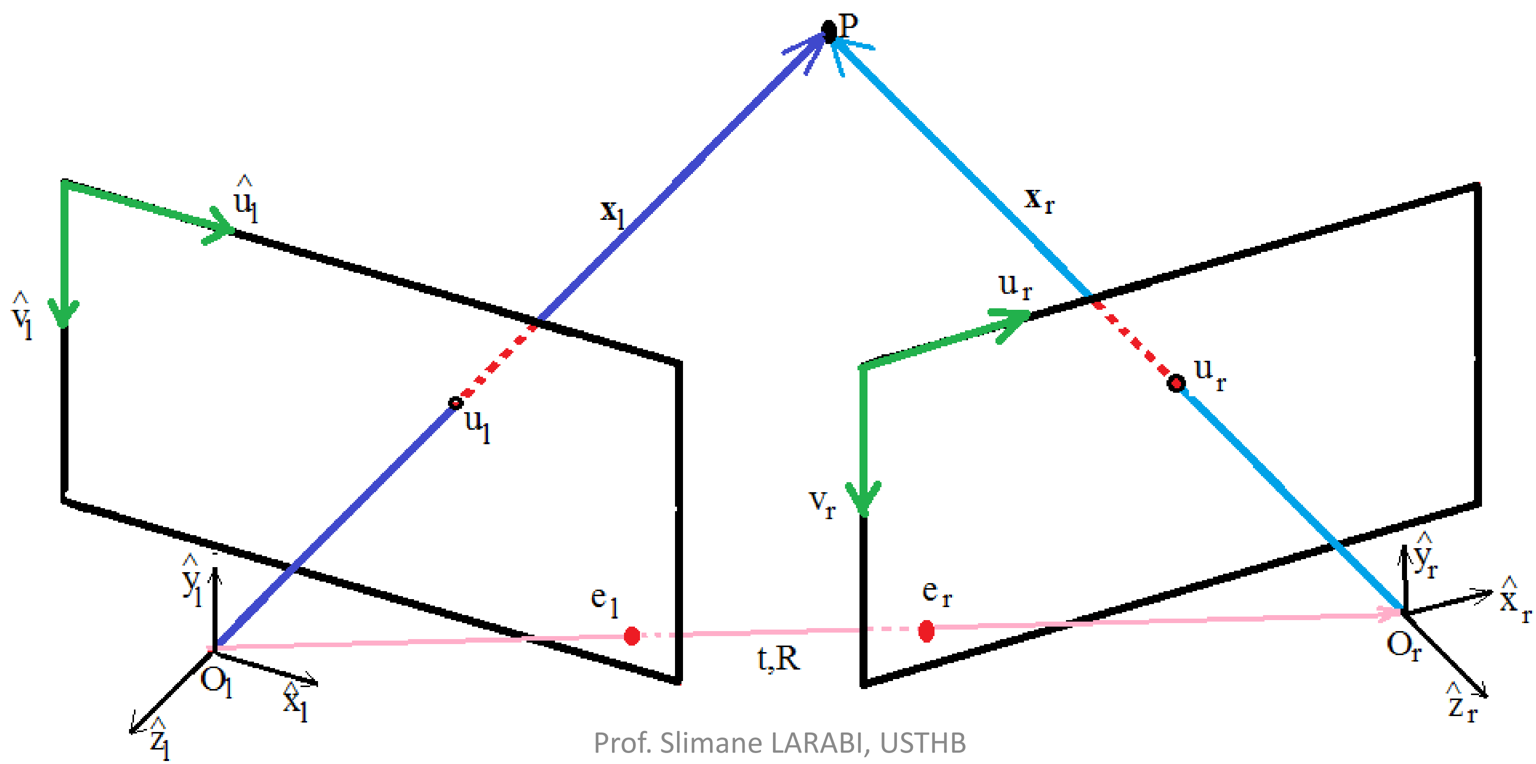
- Find reliable corresponding points on the two images



3.3 Epipolar Geometry

The epipoles

Is defined as the image point of pinhole of one camera as viewed by the other camera. In the figure e_l and e_r are the two **epipoles**, they are unique for a given stereo pair.



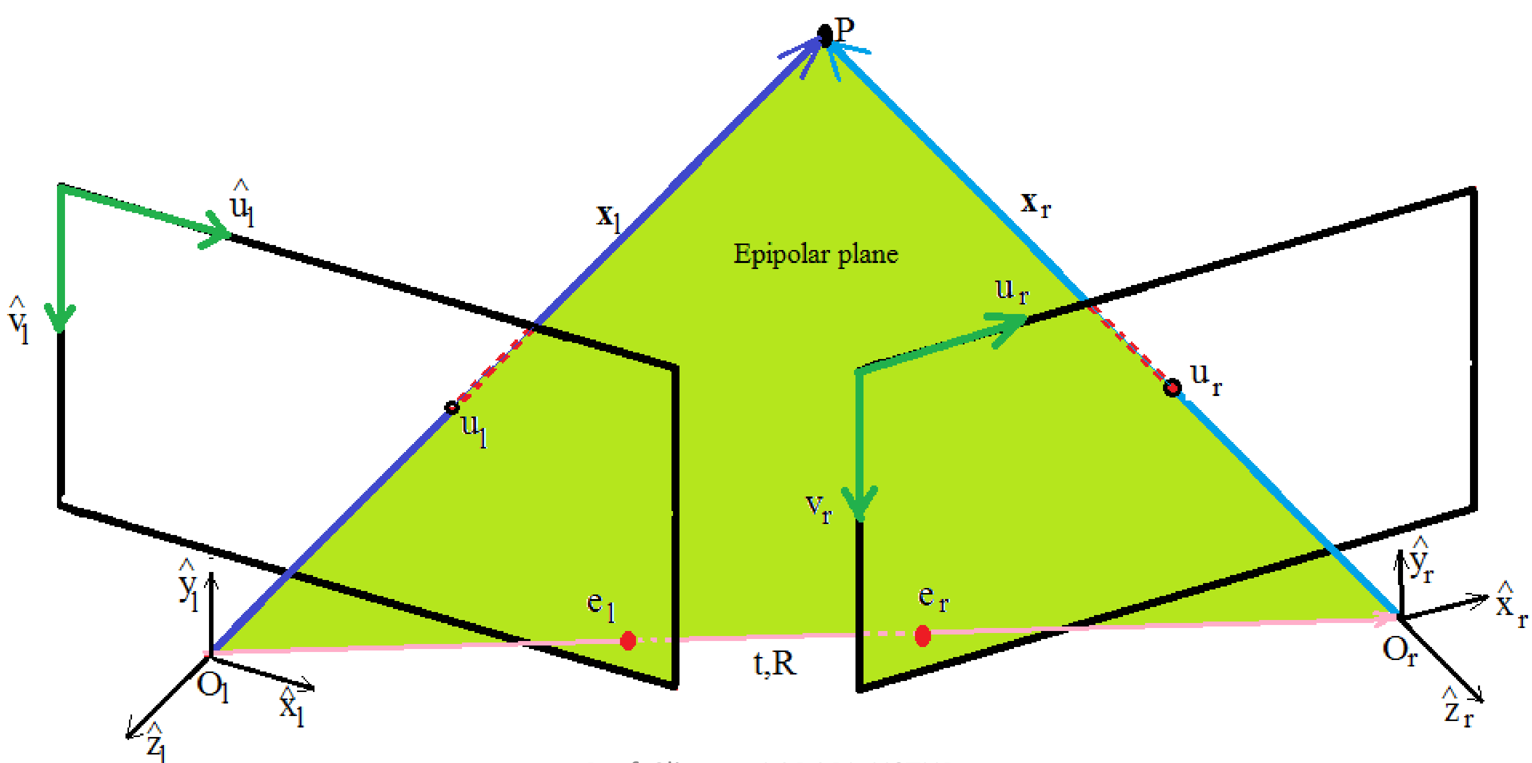
Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar plane

Is associated to a scene point P: is formed by camera origins O_l and O_r , epipoles e_l and e_r and scene point P.

Every scene point P lies on a unique epipolar plane.



Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

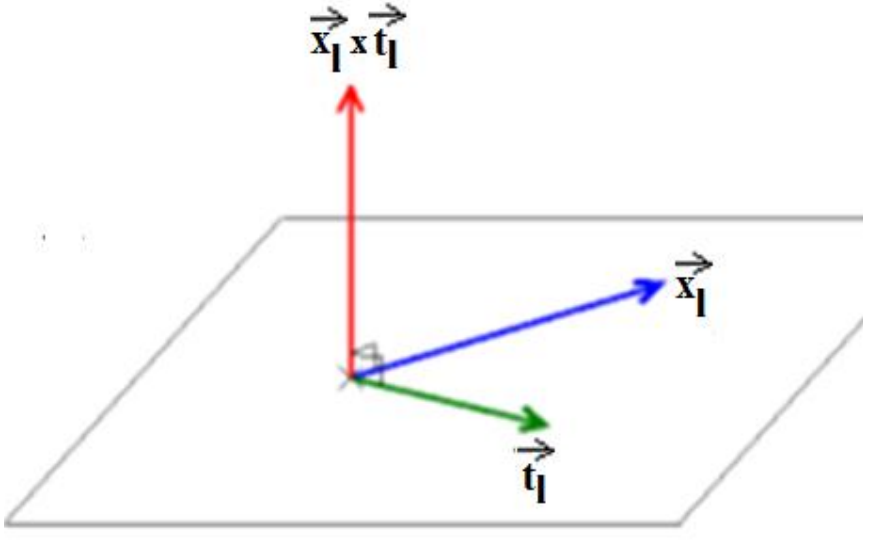
Epipolar constraint

Let n bet the vector normal to the epipolar plane.

$$n = t \times x_l$$

$$x_l . n = x_l . (t \times x_l) = 0 \quad : \; epipolar\ constraint$$

$$\vec{x}_l \times \vec{t} = \|\vec{x}_l\| . \|\vec{t}\| . \sin(\vec{x}_l, \vec{t})$$



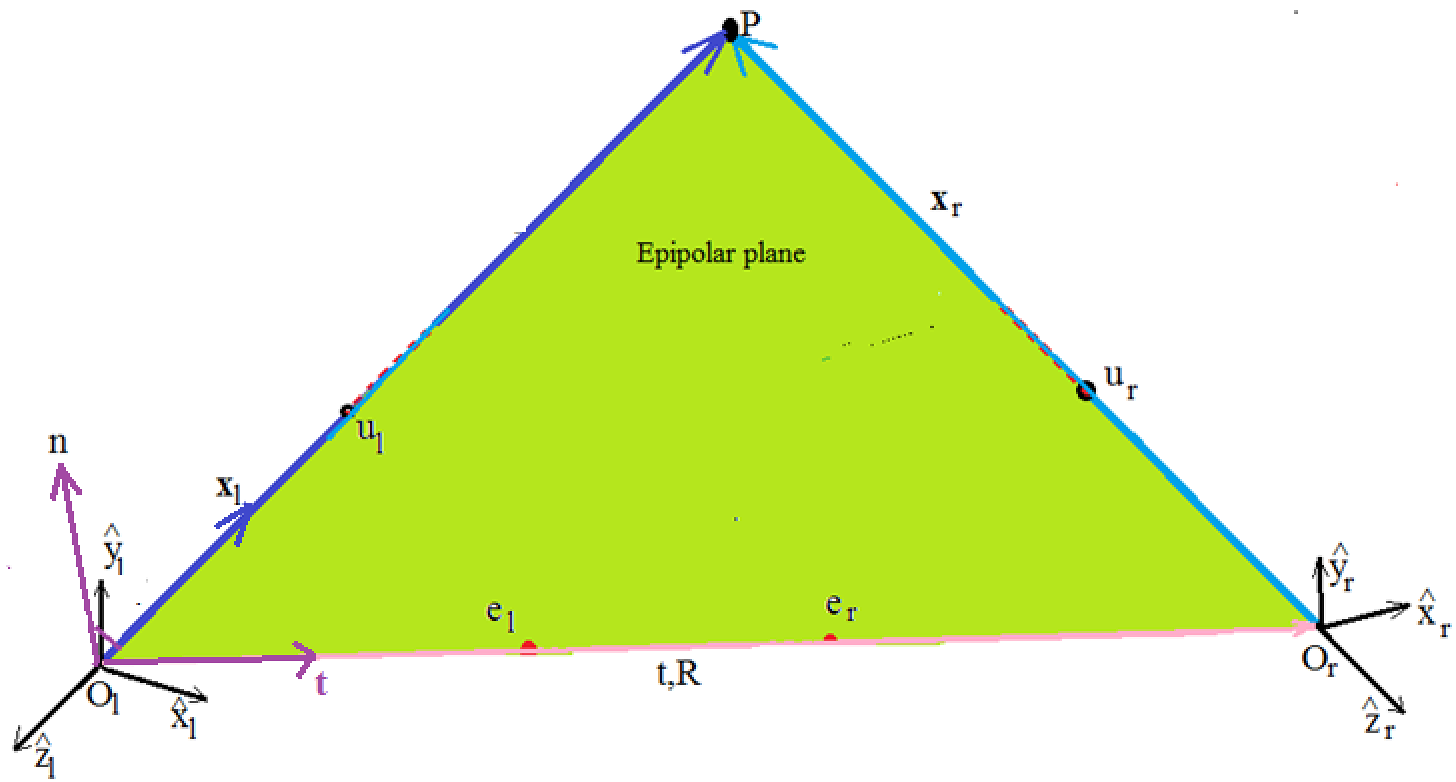
Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar constraint

Let n bet the vector normal to the epipolar plane.

$$n = t \times x_l \qquad x_l.n = x_l.(t \times x_l) = 0 \ (dot\ product) \ : \ epipolar\ constraint$$



Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar constraint

Writing the epipolar constraint in matrix form:

$$\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \times \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} t_y z_l - t_z y_l \\ t_z x_l - t_x z_l \\ t_x y_l - t_y x_l \end{bmatrix}$$

$$x_l.(t \times x_l) = 0$$

$$[x_l \; y_l \; z_l] \begin{bmatrix} t_y z_l - t_z y_l \\ t_z x_l - t_x z_l \\ t_x y_l - t_y x_l \end{bmatrix} = 0 \qquad [x_l \; y_l \; z_l] \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = 0 \qquad [x_l \; y_l \; z_l] \, T_x \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = 0$$

We note $t_{3\times1}$: $\left[t_x \; t_y \; t_z\right]$ is the position of the right camera in the left camera frame.

Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar constraint

$$[x_l\ y_l\ z_l]\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}=0 \qquad [x_l\ y_l\ z_l]\ T_x \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}=0$$

We note $t_{3\times 1}$: $\left[t_x\ t_y\ t_z\right]$ is the position of the right camera in the left camera frame.
We note $R_{3\times 3}$: the orientation of the left camera in the right camera's frame

$$x_l = R\ x_r + t \qquad \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}+\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar constraint

$$[x_l \; y_l \; z_l]\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}=0 \qquad \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}+\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$[x_l \; y_l \; z_l]\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}\left(\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}+\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}\right)=0$$

$$[x_l \; y_l \; z_l]\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}+\underbrace{[x_l \; y_l \; z_l]\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}}_{=0}=0$$

Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar constraint

$$[x_l\ y_l\ z_l]\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}=0$$

$$[x_l\ y_l\ z_l]T_xR\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}=0 \qquad [\mathbf{x_l}\ \mathbf{y_l}\ \mathbf{z_l}]\begin{bmatrix} \mathbf{e_{11}} & \mathbf{e_{12}} & \mathbf{e_{13}} \\ \mathbf{e_{21}} & \mathbf{e_{22}} & \mathbf{e_{23}} \\ \mathbf{e_{31}} & \mathbf{e_{32}} & \mathbf{e_{33}} \end{bmatrix}\begin{bmatrix} \mathbf{x_r} \\ \mathbf{y_r} \\ \mathbf{z_r} \end{bmatrix}=\mathbf{0}$$

$E = T_xR$ *is the* ***essential matrix***

This is the equation relating the 3D coordinates of P with respect to right and left coordinates frames

Longuet-Higgins 1981

Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar constraint

Essential matrix E: Decomposition

It is possible to decouple R and T_x from $E\ using\ SVD$

How to find E?

$$[x_l\ y_l\ z_l]\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}=0$$

$$x_l^T E x_r = 0$$

$\boldsymbol{x}_l$and $\boldsymbol{x}_r$ are unknown, but we know corresponding points in image coordinates

3.3 Epipolar Geometry

Epipolar constraint
How to find E?

The perspective projection of each camera:

$$u = f_x \frac{x_c}{z_c} + O_x \qquad v = f_y \frac{y_c}{z_c} + O_y$$

$$z_l u_l = f_x^l x_l + z_l O_x^l \qquad z_l v_l = f_y^l y_l + z_l O_y^l$$

$$z_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} z_l u_l \\ z_l v_l \\ z_l \end{bmatrix} = \begin{bmatrix} f_x^l x_l + z_l O_x^l \\ f_y^l y_l + z_l O_y^l \\ z_l \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^l & 0 & O_x^l \\ 0 & f_y^l & O_y^l \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Known: Calibration matrix}} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}$$

Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar constraint
How to find E?

$$z_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^l & 0 & O_x^l \\ 0 & f_y^l & O_y^l \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}$$

K_l: $Calibration\ matrix$ of left camera

$$z_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^r & 0 & O_x^r \\ 0 & f_y^r & O_y^r \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}$$

K_r: $Calibration\ matrix$ of right camera

We obtain:

$$x_l^T = [u_l \ \ v_l \ \ 1] z_l K_l^{-1\ T}$$

$$x_r = K_r^{-1} z_r \begin{bmatrix} u_l \\ v_r \\ 1 \end{bmatrix}$$

Chapitre 3. Uncalibrated stereo

3.3 Epipolar Geometry

Epipolar constraint
How to find E?

Epipolar constraint:

$$[\mathbf{x_l}\ \mathbf{y_l}\ \mathbf{z_l}]\begin{bmatrix} \mathbf{e_{11}} & \mathbf{e_{12}} & \mathbf{e_{13}} \\ \mathbf{e_{21}} & \mathbf{e_{22}} & \mathbf{e_{23}} \\ \mathbf{e_{31}} & \mathbf{e_{32}} & \mathbf{e_{33}} \end{bmatrix}\begin{bmatrix} \mathbf{x_r} \\ \mathbf{y_r} \\ \mathbf{z_r} \end{bmatrix}=0$$

Rewriting in terms of image coordinates:

$$x_l^T = [u_l \;\; v_l \;\; 1] z_l K_l^{-1\;T} \qquad x_r = K_r^{-1} z_r \begin{bmatrix} u_l \\ v_r \\ 1 \end{bmatrix}$$

$$[u_l \;\; v_l \;\; 1] z_l K_l^{-1\;T} \begin{bmatrix} \mathbf{e_{11}} & \mathbf{e_{12}} & \mathbf{e_{13}} \\ \mathbf{e_{21}} & \mathbf{e_{22}} & \mathbf{e_{23}} \\ \mathbf{e_{31}} & \mathbf{e_{32}} & \mathbf{e_{33}} \end{bmatrix} K_r^{-1} z_r \begin{bmatrix} u_l \\ v_r \\ 1 \end{bmatrix}=0$$

As z_l and z_r are non zero:

3.3 Epipolar Geometry

Epipolar constraint

How to find E?

Faugeras, Luong 1992,

$$[u_l \;\; v_l \;\; 1] K_l^{-1\,T} \begin{bmatrix} \mathbf{e}_{11} & \mathbf{e}_{12} & \mathbf{e}_{13} \\ \mathbf{e}_{21} & \mathbf{e}_{22} & \mathbf{e}_{23} \\ \mathbf{e}_{31} & \mathbf{e}_{32} & \mathbf{e}_{33} \end{bmatrix} K_r^{-1} \begin{bmatrix} u_l \\ v_r \\ 1 \end{bmatrix} = \mathbf{0}$$

$$[u_l \;\; v_l \;\; 1] \begin{bmatrix} \mathbf{f}_{11} & \mathbf{f}_{12} & \mathbf{f}_{13} \\ \mathbf{f}_{21} & \mathbf{f}_{22} & \mathbf{f}_{23} \\ \mathbf{f}_{31} & \mathbf{f}_{32} & \mathbf{f}_{33} \end{bmatrix} \begin{bmatrix} u_l \\ v_r \\ 1 \end{bmatrix} = x_l^T F x_r = 0$$

F= **Fundamental matrix**

F=$K_l^{-1\,T} E K_r^{-1}$

$\boldsymbol{K_l^T}$**F**$\boldsymbol{K_r}$=$\boldsymbol{K_l^T} K_l^{-1\,T} E K_r^{-1} \boldsymbol{K_r}$

E: **Essential matrix, E=**$\boldsymbol{K_l^T F K_r}$

Chapitre 3. Uncalibrated stereo

3.4 Estimating Fundamental Matrix

Initial correspondence

Find a set of corresponding features (at least 8) in left and right images (using SIFT or hand-picked)

$(u_1^l, v_1^l), (u_2^l, v_2^l), \dots (u_m^l, v_m^l)$ $(u_1^r, v_1^r), (u_2^r, v_2^r), \dots (u_m^r, v_m^r)$

Chapitre 3. Uncalibrated stereo

3.4 Estimating Fundamental Matrix

For each stereo correspondence (i), write out the epipolar constraint:

$$\begin{bmatrix} u_l^i & v_l^i & 1 \end{bmatrix} \begin{bmatrix} \mathbf{f_{11}} & \mathbf{f_{12}} & \mathbf{f_{13}} \\ \mathbf{f_{21}} & \mathbf{f_{22}} & \mathbf{f_{23}} \\ \mathbf{f_{31}} & \mathbf{f_{32}} & \mathbf{f_{33}} \end{bmatrix} \begin{bmatrix} u_r^i \\ v_r^i \\ 1 \end{bmatrix} = 0$$

We obtain for all stereo correspondences the linear system:

Chapitre 3. Uncalibrated stereo

3.4 Estimating Fundamental Matrix

$$\begin{bmatrix} u_l^1 u_r^1 & u_l^1 v_r^1 & u_l^1 & v_l^1 u_r^1 & v_l^1 v_r^1 & v_l^1 & u_r^1 & v_r^1 & 1 \\ - & - & - & - & - & - & - & - & - \\ u_l^i u_r^i & u_l^i v_r^i & u_l^i & v_l^i u_r^i & v_l^i v_r^i & v_l^i & u_r^i & v_r^i & 1 \\ - & - & - & - & - & - & - & - & - \\ u_l^m u_r^m & u_l^m v_r^m & u_l^m & v_l^m u_r^m & v_l^m v_r^m & v_l^m & u_r^m & v_r^m & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$Af = 0$$

3.4 Estimating Fundamental Matrix

The fundamental matrix F and kF describe the same epipolar geometry.
F is then defined only up to scale.

$$[u_l^i \;\; v_l^i \;\; 1]\begin{bmatrix} \boldsymbol{k}\mathbf{f}_{11} & \mathbf{kf}_{12} & \mathbf{kf}_{13} \\ \mathbf{kf}_{21} & \mathbf{kf}_{22} & \mathbf{kf}_{23} \\ \mathbf{kf}_{31} & \mathbf{kf}_{32} & \mathbf{kf}_{33} \end{bmatrix}\begin{bmatrix} u_r^i \\ v_r^i \\ 1 \end{bmatrix} = 0$$

We set the matrix F to some arbitrary scale: $\|f\|^2 = 1$

We want $\boldsymbol{Af}$ close to zero as possible and : $\|f\|^2 = 1$.

$\min_{f} \|Af\|^2$ such that $\|f\|^2 = 1$

This is the same problem like solving Projection matrix during camera calibration, or Homography matrix for image stitching.

Chapitre 3. Uncalibrated stereo

3.4 Estimating Fundamental Matrix

The next step is the computation of the **Essential** matrix E:

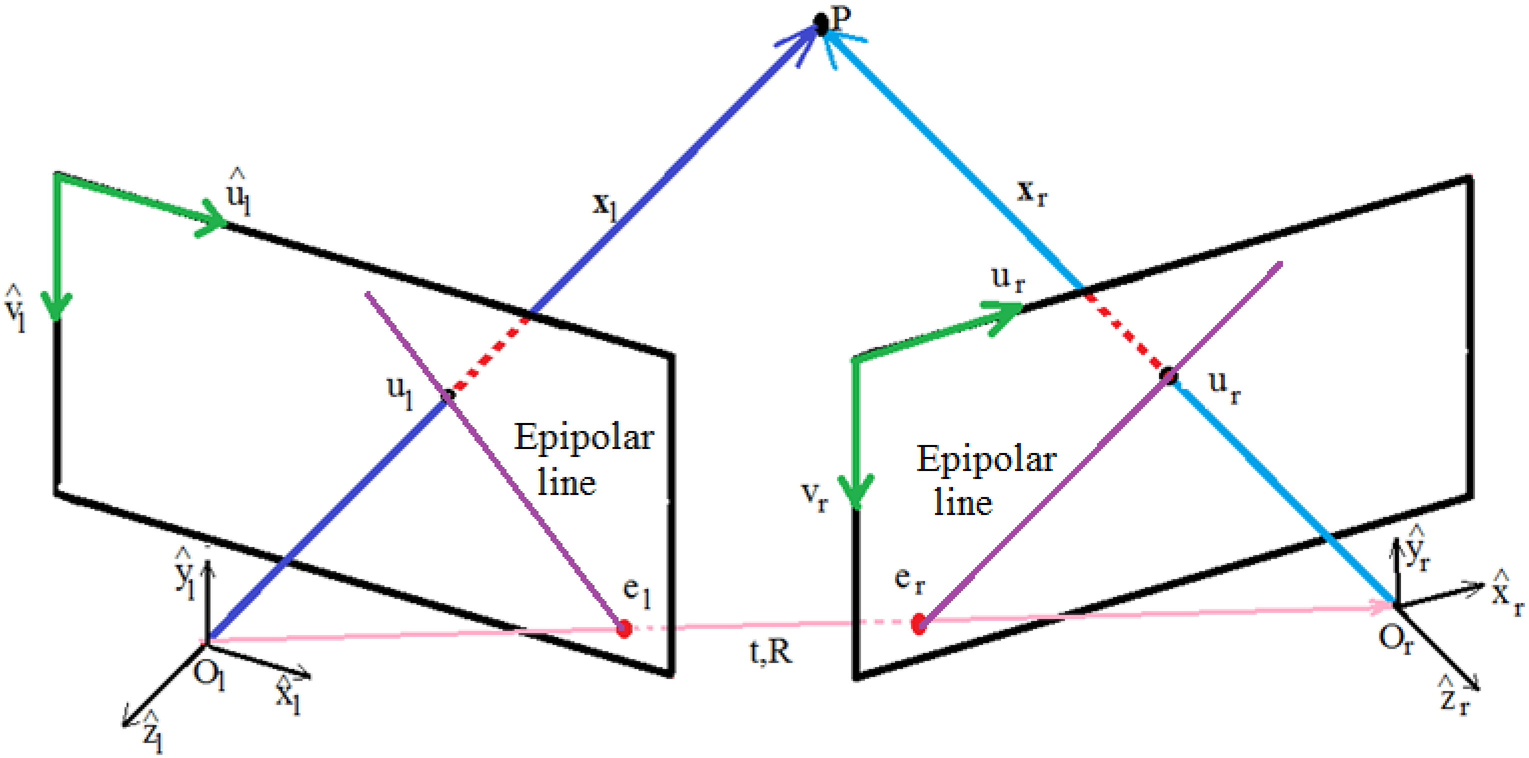
$$E = K_l^T F K_r$$

And extract the matrix $\boldsymbol{R}$ and vector $\boldsymbol{t}$ from $\boldsymbol{E\ using\ SVD}$ ($\boldsymbol{Singular\ Value\ Decomposition}$

$$E = R \times t$$

Chapitre 3. Uncalibrated stereo

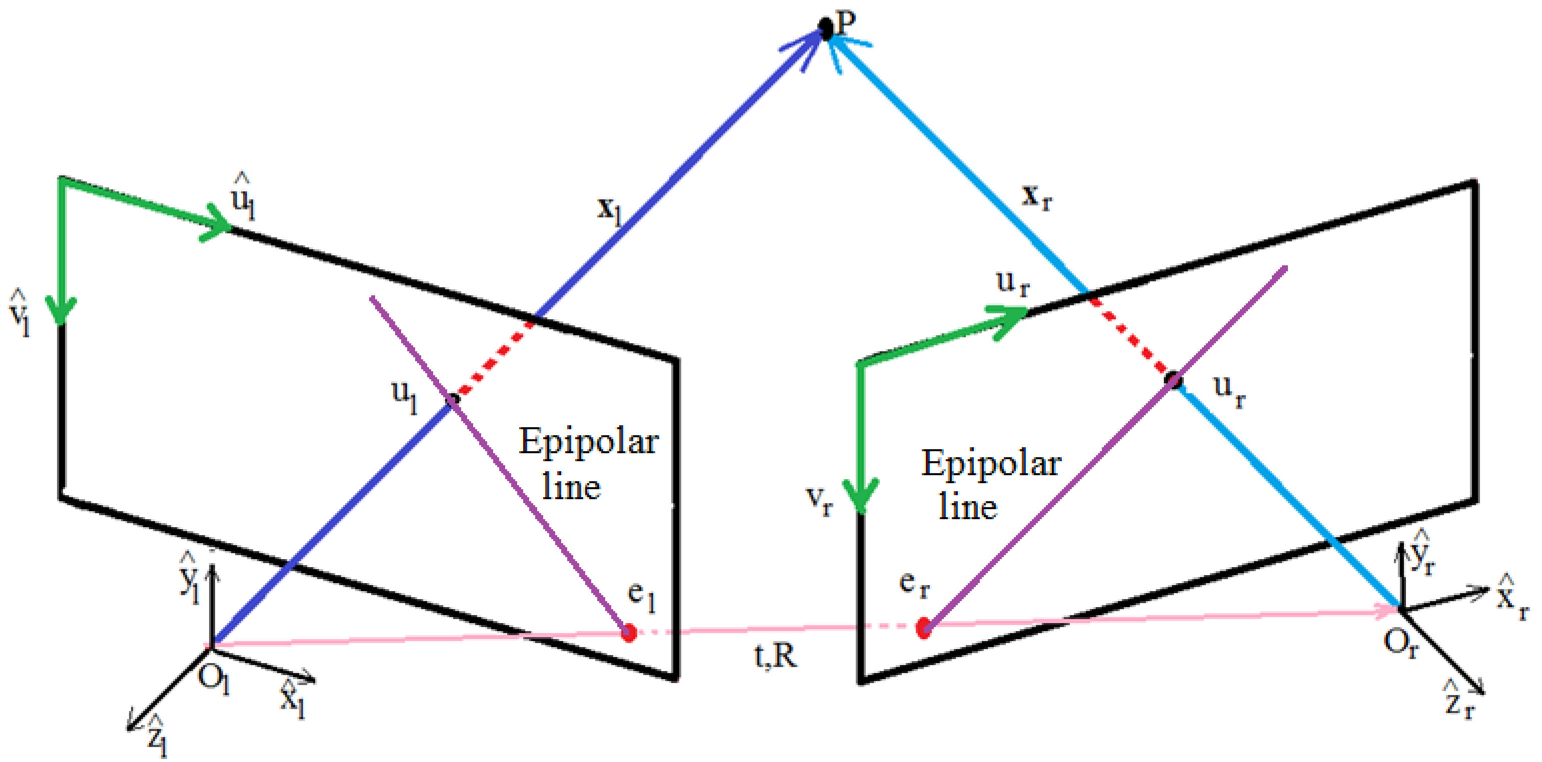
3.5 Finding Correspondances



Epipolar line is the intersection of image plane and epipolar plane.
At each scene point, there are two corresponding epipolar line,
one each on the two image planes.

Chapitre 3. Uncalibrated stereo

3.5 Finding Correspondances



Given one point on the left image, the corresponding point on the right image must lie on the epipolar line.
Finding correspondences is then reduced to 1D search.

3.5 Finding Correspondances

Finding Epipolar lines:
Given the Fundamental matrix and point on left image ()
Find the epipolar line in the right image.

$$[u_l \;\; v_l \;\; 1]\begin{bmatrix} \mathbf{f}_{11} & \mathbf{f}_{12} & \mathbf{f}_{13} \\ \mathbf{f}_{21} & \mathbf{f}_{22} & \mathbf{f}_{23} \\ \mathbf{f}_{31} & \mathbf{f}_{32} & \mathbf{f}_{33} \end{bmatrix}\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

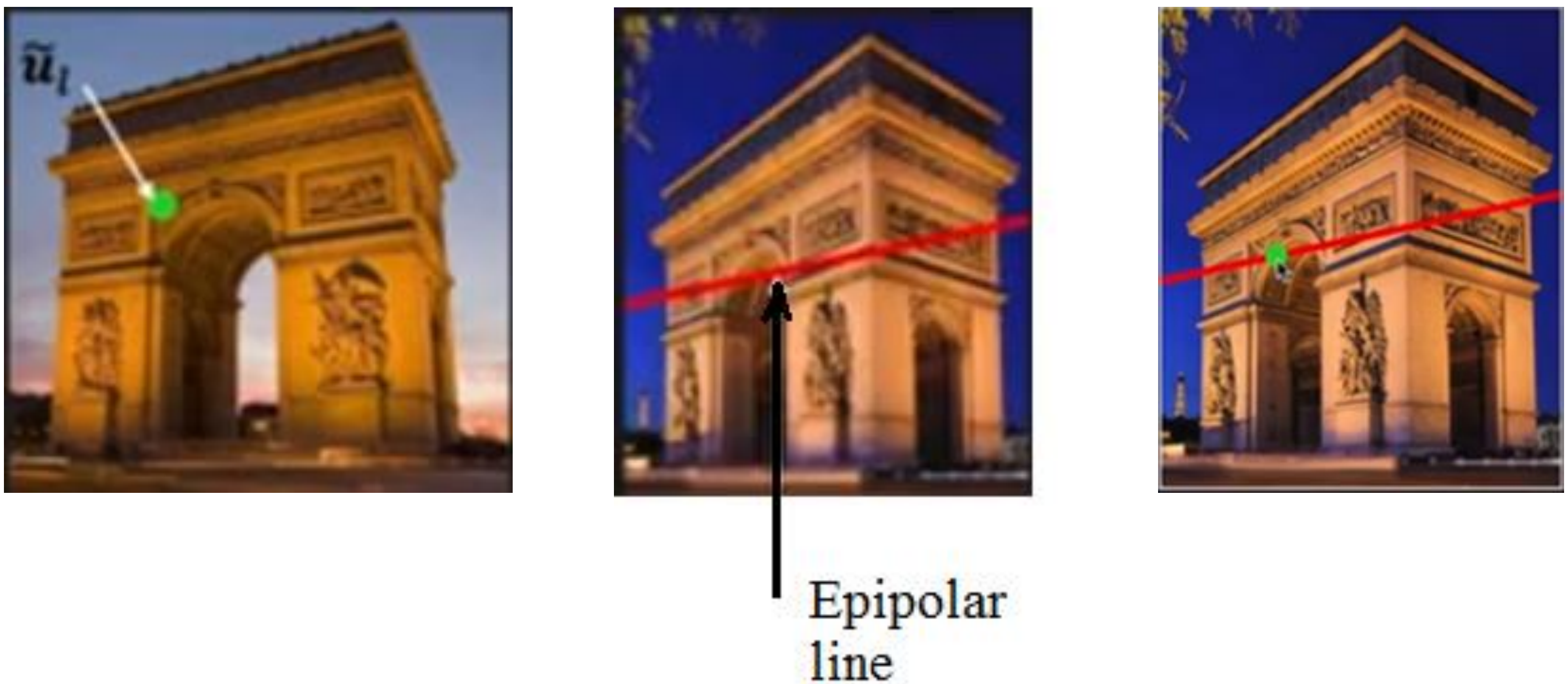
Expanding this equation:

$$u_r[u_l f_{11} + v_l f_{12} + f_{13}] + v_r[u_l f_{21} + v_l f_{22} + f_{23}] + [u_l f_{31} + v_l f_{32} + f_{33}] = 0$$

$$a_l u_r + b_l v_r + c_l = 0$$

Chapitre 3. Uncalibrated stereo

3.5 Finding Correspondances



Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Given the intrinsic parameters, the projections of scene point are:

$$\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^l & 0 & O_x^l & 0 \\ 0 & f_y^l & O_y^l & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix} \qquad \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^r & 0 & O_x^r & 0 \\ 0 & f_y^r & O_y^r & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

Given also the relative position and orientation between the two cameras:

$$\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

3.6 Computing Depth

We obtain then for the left camera:

$$\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^l & 0 & O_x^l & 0 \\ 0 & f_y^l & O_y^l & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} \qquad \tilde{u}_l = P_l \widetilde{\boldsymbol{x}}_r$$

For the right camera:

$$\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^r & 0 & O_x^r & 0 \\ 0 & f_y^r & O_y^r & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} \qquad \tilde{u}_r = M_{intr} \widetilde{\boldsymbol{x}}_r$$

3.6 Computing Depth

We obtain then for the left camera:

$$\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} \qquad \tilde{u}_l = P_l \widetilde{\boldsymbol{x}}_r$$

For the right camera:

$$\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} \qquad \tilde{u}_r = M_{intr} \widetilde{\boldsymbol{x}}_r$$

3.6 Computing Depth

Applying the cross product for the left camera:

$$\tilde{u}_l \times \tilde{u}_l = 0 = \tilde{u}_l \times P_l \widetilde{\boldsymbol{x}}_r$$

$$\begin{bmatrix} \boldsymbol{u_l} \\ \boldsymbol{v_l} \\ \boldsymbol{1} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{P_{l1}\widetilde{x}_r} \\ \boldsymbol{P_{l2}\widetilde{x}_r} \\ \boldsymbol{P_{l3}\widetilde{x}_r} \\ P_{l4}\widetilde{\boldsymbol{x}}_r \end{bmatrix} = 0$$

$$v_l P_{l3} \widetilde{\boldsymbol{x}}_r - P_{l2} \widetilde{\boldsymbol{x}}_r = 0$$

$$u_l P_{l3} \widetilde{\boldsymbol{x}}_r - P_{l1} \widetilde{\boldsymbol{x}}_r = 0$$

$$u_l P_{l2} \widetilde{\boldsymbol{x}}_r - P_{l2} \widetilde{\boldsymbol{x}}_r = 0$$

$$\begin{bmatrix} v_l P_{l3} - P_{l2} \\ u_l P_{l3} - P_{l1} \\ u_l P_{l2} - P_{l2} \end{bmatrix} \widetilde{\boldsymbol{x}}_r = 0$$

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Applying the cross product for the right camera:

$$\tilde{u}_r \times \tilde{u}_r = 0 = \tilde{u}_r \times M_r \widetilde{\boldsymbol{x}}_r$$

Following the same steps, we will obtain:

$$\begin{bmatrix} v_r M_{r3} - M_{r2} \\ u_r M_{r3} - M_{r1} \\ u_r M_{r2} - M_{r2} \end{bmatrix} \widetilde{\boldsymbol{x}}_r = 0$$

3.6 Computing Depth

We rearrange the terms:

$$\begin{bmatrix} u_r m_{31} - m_{11} & u_r m_{32} - m_{12} & u_r m_{33} - m_{13} \\ v_r m_{31} - m_{21} & v_r m_{32} - m_{22} & v_r m_{33} - m_{23} \\ u_l p_{31} - p_{11} & u_l p_{32} - p_{12} & u_l p_{33} - p_{13} \\ v_l p_{31} - p_{21} & v_l p_{32} - p_{22} & v_l p_{33} - p_{23} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} m_{34} - m_{14} \\ m_{34} - m_{24} \\ p_{34} - p_{14} \\ p_{34} - p_{24} \end{bmatrix}$$

A $\qquad$ $\boldsymbol{x_r}$ = b

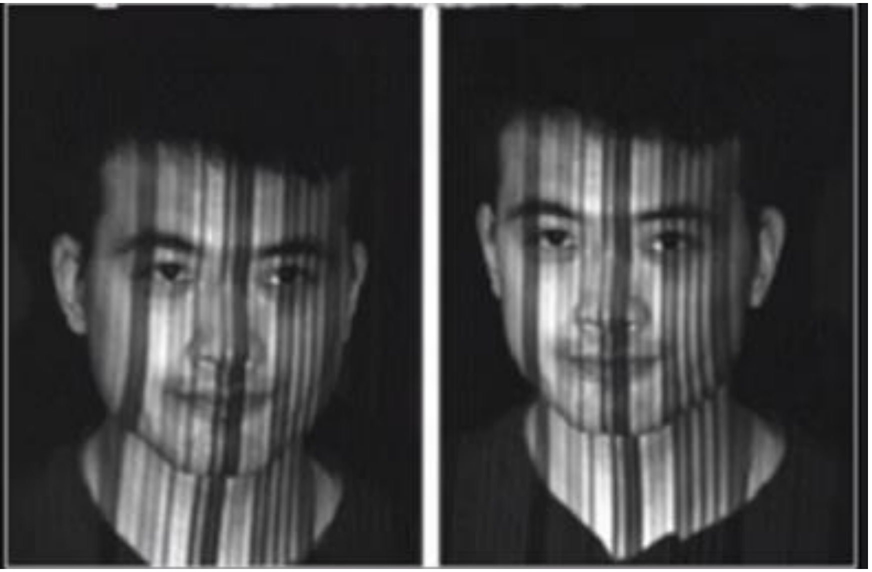
Known $\qquad$ Unknown $\qquad$ Known

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

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Example of computing depth.



Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Example of computing depth.

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
shrink = 0.2
# reduce the image size

K = numpy.zeros((3, 3), numpy.float32)
#Return a new array of given shape and type, filled with zeros.
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
# Initializing the calibration matrix K
###############################################

K[0][0] = 4262*shrink
K[1][1] = 4240*shrink
K[2][2] = 1
K[0][2] = 1162*shrink
K[1][2] = 1623*shrink
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
img1 = cv2.imread("imd.jpg")
img2 = cv2.imread("img.jpg")
img1 = cv2.resize(img1, (0,0), fx=shrink, fy=shrink, interpolation=cv2.INTER_CUBIC)
img2 = cv2.resize(img2, (0,0), fx=shrink, fy=shrink, interpolation=cv2.INTER_CUBIC)

cv2.imshow("img1", img1)
cv2.imshow("img2", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
# Compute sift keypoints and descriptors
############################################

sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
# Matching of keypoints
########################

bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)

if len(matches) < 10:
        img_err = cv2.imread("error.png")
        cv2.imshow("Error", img_err)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        exit()
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
#Finding good matches
######################

nice_match = []
for m, n in matches:
        if m.distance < 0.8 * n.distance:
                nice_match.append([m])
if len(nice_match) < 10:
        img_err = cv2.imread("error.png")
        cv2.imshow("Error", img_err)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        exit()
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
# Draw the matches
###################

img3 = cv2.drawMatchesKnn(img1, kp1, img2, kp2, nice_match[:], None, flags=2)
cv2.imshow("img3", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.6 Computing Depth

Practical aspect

```
#For Fundamental matrix computation
#work only with good matches, find at least 8 matches (returned in masko)
src_ptso = numpy.float32([kp1[m[0].queryIdx].pt for m in nice_match ]).reshape(-1, 1, 2)
dst_ptso = numpy.float32([kp2[m[0].trainIdx].pt for m in nice_match ]).reshape(-1, 1, 2)

H, masko = cv2.findHomography(src_ptso, dst_ptso, cv2.RANSAC, 3, None, 100, 0.99)
matchesMasko = masko.ravel().tolist()
inlier_matcho = []
for i in range(len(nice_match)):
        if matchesMasko[i]:
                inlier_matcho.append(nice_match[i])
if len(inlier_matcho) < 8:
        img_err = cv2.imread("error.png")
        cv2.imshow("Error", img_err)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        exit()
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
#For Fundamental matrix computation
#work only with good matches, find at least 8 matches (returned in mask0)

img3 = cv2.drawMatchesKnn(img1, kp1, img2, kp2, inlier_match0[:], None, flags=2)
cv2.imshow("img3", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
#For Fundamental matrix computation
#work only with good matches, find at least 8 matches (returned in masko)

img3 = cv2.drawMatchesKnn(img1, kp1, img2, kp2, inlier_matcho[:], None, flags=2)
cv2.imshow("img3", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()

src_pts1 = numpy.float32([kp1[m[0].queryIdx].pt for m in inlier_matcho ])
                    .reshape(-1, 1, 2)
dst_pts1 = numpy.float32([kp2[m[0].trainIdx].pt for m in inlier_matcho ])
                    .reshape(-1, 1, 2)
F, mask1 = cv2.findFundamentalMat(src_pts1, dst_pts1, cv2.FM_RANSAC, 1, 0.99)
print("FundamentalMat is:\n", F)
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
#For Essential matrix computation
#work only with good matches, find at least 5 matches (returned in mask1)

matchesMask1 = mask1.ravel().tolist()
inlier_match1 = []
for i in range(len(inlier_match0)):
        if matchesMask1[i]:
                inlier_match1.append(inlier_match0[i])
if len(inlier_match1) < 5:
        img_err = cv2.imread("error.png")
        cv2.imshow("Error", img_err)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        exit()
```

3.6 Computing Depth

Practical aspect

```
#For Essential matrix computation
#work only with good matches, find at least 5 matches (returned in mask1)

print("the matches used for essential matrix")
img3 = cv2.drawMatchesKnn(img1, kp1, img2, kp2, inlier_match1[:], None, flags=2)
cv2.imshow("img3", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
#For Essential matrix computation
#work only with good matches, find at least 5 matches (returned in mask1)

src_pts2 = numpy.float32([kp1[m[0].queryIdx].pt for m in inlier_match1 ])
                            .reshape(-1, 1, 2)
dst_pts2 = numpy.float32([kp2[m[0].trainIdx].pt for m in inlier_match1 ])
                            .reshape(-1, 1, 2)
E, mask2 = cv2.findEssentialMat(src_pts2, dst_pts2, K, cv2.RANSAC, 0.99, 1)
print("EssentialMat is:\n", E)

#Nistér, David. "An efficient solution to the five-point relative pose
#problem."IEEE Transactions on Pattern Analysis and Machine Intelligence, (2004).
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
#For Essential matrix computation
#work only with good matches, find at least 5 matches (returned in mask1)

matchesMask2 = mask2.ravel().tolist()
inlier_match2 = []
for i in range(len(inlier_match1)):
        if matchesMask2[i]:
                inlier_match2.append(inlier_match1[i])
if len(inlier_match2) < 5:
        img_err = cv2.imread("error.png")
        cv2.imshow("Error", img_err)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        exit()
img3 = cv2.drawMatchesKnn(img1, kp1, img2, kp2, inlier_match2[:], None, flags=2)
cv2.imshow("img3", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
#Recover the matrices R and T

src_pts3 = numpy.float32([kp1[m[0].queryIdx].pt for m in inlier_match2 ])
                        .reshape(-1, 1, 2)
dst_pts3 = numpy.float32([kp2[m[0].trainIdx].pt for m in inlier_match2 ])
                        .reshape(-1, 1, 2)
M, R, t, mask3 = cv2.recoverPose(E, src_pts3, dst_pts3, K)
print("Rotation is:\n", R)
print("Translation is:\n", t)
```

Chapitre 3. Uncalibrated stereo

3.6 Computing Depth

Practical aspect

```
#Use the good points for  3D reconstruction

matchesMask3 = mask3.ravel().tolist()
inlier_match3 = []
for i in range(len(inlier_match2)):
        if matchesMask3[i]:
                inlier_match3.append(inlier_match2[i])
if len(inlier_match3) < 5:
        img_err = cv2.imread("error.png")
        cv2.imshow("Error", img_err)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        exit()

img3 = cv2.drawMatchesKnn(img1, kp1, img2, kp2, inlier_match3[:], None, flags=2)
cv2.imshow("img3 used for triang", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.6 Computing Depth

Practical aspect

```
#Use the good points for  3D reconstruction

src_pts4 = numpy.float32([kp1[m[0].queryIdx].pt for m in inlier_match3 ])
                        .reshape(-1, 1, 2)
dst_pts4 = numpy.float32([kp2[m[0].trainIdx].pt for m in inlier_match3 ])
                        .reshape(-1, 1, 2)
src_pts4 = (src_pts4.reshape(-1, 2)).transpose(1,0)
dst_pts4 = (dst_pts4.reshape(-1, 2)).transpose(1,0)
proj1 = numpy.column_stack((numpy.eye(3, dtype=numpy.float32), numpy.zeros((3, 1),
dtype=numpy.float32)))
proj2 = numpy.column_stack((R, t))

points4D = cv2.triangulatePoints(proj1, proj2, src_pts4, dst_pts4)
```

3.6 Computing Depth

Practical aspect

```
#Use the good points for  3D reconstruction
ptList = []
x_3D = []
y_3D = []
z_3D = []
for i in range(len(inlier_match3)):
        ptList.append(kp1[inlier_match3[i][0].queryIdx].pt)
        #print("\n point 3D :")
        x_3D.append((points4D[0][i] / points4D[3][i]))
        #print("x= ",points4D[0][i] / points4D[3][i])
        y_3D.append((points4D[1][i] / points4D[3][i]))
        #print("y= ",points4D[1][i] / points4D[3][i])
        z_3D.append((points4D[2][i] / points4D[3][i]))
        #print("z= ",points4D[2][i] / points4D[3][i])
```