Introduction

The goal of 3D reconstruction is to recover the 3D **properties** of a geometric entity from its 2D images. Depending on the types of geometric entities, the 3D properties may include 3D coordinates of a 3D point (full reconstruction) or it's 3D depth (z coordinate) or its 3D shape (3D orientation) (partial reconstruction) of an object.

For visualization, the recovered 3D data is often represented as an image. Such an image is often referred to as *range* image, *depth* image/map, *surface profiles*, or *2.5-D images*.

Techniques

- Passive Approach

  Shape from monocular images (single image).

  Shape from multiple images (two images and a sequence of images).

- Active approach

  Active stereo with projector (e.g. MS Kinect sensor or Intel RealSense)

  Shape from range sensors (e.g. RADAR or Sonar or LiDAR) that emits either radio, sound or light wave and measures their time of flight.

3D from Optical Images

These techniques recover 3D information from optical images. They can be divided into two groups

- 3D reconstruction from single images

  Collectively referred to as **Shape from X** techniques, these techniques infer 3D data from a single image in conjunction with other visual cues or geometric properties.

- Stereo

  Recover 3D information from two images from different view points (passive stereo) or one image and a projector (active stereo)

- Structure from motion

  Recover 3D structure from a sequence of images or video.

Stereo Vision

Stereo vision is a technique for the reconstruction of the three-dimensional description of a scene from images observed from multiple viewpoints.

Stereo Vision (cont'd)

Passive stereo (for a review see Poggio [14])

Active stereo

Passive Stereo

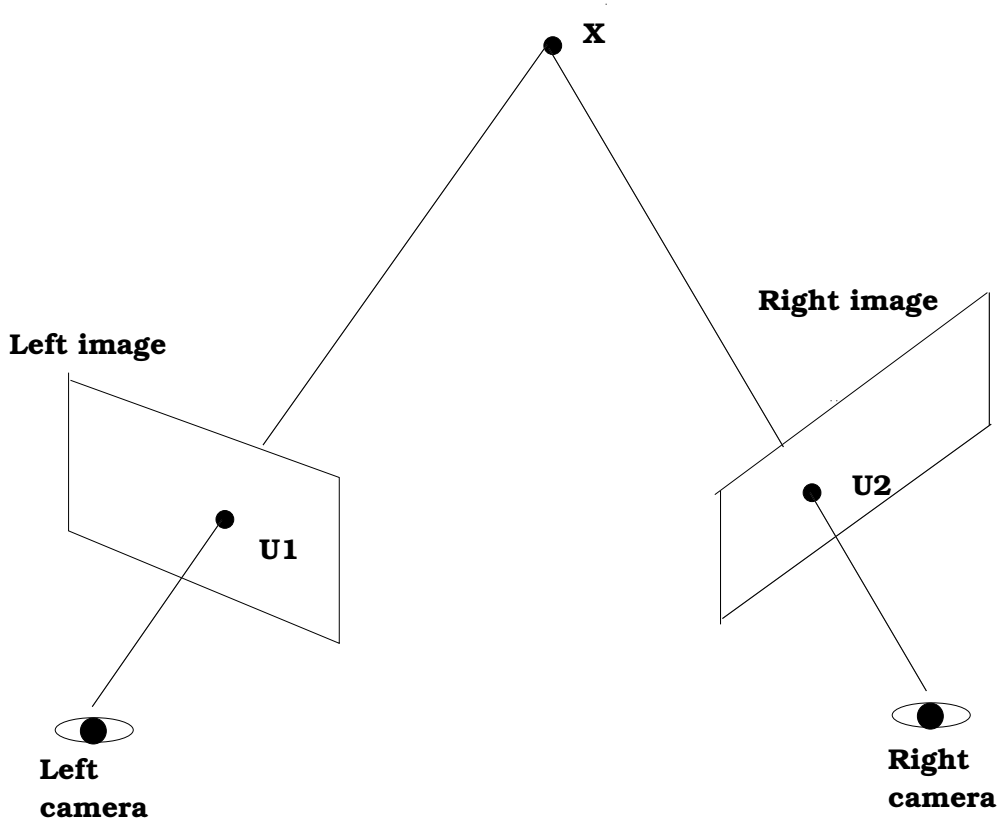
- Token detection.
- Token matching (e.g. pointwise matching).
- 3D reconstruction using the matched tokens.

Different tokens

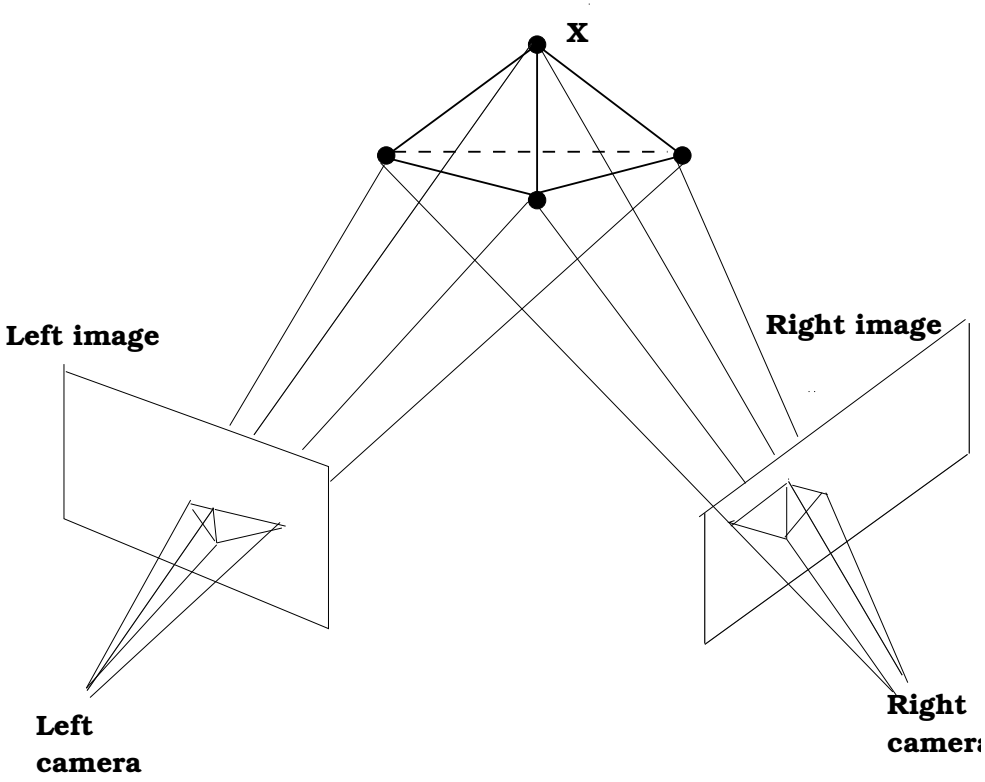
- Points
- Lines
- Conic curves

Passive Stereo Using 2D/3D Points

- Theoretical basis : triangulation



3D Triangulation Example



Passive Stereo

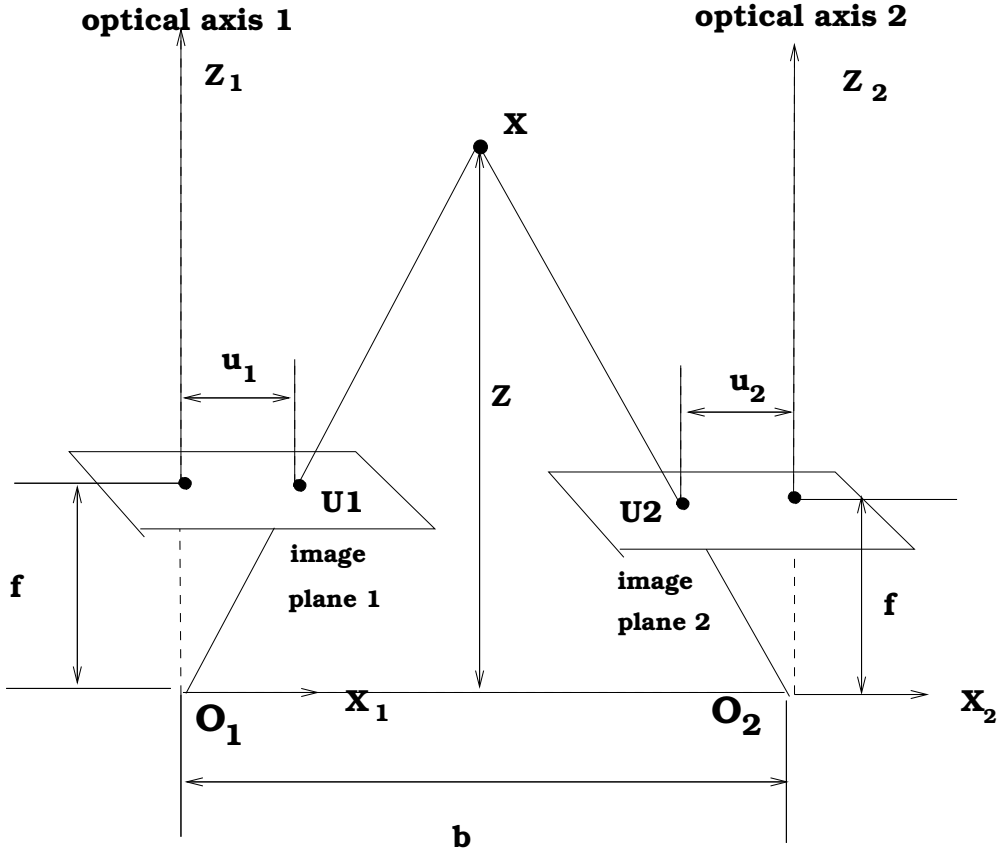
Human eyes fully exploit the idea of passive stereo. Two eyes act like two cameras. They produce two images of the 3D world. Given the two images, our brain performs the triangulation to produce the 3D information.

Passive Stereo (cont'd)

3D movies produce two images of the same world captured by two different cameras are projected onto the screen at the same time.

To enable the left eye only sees the left image and right eye only sees the right image, two images are often made under different spectrum such as old technique of using a red and blue filter or recent technique of through polarization. Current stereoscopic technology separates the stereo frames by polarization. The two images are then merged by the brain to produce the synthetic 3D perception.

Theoretical basis : triangulation under rectified images



Rectified camera geometry: two cameras optical axes parallel to each other, same focus length (image planes co-planar), and

corresponding points on the same image row.

Since triangle $XU1U2$ is similar to triangle XO_1O_2, we have

$$\frac{Z-f}{Z} = \frac{b-u_1+u_2}{b}$$

Hence,

$$Z = \frac{fb}{u_1-u_2}$$

where b is called the **base distance**, $u_1 - u_2$ the **disparity**. It is clear disparity is **inversely** proportional to depth. As depth approaches infinity, disparity approaches zero. Disparity is often used as a surrogate measure of depth.

Major Issues

- Establish correspondences
- Reconstruct 3D points from matched 2D points via triangulation.

Establishing Correspondences

Establishing correspondences amounts to matching points from one image to points in another image such that each matched pair is generated by the same 3D point.

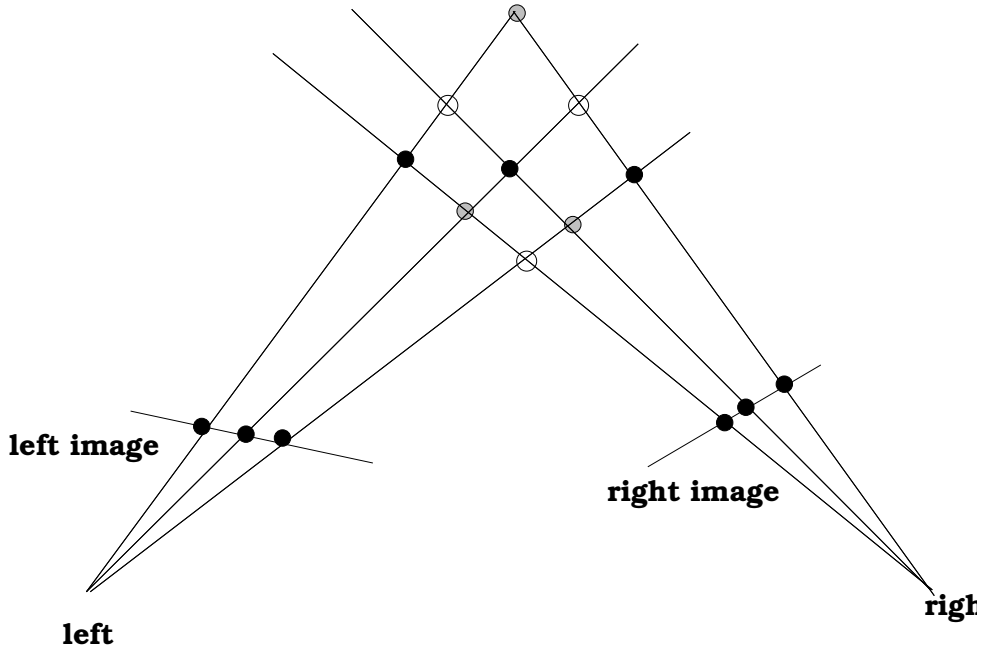
Establishing Correspondences: an example

see Figure 7.1 of Trucco's book

Establishing Correspondences (cont'd)

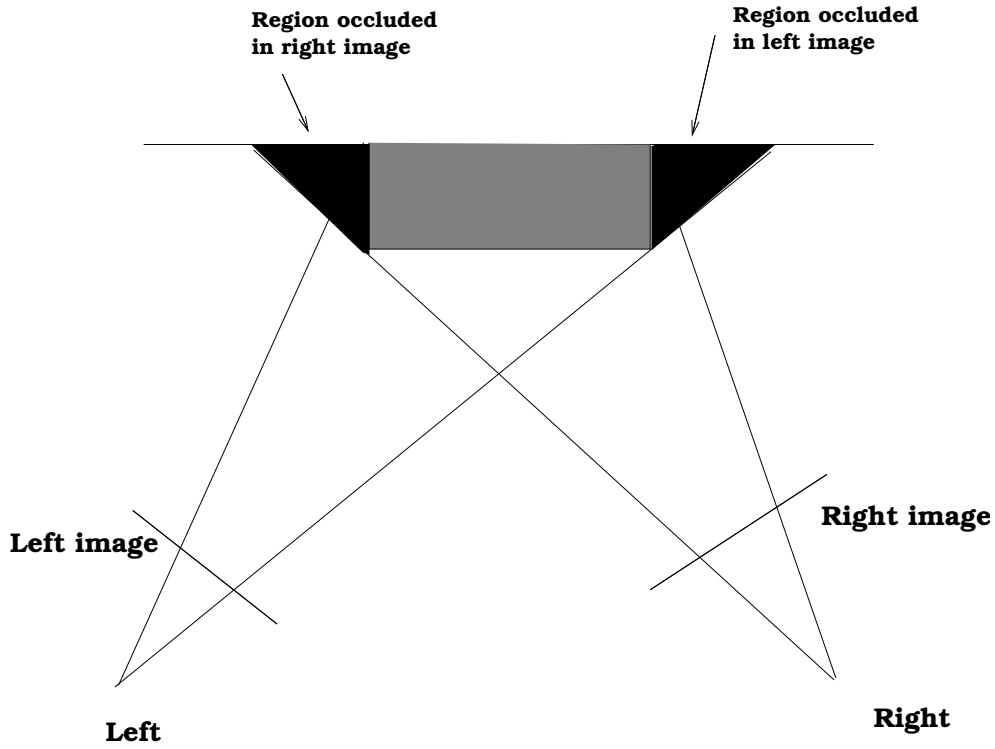
Establishing correspondences is difficult since

- the information associated with each image point is often not sufficient to uniquely establish the pointwise correspondence.



- Image points in one image may not have corresponding points

in another image due to 1) two cameras have different views; 2) occlusion; 3) missing points due to feature detection techniques.



Reducing baseline distance b (narrow-angle-stereo) can alleviate the correspondence and occlusion problem, but it may lead to less accurate depth estimate.

Let $d = u_1 - u_2$ be the disparity, then we have

$$Z = \frac{bf}{d}$$

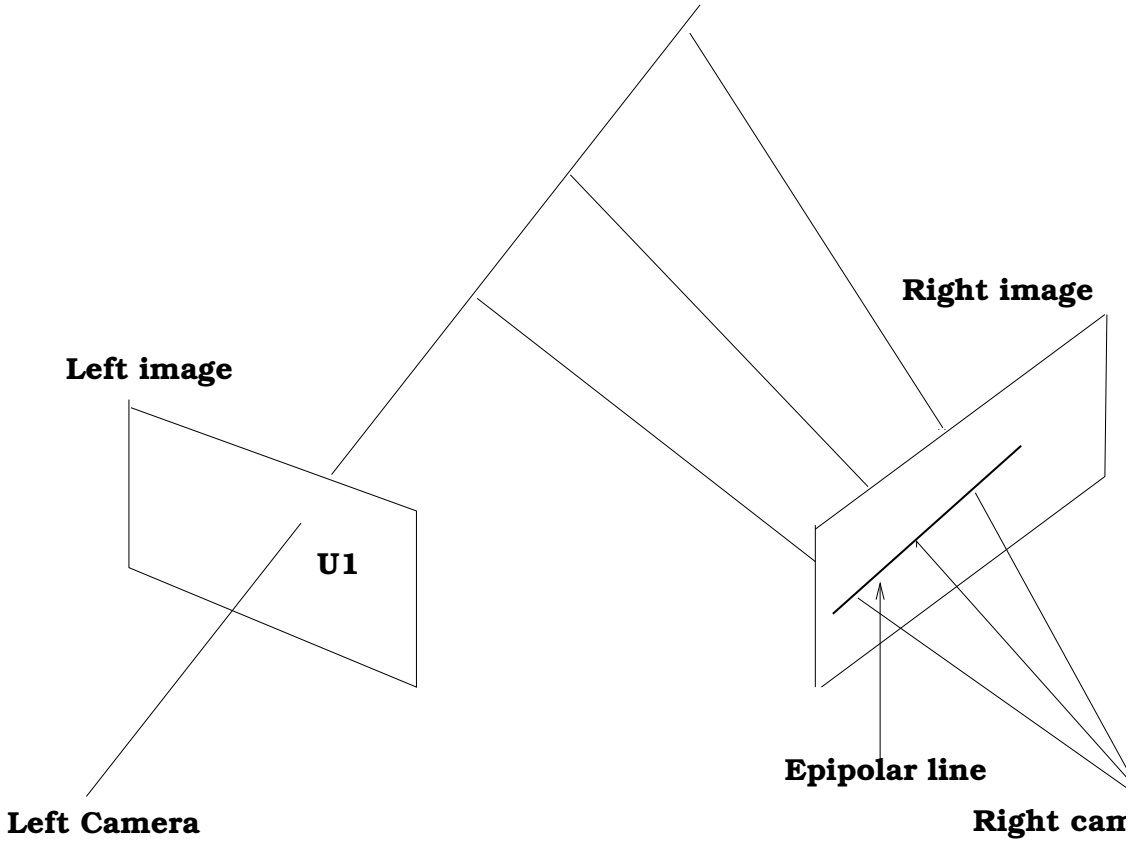
Linearizing both sides of the above equation yields

$$\Delta Z = \frac{f\Delta b}{d} - \frac{fb\Delta d}{d^2}$$

It is clear that as b decreases, ΔZ increases.

Epipolar Constraint

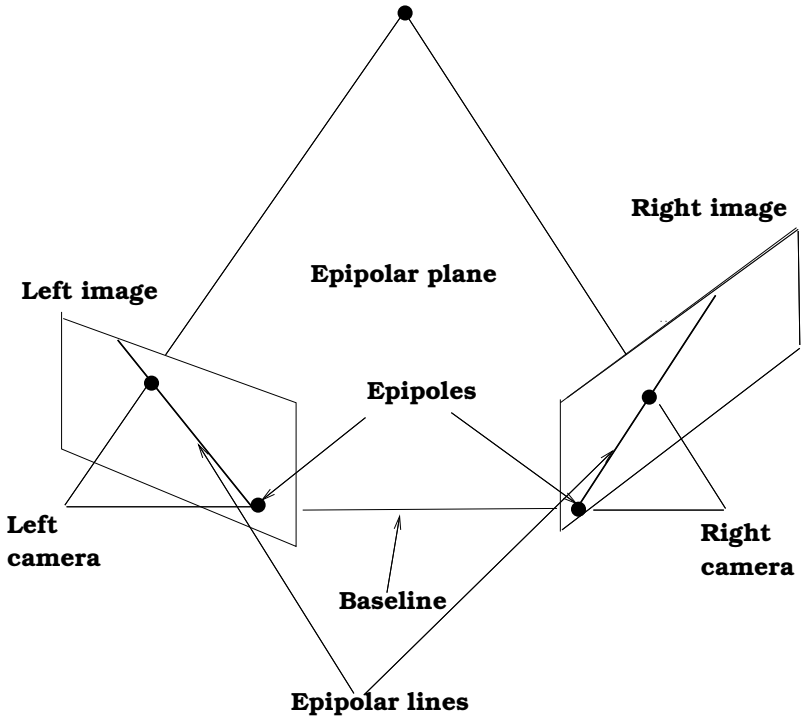
Pointwise matching can be established more efficiently using a simple yet powerful constraint: **Epipolar constraint**



Epipolar Constraint (cont'd)

Epipolar constraint says that given U_1 from the left image, its corresponding point on the right image must lie on the epipolar line. Epipolar line is the projection on the right image by the 3D line going through U_1 and the left camera center. This effectively reduces the point search from 2D to 1D, substantially reducing the search time.

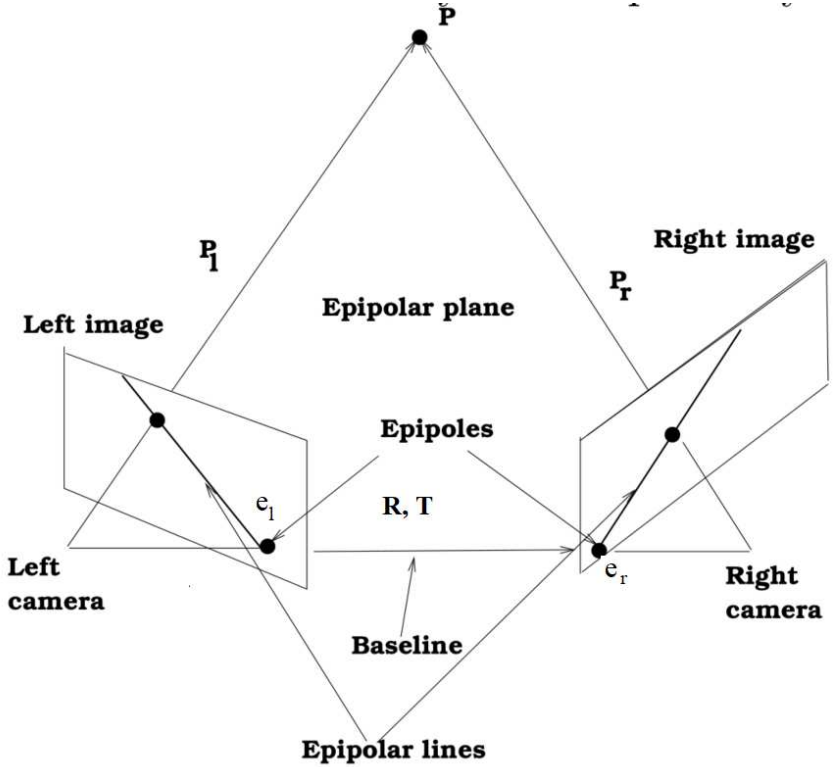
Epipolar Geometry



All epipolar lines go through the epipole. The epipole on the left (right) image is the image of the optical center of the right (left) camera. In rectified stereo images, epipolar lines are parallel and conjugate epipolar lines are collinear.

Computing Epipolar Geometry

Let P be a 3D point and P_l and P_r be the coordinates of P in the left and right camera coordinate system respectively.



Then we have

$$P_l = RP_r + T$$

or

$$P_r = R^T(P_l - T)$$

where R and T specify the relative orientation of the right camera frame with respect to the left one. The coplanar constraint on P_l, P_r, and T leads to

$$(T \times P_l)^T(P_l - T) = 0$$

Since $P_l - T = RP_r$, we have

$$(T \times P_l)^T RP_r = 0$$

Since $(T \times P_l)$=S P_l, where S a skew matrix

$$S = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$$

As a result, we have

$$P_l^T E P_r = 0$$

where $E = S^T R$ is called the **essential matrix**. Note by the construct of S, E has a rank of 2 and two of its singular values equal and the third is 0.

Essential Matrix

The essential matrix E establishes a link between the epipolar constraint and the relative orientation (rotation and translation) of the two coordinate systems. It has only 5 degrees of freedom.

Fundamental Matrix

P_l and P_r represent the 3D coordinates of P in the left and right camera coordinate systems respectively. Let U_l and U_r be the homogeneous pixel coordinates of point P in the left and right images. We know

$$\lambda_l U_l = W_l P_l$$

$$\lambda_r U_r = W_r P_r$$

where W_l and W_r are two 3×3 matrices involving the intrinsic left and right camera parameters.

Substituting the above equations into the essential matrix equation yields

$$U_l^T F U_r = 0$$

where $F = W_l^{-T} E W_r^{-1}$ is called the **fundamental matrix**.

F algebraically encodes the epipolar constraint.

What is the rank of F ?

The rank of F remains 2 due to E.

See table 8.1 (p226) of Hartley's book for a summary of epipolar geometry and the fundamental matrix.

Fundamental Matrix (cont'd)

Unlike the essential matrix E which only encodes the information about the relative orientation between the two cameras, the fundamental matrix F encodes both the relative orientation (extrinsic parameters) and the intrinsic parameters. It only has 7 degrees of freedom.

Epipolar lines determination

Given a point U_l on the left image, the equation of the epipolar line on the right image is: $F^T U_l$. Note a line equation on the right plane can be written as $\alpha c_r + \beta r_r + \gamma = 0$, where (α, β, γ) are line coefficients and they are equal to $F^T U_l$

Given a point U_r on the right image, the equation of the epipolar line on the left image is: FU_r.

Determination of Epipoles

Let the image coordinates of the left and right epipoles be e_l and e_r respectively. Since every epipolar line passes through the epipoles, for any image point on the left U_l, we have

$$U_l^T F e_r = 0$$

Since U_l in general is non-zero, so we have

$$F e_r = 0$$

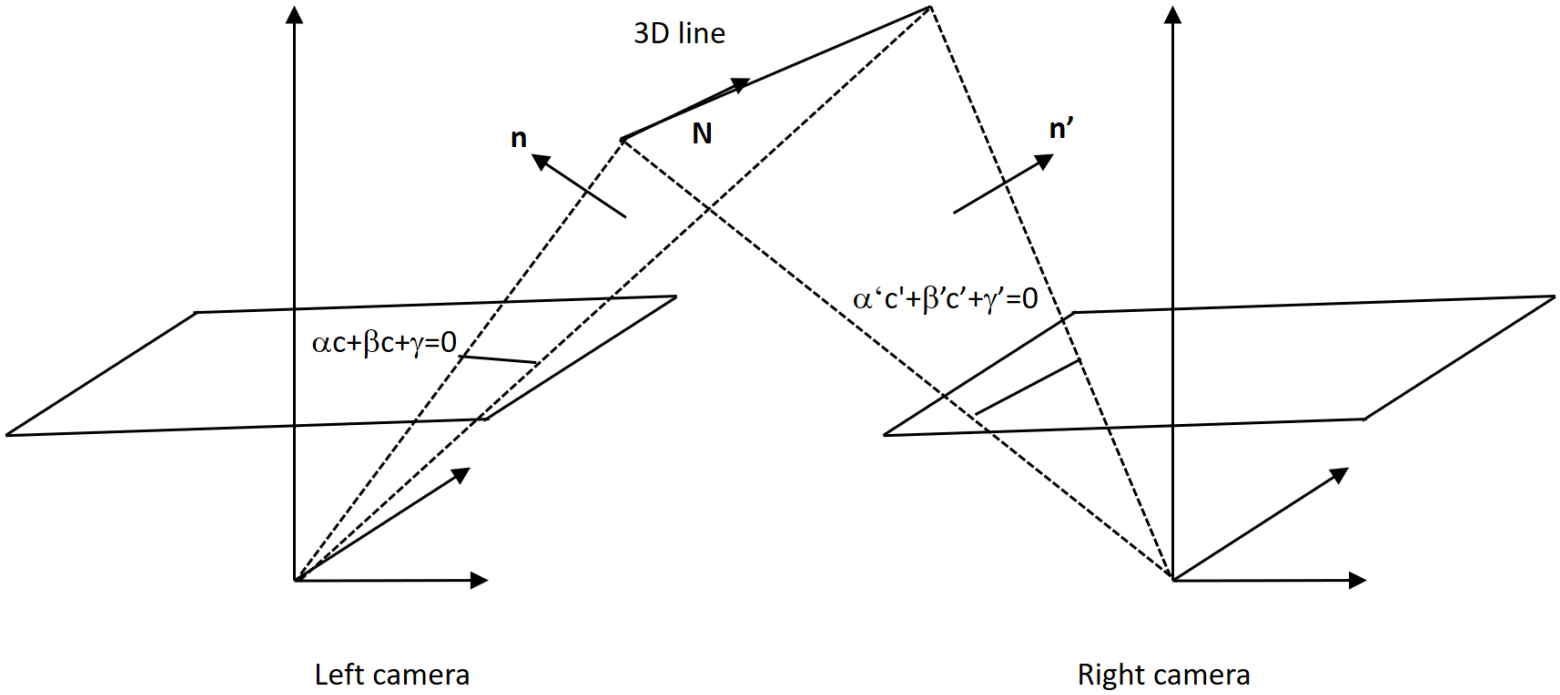
which means that the solution to e_r is the eigen vector of F that corresponds to zero eign value (the null eigen vector).

Similarly, we will find that e_l is the null vector of F^T. Refer to page 157 of Trucco's book and page 219 of Forsyth's book on the step by step algorithm to compute epipoles.

Hence, F <===> Epipoles , in other words, given F, we can

compute epioles or given epipoles we can obtain F.

Epipolar Geometry for Lines and Circles



As shown in the figure above, the left line equation is $\alpha c + \beta r + \gamma = 0$ and the corresponding right line equation is $\alpha' c' + \beta' r' + \gamma' = 0$. Given the line on the left image, we can

obtain the equation for the backprojection plane that contains the 3D line and the left camera center as

$$\begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix}^T W_l^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = 0 \tag{1}$$

Similarly, we can obtain the right backprojection plane equation as

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix}^T W_r^T \begin{pmatrix} \alpha' \\ \beta' \\ \gamma' \end{pmatrix} = 0 \tag{2}$$

where the left 3D coordiantes $\begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix}$ and the right 3D coordinates $\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix}$ are related by R and T as

$$\begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} = R \begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} + T \tag{3}$$

Substituting Eq. 3 into Eq. 1 yields

$$(R\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} + T)^T W_l^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = 0 \tag{4}$$

Or

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix}^T R^T W_l^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = -T^T W_l^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \tag{5}$$

Combining Eq. 2 with Eq. 5 by solving $\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix}$ produces

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = A^{-1}b \tag{6}$$

where

$$A = \begin{bmatrix} \begin{pmatrix} \alpha' \\ \beta' \\ \gamma' \end{pmatrix}^T & W_r \\ \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}^T & RW_l \end{bmatrix}$$

$$b = \begin{bmatrix} 0 \\ -T^T W_l^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \end{bmatrix}$$

Substituting Eq. 6 into Eq. 2 yields

$$(A^{-1}b)^T W_r^T \begin{pmatrix} \alpha' \\ \beta' \\ \gamma' \end{pmatrix} = 0 \tag{7}$$

Eq. 7 relates image line parameters with the stereo parameters W_l, W_r, R, and T.

Alternatively, let $(c_1, r_1, 1)^T$ and $(c_2, r_2, 1)^T$ be two points on the line segment on the left image, and $(c'_1, r'_1, 1)^T$ and $(c'_2, r'_2, 1)^T$ be the corresponding points on the corresponding line segment on the right image.

Then from fundamental equation we have,

$$\begin{pmatrix} c_1 \\ r_1 \\ 1 \end{pmatrix}^T F \begin{pmatrix} c_1' \\ r_1' \\ 1 \end{pmatrix} = 0$$

$$\begin{pmatrix} c_2 \\ r_2 \\ 1 \end{pmatrix}^T F \begin{pmatrix} c_2' \\ r_2' \\ 1 \end{pmatrix} = 0 \quad (8)$$

Hence,

$$\begin{pmatrix} c_1 - c_2 \\ r_1 - r_2 \\ 0 \end{pmatrix}^T F \begin{pmatrix} c_1' - c_2' \\ r_1' - r_2' \\ 0 \end{pmatrix} = 0 \quad (9)$$

$$\begin{pmatrix} c_1 - c_2 \\ r_1 - r_2 \end{pmatrix}^T F_2 \begin{pmatrix} c'_1 - c'_2 \\ r'_1 - r'_2 \end{pmatrix} = 0 \quad (10)$$

where F_2 is the first 2×2 sub-matrix of F. From the line equations, we have

$$\begin{pmatrix} c_1 - c_2 \\ r_1 - r_2 \end{pmatrix}^T \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = 0$$

$$\begin{pmatrix} c'_1 - c'_2 \\ r'_1 - r'_2 \end{pmatrix}^T \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = 0 \quad (11)$$

Substituting Eq. 11 into 10 yields the epipolar geometry for lines

as

$$\begin{pmatrix} \beta \\ -\alpha \end{pmatrix}^T F_2 \begin{pmatrix} \beta' \\ -\alpha' \end{pmatrix} = 0 \tag{12}$$

Let n and n' be respectively the normal to the back projection plane on the left and right image. We have

$$n = \frac{W_l^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}}{||W^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}||_2}$$

$$n' = \frac{W_r^T \begin{pmatrix} \alpha' \\ \beta' \\ \gamma' \end{pmatrix}}{||W_r^T \begin{pmatrix} \alpha' \\ \beta' \\ \gamma' \end{pmatrix}||_2}$$

With respect to the left camera frame, n' becomes Rn'. The orientation of the 3D line N w.r.t the left camera frame that produces the two image lines can be computed as

$$N = n \times Rn'$$

Alternatively, if we use point plus orientation representation for a line, any point on the left image can be expressed as

$$\begin{pmatrix} c \\ r \end{pmatrix} = \begin{pmatrix} c_0 \\ r_0 \end{pmatrix} + \lambda \begin{pmatrix} N_c \\ N_r \end{pmatrix}$$

It can be re-written as $\begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} + \lambda \begin{pmatrix} N_c \\ N_r \\ 0 \end{pmatrix}$

Similarly for a point on the right image, we have

$$\begin{pmatrix} c' \\ r' \\ 1 \end{pmatrix} = \begin{pmatrix} c'_0 \\ r'_0 \\ 1 \end{pmatrix} + \lambda' \begin{pmatrix} N'_c \\ N'_r \\ 0 \end{pmatrix}$$

Plugging the two equations above into the fundamental equation yields

$$[\begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} + \lambda \begin{pmatrix} N_c \\ N_r \\ 0 \end{pmatrix}]^T \; F \; [\begin{pmatrix} c'_0 \\ r'_0 \\ 1 \end{pmatrix} + \lambda' \begin{pmatrix} N'_c \\ N'_r \\ 0 \end{pmatrix}]=0.$$ After simplification, we have

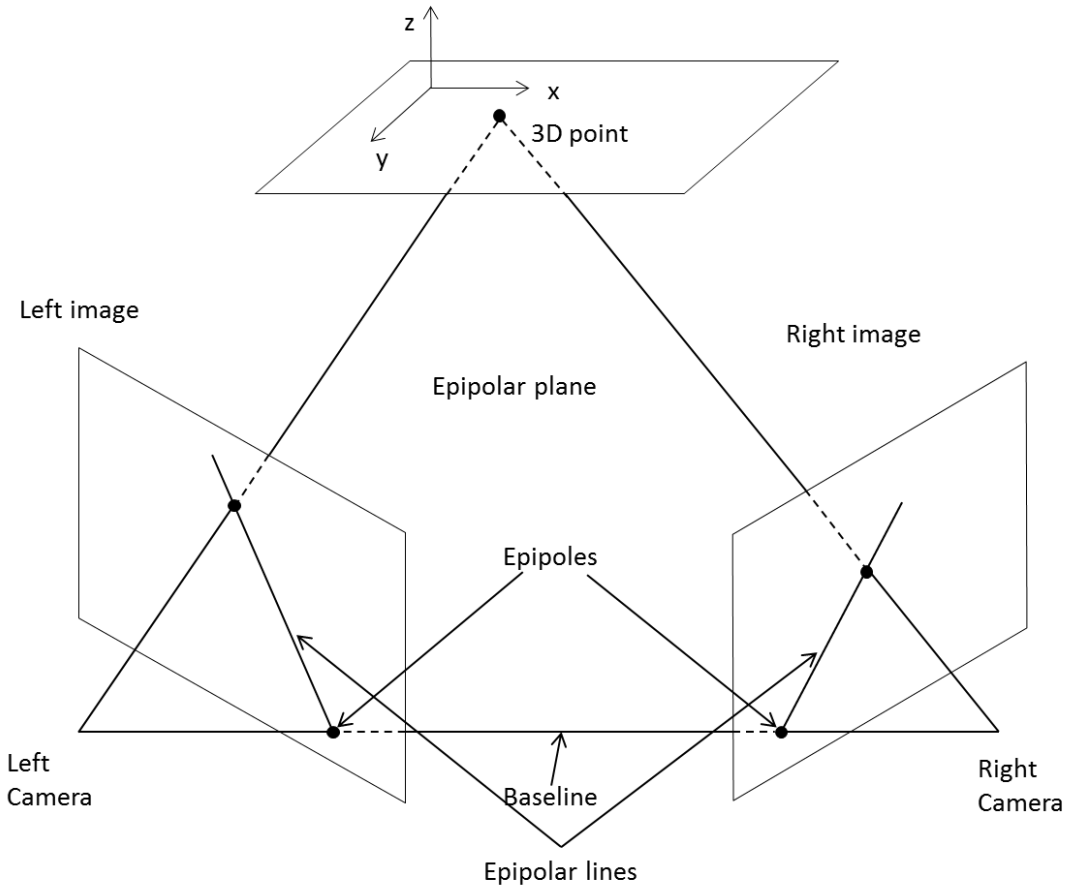
$$\lambda' \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} F \begin{pmatrix} N'_c \\ N'_r \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} N_c \\ N_r \\ 0 \end{pmatrix} F \begin{pmatrix} c' \\ r' \\ 1 \end{pmatrix} =0$$

Geometrically, we can also think two epipolar planes, one constructed by P_0 and the left and right camera center and another by P and the left and right camera center, where P_0 is a given point on the 3D line and P is any point on the line.

Homography

Fundamental equations is true for any 3D point. But its mapping is one to many and cannot uniquely determine the corresponding points. To have a one-to-one mapping, assumption must be made about the 3D point. If we assume it is located on a plane (note other geometric constraints can also apply) or only pure rotation involves between the left and right images, the correspondence becomes unique. Such a unique correspondence relationship can be characterized by homography.

Homography is to establish relationship between two corresponding points in two different images, assuming the corresponding image points are generated by 3D planar points. With this, a unique relationship can established between the two matched points.


z
x
y
3D point
Left image
Right image
Epipolar plane
Epipoles
Left
Camera
Baseline
Right
Camera
Epipolar lines

$$
\begin{aligned}
\lambda_l \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} &= W_l \begin{bmatrix} R_l & T_l \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \\
&= W_l \begin{bmatrix} R_l & T_l \end{bmatrix} \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} \\
&= W_l \begin{bmatrix} R_{l2} & T_l \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad (13) \\
&= W_l^{3\times 3} M_{l2}^{3\times 3} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad (14)
\end{aligned}
$$

where R_{l2} is the first two columns of R_l, $M_{l2} = \begin{bmatrix} R_{l2} & T_l \end{bmatrix}$.
Similarly, for the right camera, we have,

$$\lambda_r \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix} = W_r M_{r2} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{15}$$

Combining Eqs (1) and (2) by eliminating $\begin{bmatrix} x & y & 1 \end{bmatrix}^T$ yields,

$$\lambda_l M_{l2}^{-1} W_l^{-1} \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} = \lambda_r M_{r2}^{-1} W_r^{-1} \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix}$$

$$\lambda \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} = W_l M_{l2} M_{r2}^{-1} W_r^{-1} \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix} \quad (16)$$

where $\lambda = \frac{\lambda_l}{\lambda_r}$.

Let $\mathbf{H} = W_l M_{l2} M_{r2}^{-1} W_r^{-1}$, we have,

$$\lambda \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix}$$

This is the Homography equation, and $\mathbf{H}$ is the homography matrix. $\mathbf{H}$ is a function of the camera parameters, independent of points. But note H varies with the plane on which 3D points are located as the plane equation is determined by R_l, T_l, R_r, and T_r.

Homography-pure rotation

When there is only a pure rotation between the left and right camera frame, we have $P_l = RP_r$.

$$\lambda \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} = W_L R W_r^{-1} \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix} = H \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix} \tag{17}$$

Homography Matrix

When the 3D scene are planar or when only rotation is involved between the two cameras, the two corresponding points are uniquely related via the homography. Homography matrix describes completely the relationship between corresponding points. Let H be the homography matrix,

$$\lambda \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} = H \begin{pmatrix} c_r \\ c_r \\ 1 \end{pmatrix}$$

with F, with one to many mapping (i.e. one point maps to many points on the epipolar line). With H, we have a one-to-one mapping.

Note H and F can co-exist for planar 3D point. But pure

rotation, F does not exist, i.e., for stereo, two images must be taken from two different view points.

Stereo Calibration

Stereo calibration involves determining the parameters of a stereo system using corresponding 2D points from the left and right images. The parameters of a stereo system include the **intrinsic** parameters (focal length and principle points) W_l and W_r, and the **extrinsic** parameters R and T including the rigid transformation describing the relative position and orientation of two cameras.

Stereo Calibration

Stereo calibration can be done either through separate calibration of each camera or joint calibration of two cameras via fundamental matrix.

Stereo Calibration by Each Camera

We can perform calibration on the left and right camera separately, producing W_l, R_l, and T_l for the left camera and W_r, R_r, and T_r for the right camera respectively.

We can then have (try to prove this yourself)

$R = R_l R_r^T$

$T = T_l - RT_r$

Stereo Calibration using Fundamental Matrix

Like conventional camera calibration, it involves two steps: recover F and extract camera parameters from F.

Given n pairs of corresponding points, a system of linear equations can be established involving the elements of F. F can then be solved as a linear least-squares problem by minimizing

$$\sum_{i=1}^{n} \begin{pmatrix} c_{l_i} \\ r_{l_i} \\ 1 \end{pmatrix}^T F \begin{pmatrix} c_{r_i} \\ r_{r_i} \\ 1 \end{pmatrix}$$

subject to rank (F)=2

What is the minimum value of n ?

Solution to F

Given each pair of matched 2D points $U_l = (c_l, r_l, 1)^T$ and $U_r = (c_r, r_r, 1)^T$ and the fundamental equation, $U_l^T F U_r = 0$, offers a linear equation for the 9 unknowns in F.

Let F_1, F_2, and F_3 be the three columns of F matrix, the fundamental equation $U_l^T F U_r = 0$ can be written re-written as

$$U_l^T F_1 c_r + U_l^T F_2 r_r + U_l^T F_3 = 0$$

which can be written as

$$(U_l^T c_r \quad U_l^T r_r \quad U_l^T F_3) V = 0$$

where $V^{9\times 1} = (F_1 \quad F_2 \quad F_3)$

Each pair of left and right point produces one linear equation for F. Given N pairs of points (U_{l_i}, U_{r_i}), i=1,2,...,N, we can solve V

by minimizing $||AV||_2^2$, which leads to

$$AV = 0$$

where A is defined as follows

$$A^{N\times 9} = \begin{pmatrix} U_{l_1}^T c_{r_1} & U_{l_1}^T r_{r_1} & U_{l_1}^T \\ \vdots & & \\ U_{l_N}^T c_{r_N} & U_{l_N}^T r_{r_N} & U_{l_N}^T \end{pmatrix}$$

So, given a minimum of 8 pairs of points (cannot be co-planar), V can be solved, depending on the rank of A.

If rank(A)=8, the solution to V is the only null vector of A, and the solution is up to a scale factor. See Algorithm EIGHT_POINT on page 156 of Trucco's book for details.

If rank(A) < 8, then there many solutions to V, equal to linear

combinations of all null vectors of A.

Since F is singular, SVD can be used to find another matrix F' that is closest to the computed F but still singular.

$$F = UDS^T$$

setting the smallest value in D to zero yields D'

$$F' = UD'S^T$$

Alternative, we can impose rank(F)=2 during estimation of F, i.e., find V by minimizing

$$||AV||_2^2$$

subject to rank(F)=2, which can be implemented as $F_3 = k_1F_1 + k_2F_2$, where F_i represents the ith column of F. The constraint provides 3 additional linear equations for F but it also

introduces two unknown parameters k_1 and k_2. This will lead to a total of 11 unknowns with $N + 3$ equations. A minimum of 7 points is enough to solve for F up to a scale factor. But the solution will no longer be linear as $F_3 = k_1 F_1 + k_2 F_2$ involves multiplications of F_i with k_1 and k_2.

Nonlinear solution to F

The linear solution is often not accurate and it can be improved with a non-linear optimization method proposed by Luong ea al (1993). It vastly improves the results. Their non-linear method minimizes

$$\sum_{i=1}^{N}[(U_{l_i}^T F U_{r_i})^2 + (U_{r_i}^T F^T U_{l_i})^2]$$

subject to rank(F)=2

Without imposing the rank constraint, the above equation can be implemented in a linear manner with two A matrices (one for first term and one for second term) on top of each other. However, adding the rank constraint via $F_3 = k_1 F_1 + k_2 F_2$, the solution can no longer be linear. First order and second order non-linear

methods may be used.

To further improve the results, Hartley introduced a normalization procedure (page 156 of Trucco's book) to avoid numerical instability due to ill-conditioned matrix.

Torr and Fitzgibbon (PAMI,BMVC, 2003) show that the 8-point may not be the best method since its results depend on the coordinate system used. They proposed an invariant approach to estimate the fundamental matrix.

Degeneracies

The solution to F may not be unique (not even up to a scale factor) when degeneracies occur. Degeneracies occur if the rank of matrix A is less than 8. Degeneracy may occur because of a special configuration of the 3D control points such as coplanar points (or points on a special quadric surface) or because of special configuration between two cameras such as pure rotation. The former may be referred to as *critical surface* while the latter may be referred to as *critical motion*. See section 10.9 of Hartley's book for more in-depth discussion on degeneracies.

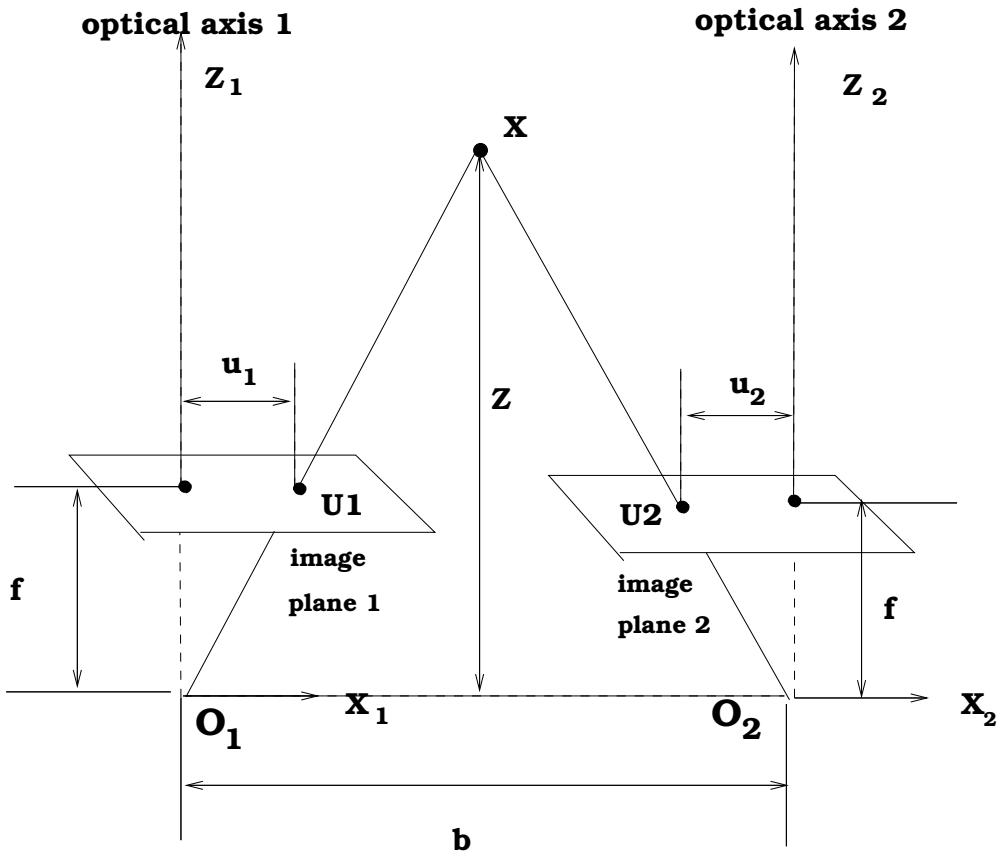
Stereo Calibration

Given F, it is still difficult to simultaneously recover W_l, W_r, R, and T. If the two cameras are calibrated, we can recover E from F using $E = W_l^T F W_r$. Given E, we can then recover R and T using Horn's method (to be discussed later)

Rectified Stereo Geometry

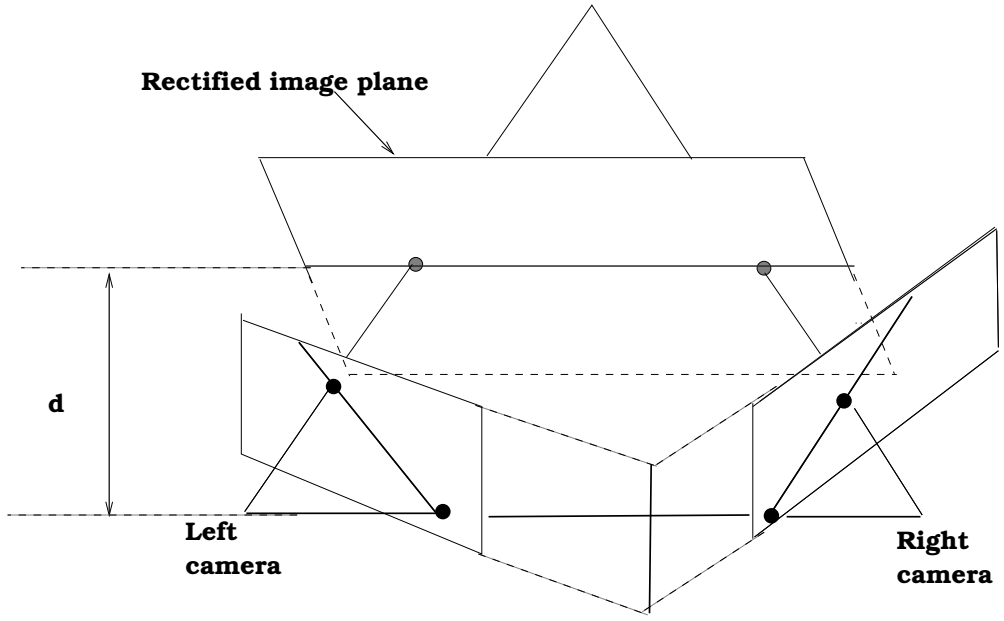
For computational convenience, the two image planes are often chosen to be coplanar and parallel to their base line (this means equality in focus length). Such an arrangement can be accomplished either physically or through analytic transformation. This arrangement makes the search for correspondence points much easier. The corresponding point for any point on the left image may be found on the same row in the right image.

Rectified Geometry



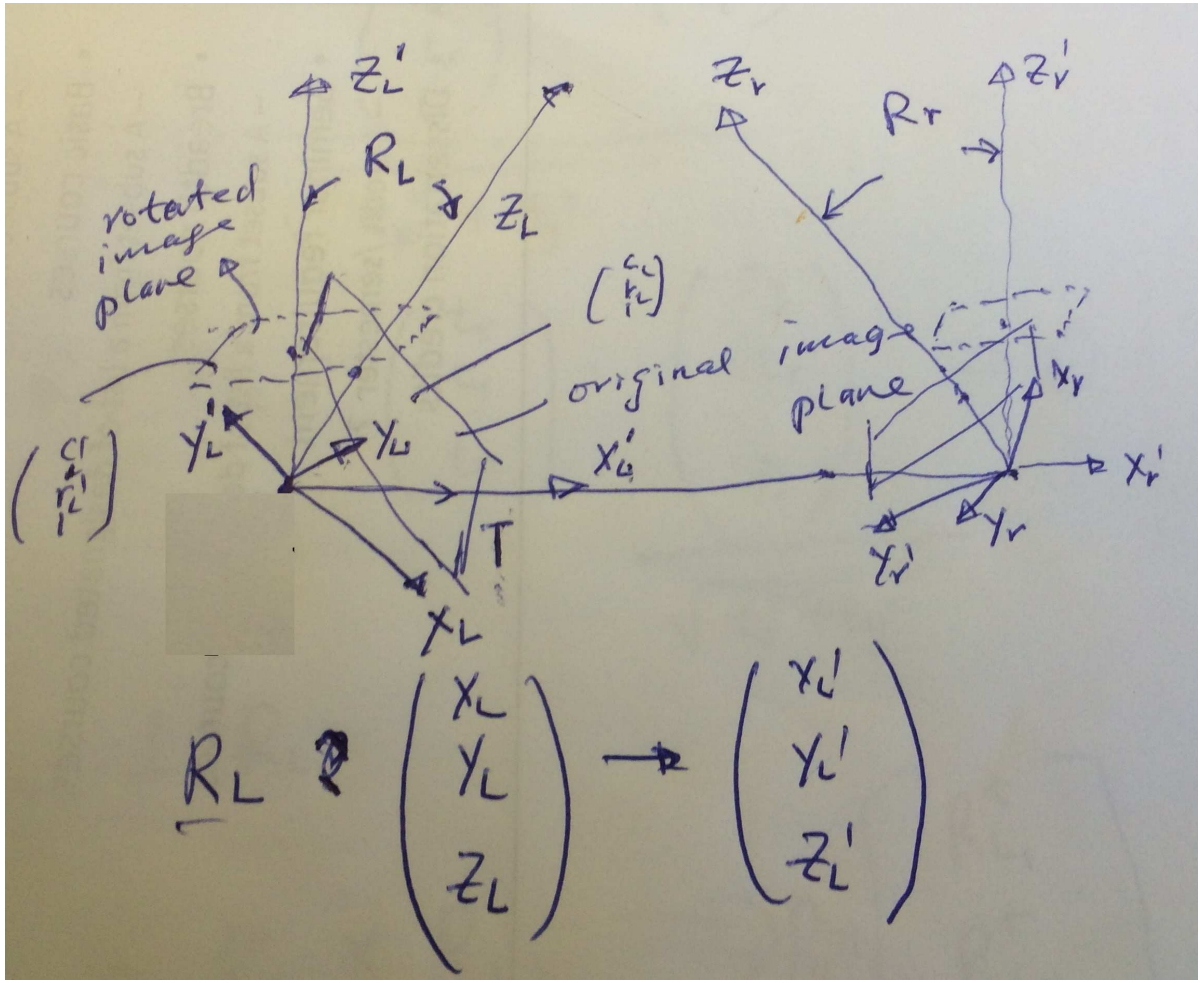
Rectification

Given a calibrated stereo, we can compute the rotation matrices needed to rotate the cameras such that the conjugate epipolar lines are collinear, the rectified left image and right images are co-planar and they are parallel to the base line (note the scale factor d can vary). The new image coordinates are obtained by projecting the original coordinates on the new image plane.



Also see figure 7.8 of Trucco's book.

Rectification Algorithm



Rectification Algorithm

Assuming the relative orientation between the two cameras has been obtained via a stereo calibration procedure, i.e., we have R and T, which specify the relative orientation and translation of the right camera frame with respect to that of the left camera frame.

- Identify a rotation matrix R_l such that when it is applied to the left camera, the image plane of the left camera is parallel to the base line. Note R_l specifies the relative ordination of camera frame before rotation to the camera frame after rotation. For example, let T be the baseline vector relative the original left camera frame and let $(1\ 0\ 0)^T$ be the x-axis after rotation, we then have $R_l \frac{T}{||T||} = (1\ 0\ 0)^T$. This means the first row of R_l is $\frac{T}{||T||}$. Let r_{l_1}, r_{l_2},and r_{l_3} be the three

rows of R_l, and

$$\frac{T}{||T||} = \frac{\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}}{\sqrt{t_x^2 + t_y^2 + t_z^2}} = \begin{pmatrix} t'_x \\ t'_y \\ t'_z \end{pmatrix} \tag{18}$$

Then

$$r_{l_1} = \begin{pmatrix} t'_x \\ t'_y \\ t'_z \end{pmatrix}^T \tag{19}$$

since rows of R_l are orthogonal to each other, we can set

$$r_{l_2} = \frac{1}{\sqrt{t_x'^2 + t_y'^2}} \begin{pmatrix} t_y' \\ -t_x' \\ 0 \end{pmatrix}^T \tag{20}$$

Note this step shows that the selection of the r_{l_2} is arbitrary as long as it is orthogonal to r_{l_1}. This freedom means, with a different r_{l_2}, we may end up with a different rotated left camera frame and hence a different rectified left image. It is only natural to ask if the different rectified images are the same or not. If not, is it possible to select a r_{l_2} that can lead to the best rectified image in terms of distortion, coverage, etc.??

Finally,

$$r_{l_3} = r_{l_1} \times r_{l_2} \tag{21}$$

- Create the rectified left image

  Let $U_l = (c_l, r_l, 1)^T$ and $U_l' = (c_l', r_l', 1)^T$ be the corresponding image point on the left camera before and after rectification. They can be related

$$\lambda \begin{pmatrix} c_l' \\ r_l' \\ 1 \end{pmatrix} = W_l' R_l W_l^{-1} \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} \tag{22}$$

  where λ is a scale factor and W_l' is the intrinsic camera matrix for the rotated left camera. W_l' is usually equal to W_l

but it can be different and can be changed if needed. The above equation is obtained from equations $\lambda(c_l\ r_l\ 1)^T = W(x_c\ y_c\ z_c)^T$, $\lambda(c'_l\ r'_l\ 1)^T = W'(x'_c\ y'_c\ z'_c)^T$, and and $(x'_c\ y'_c\ z'_c)^T = R_L(x_c\ y_c\ z_c)^T$. Because of pure rotation, an image point on the left image before rotation can be uniquely mapped to the corresponding left image after rotation through the homography matrix $W'_l R_l W_l^{-1}$. As a result, we can solve for (c'_l, r'_l) uniquely.

- Find the rotation matrix for the right camera

  For the right camera, we cannot follow the same procedure as we did for the left camera to produce R_r as the rotated right camera image plane must be co-planar with the rotated left image plane. This means after rotation, the left and right camera frames must have the same orientation, plus a translation, i.e., $(x'_l, y'_l, z'_l)^T = (x'_r, y'_r, z'_r)^T + T \Rightarrow$

$R_l(x_l, y_l, z_l)^T = R_r(x_r, y_r, z_r)^T + T$. Combining it with the fact that $(x_l, y_l, z_l)^T = R(x_r, y_r, z_r)^T + T$, we have $R_r = R_l R$

- Create the rectified image on the right camera

  Apply $W_r' R_r W_r^{-1}$ to each point on the right image to compute the new image coordinates for the rectified right image using equation 22.

- Adjust the scale d (fs_x and fs_y) in W_l' and W_r' appropriately so that the rectified image fits to the original image size. The same scale factor should be applied to the row and column for both left and right images.

Please note W_l' and W_r' are the intrinsic matrix for the left and right camera and they are assumed to the same. The rectification procedure needs be modified if W_l' and W_r' are different. One way is to compute $W' = 0.5(W_l' + W_r')$. Then, replace both W_l' and

W_r' by W'. Note only W_l' and W_r' are changed to W'. W_l and W_r remain unchanged.

Rectification

Questions: 1) how to find the best r_{l2}, and (2) how to adjust the scale factor $d = f s_x$

Backward Mapping

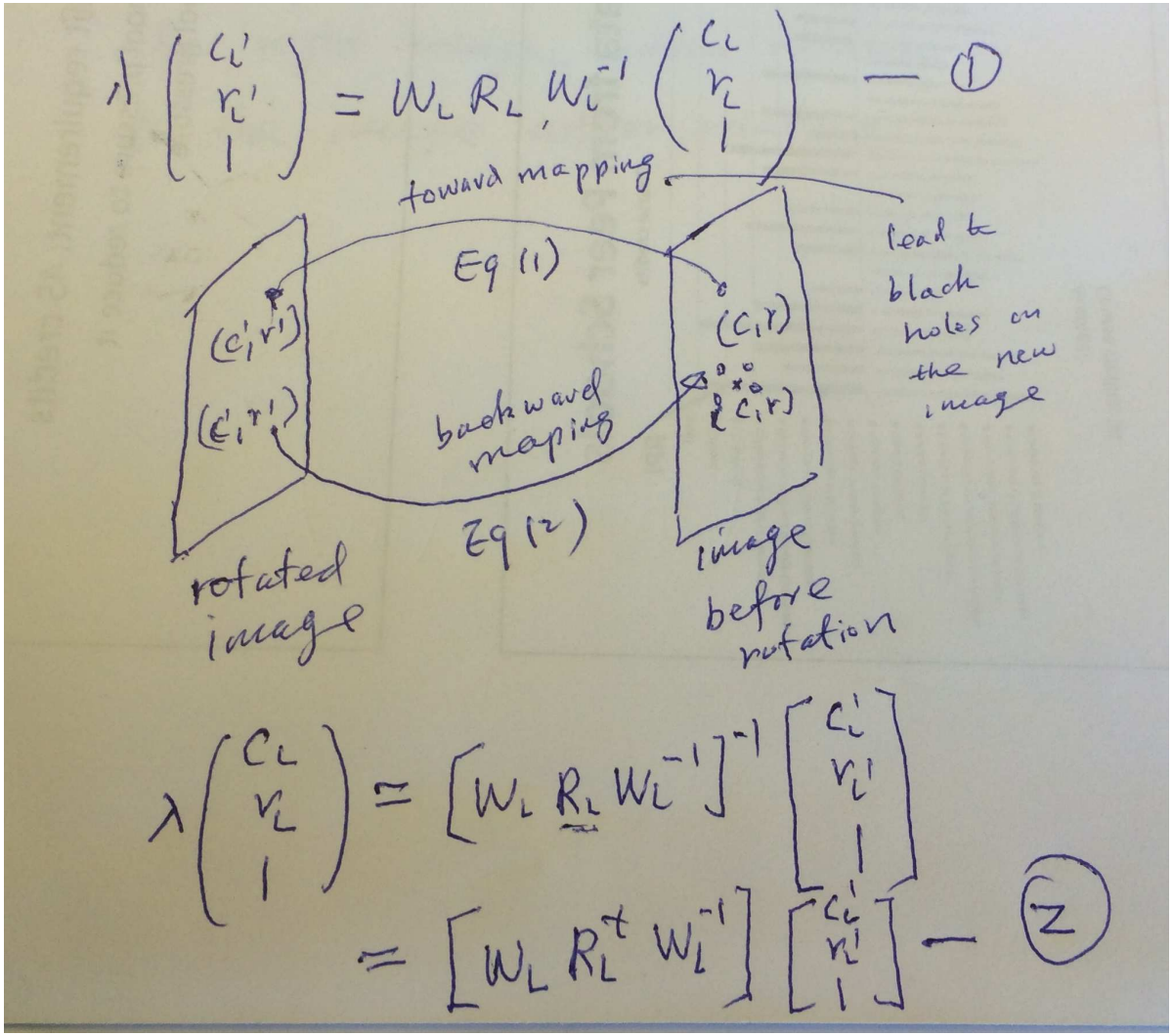
While creating the rectified image from the original image, instead of mapping from (c, r) to (c', r'), we can map from (c', r') to (c, r) to avoid holes in the rectified image.

This can be done as follows

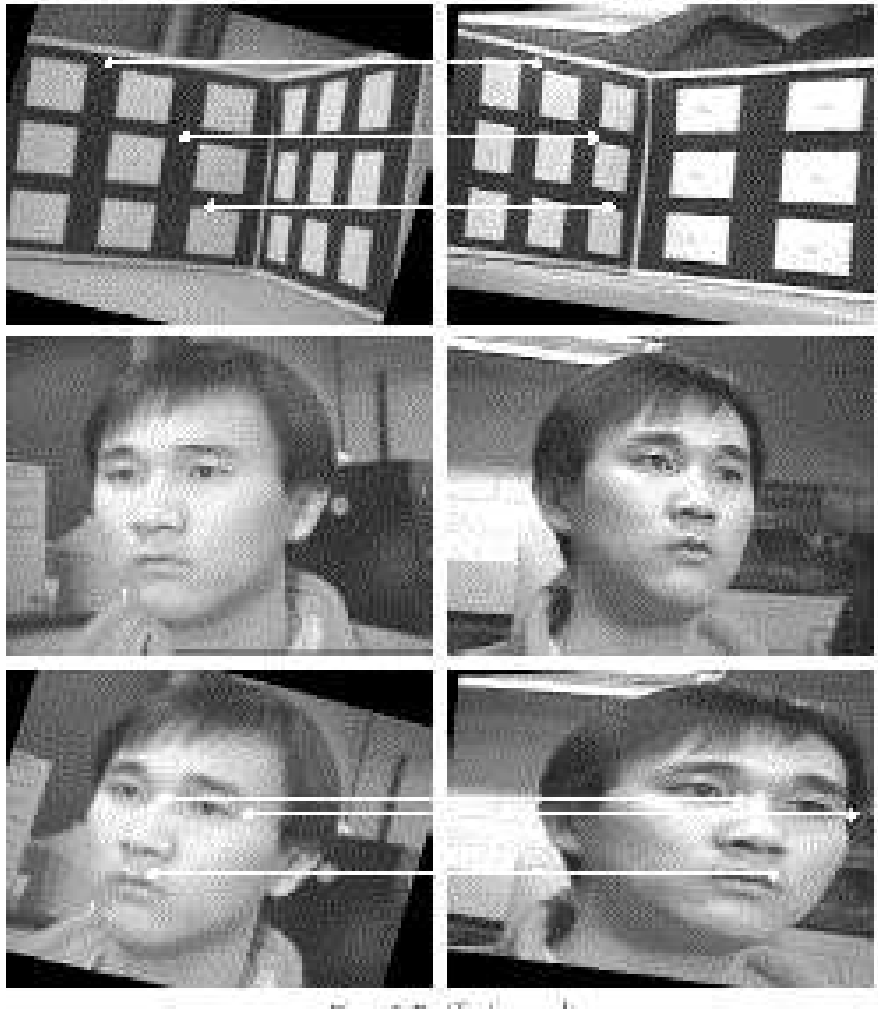
For each pixel (c', r') in the rectified image, we can use equation 22 to identify an image point (c, r) in the original image and transfer the intensity of (c, r) to that of (c', r'). If (c, r) is located between pixels, we can use the intensities of its four neighbors to infer its intensity through a linear interpolation.

Additional information on backward mapping can be found in page 161 of Trucco's book

Backward Mapping



Examples of Rectification



Techniques for establishing correspondence

- Correlation methods (dense matching)
- Feature-based sparse matching
- Matching constraints
- Hypothesis generation and verification

For all above methods, we assume we are dealing with rectified images and that match takes place along the same row with a disparity (δd) range. The disparity range can be determined by the max and min z distance to the camera, using the fact

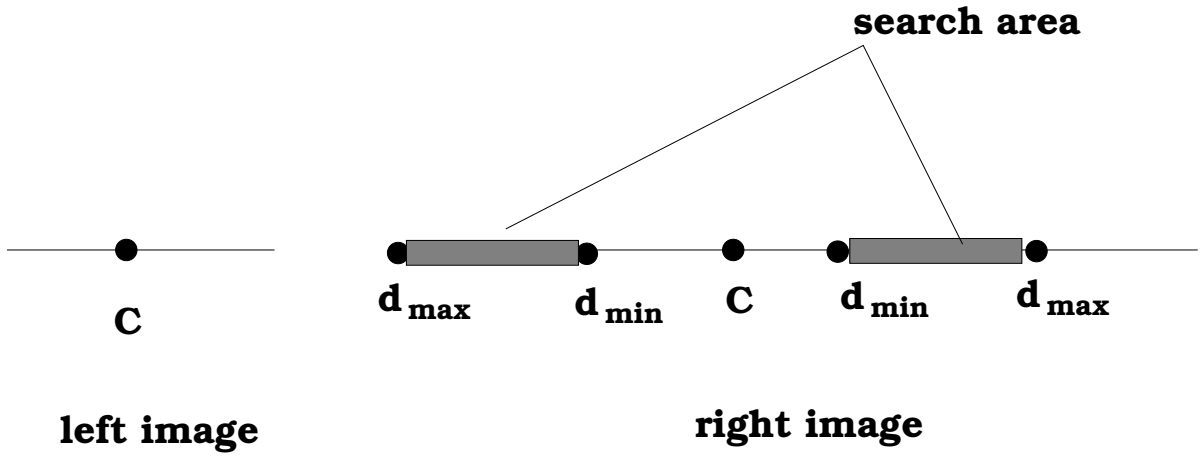
$$\delta d = \frac{fb}{z}$$

Hence, the disparity range for a point (c, r) is

$[c \pm \delta d_{min}, c \pm \delta d_{max}]$, where

$$\delta d_{min} = \frac{fb}{z_{max}}$$

$$\delta d_{max} = \frac{fb}{z_{min}}$$



Correlation methods

The principle of the correlation-based methods is to find two corresponding points based on the intensity distributions of their neighborhoods. The similarity between two neighborhoods is measured by their cross-correlation.

The underlying assumptions include: 1) corresponding image regions look similar, 2) pointed and distant or single light source, 3) corresponding points are visible from both viewpoints.

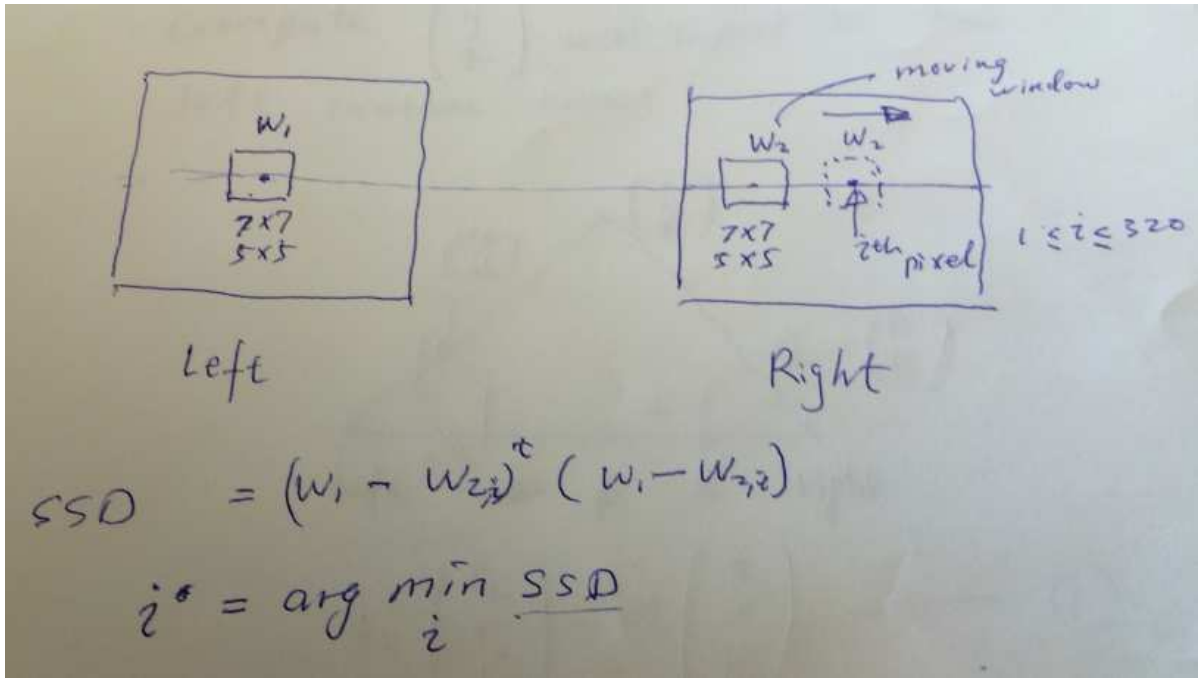
Correlation methods (cont'd)

Given an image point on the left image, the task is to locate another point in a specified region on the right image that is maximally correlated with the one on the left.

Let W_1 and W_2 be the vectors representing elements in window 1 and window 2, their correlations can be computed using two matching metrics:

- Sum of Squared Differences (SSD), which computes the squared difference between the corresponding elements in W_1 and W_2 and obtain their sum, and use the sum to represent the correlation. SSD is defined as follows

$$SSD(c) = (W_1 - W_2(c))^T (W_1 - W_2(c)) \downarrow$$

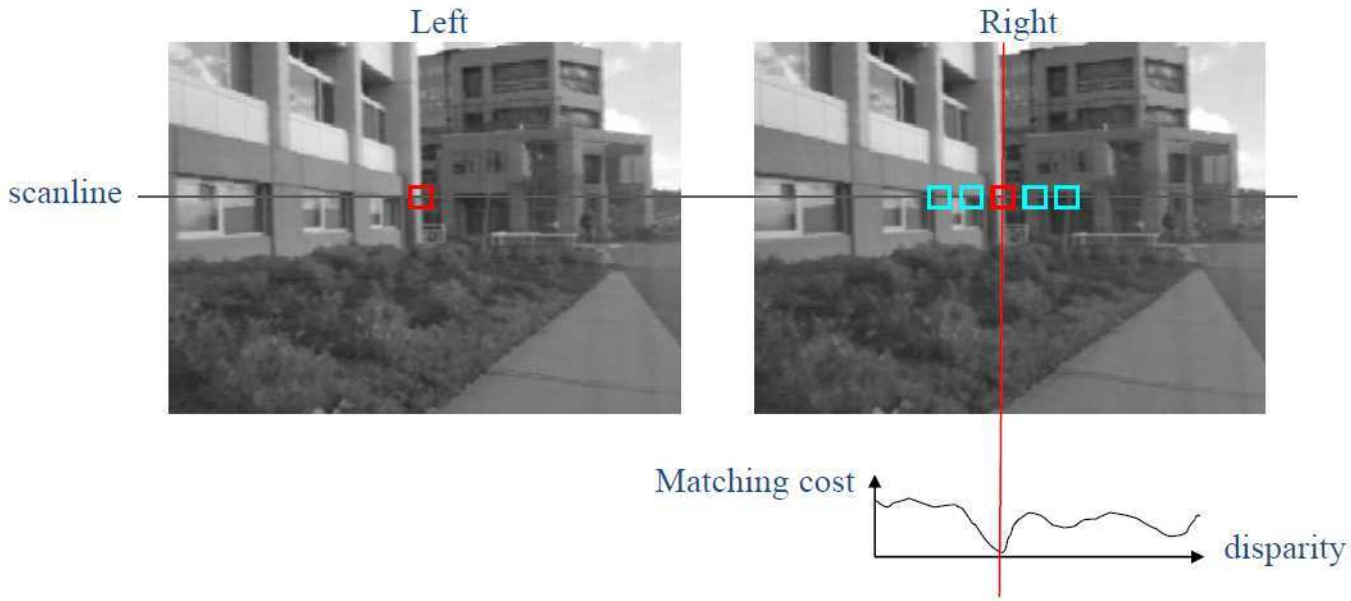


- Normalized cross-correlation $C_{12} = \frac{\sigma_{12}^2}{\sigma_1 \sigma_2} \uparrow$.

  The window size varies, typically 5×5 or 7×7.

Example of Correlation-based Matching)

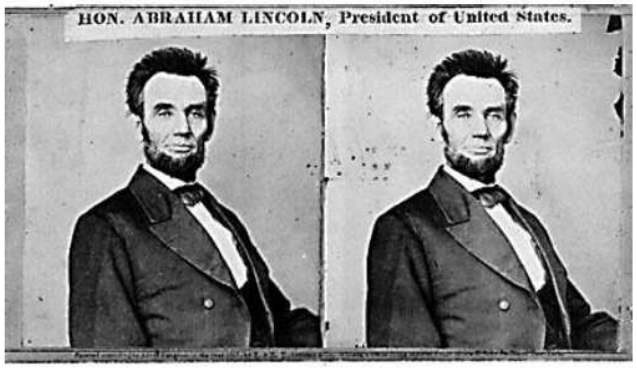
Correspondence search



- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation

Courtesy of Derek Hoiem, University of Illinois

Failures of correspondence search



Textureless surfaces

Occlusions, repetition

Non-Lambertian surfaces, specularities

Courtesy of Derek Hoiem, University of Illinois

Correlation methods (cot'd)

Issues in using correlation based method

- Different methods may be used to compute the correlations. Two measures (cross correlation and sum of squared differences (ssd)) to compute correlation are introduced in Trucco's book.
- The correlation window size is usually chosen differently, depending on images. Some algorithms determine the window size automatically and adapt to different parts of the image.
- The search region also varies from algorithms. If the object is far from the camera, small disparity is expected and we can then search in a small neighborhood around the point with the same as the left image point. If we are dealing with a rectified image, we can only search along the same row on the

right image. Many techniques assume $\delta d_{min} = 0$ and the window size is set to $[c - \delta d_{max}, c + \delta d_{max}]$. But in practice, due to errors with image, it is often necessary to search the **entire row** as well as the the **neighboring rows**.

- To reduce the impact of noise, this technique is often preceded by a smoothing operation.
- Correlation methods need textured images to work well.
- To account for different illuminations, image intensity is usually normalized by subtracting the mean (remove the brightness) and divided by the standard deviation (removes contrast).
- To impose the local smoothness constraint on the disparity map, constraints often employed to encourage small disparity (depth) variation for neighboring pixels. Alternatively, we can

also perform a post smoothing processing such as median fitering to smooth the disparity map.

- Global methods are often used to perform matching for all pixels simultaneously using models such as the Markov Random Field as a prior to explicitly impose surface the smoothness constraint by solving an energy minimization problem. See section 12.5 of [17].
- With the availability of training datasets such as KITTI [4], deep learning methods such as [10] have achieved better performance.

Sparse Matching

Instead of matching every pixel, sparse matching methods restrict the search for correspondences to a sparse set of geometric entities. Typical examples of geometric entities include edge points, corner points, and lines. Each entity is described by some feature descriptors . For example, for edges, feature descriptors may include edge direction and strength. For dominant points, aggregated feature descriptors such as histograms of gradients (HOGs) are often used.

Using the feature descriptors, points are then matched based on the closeness between their feature descriptors. The assumption is that feature descriptor values remain unchanged across images for the corresponding points.

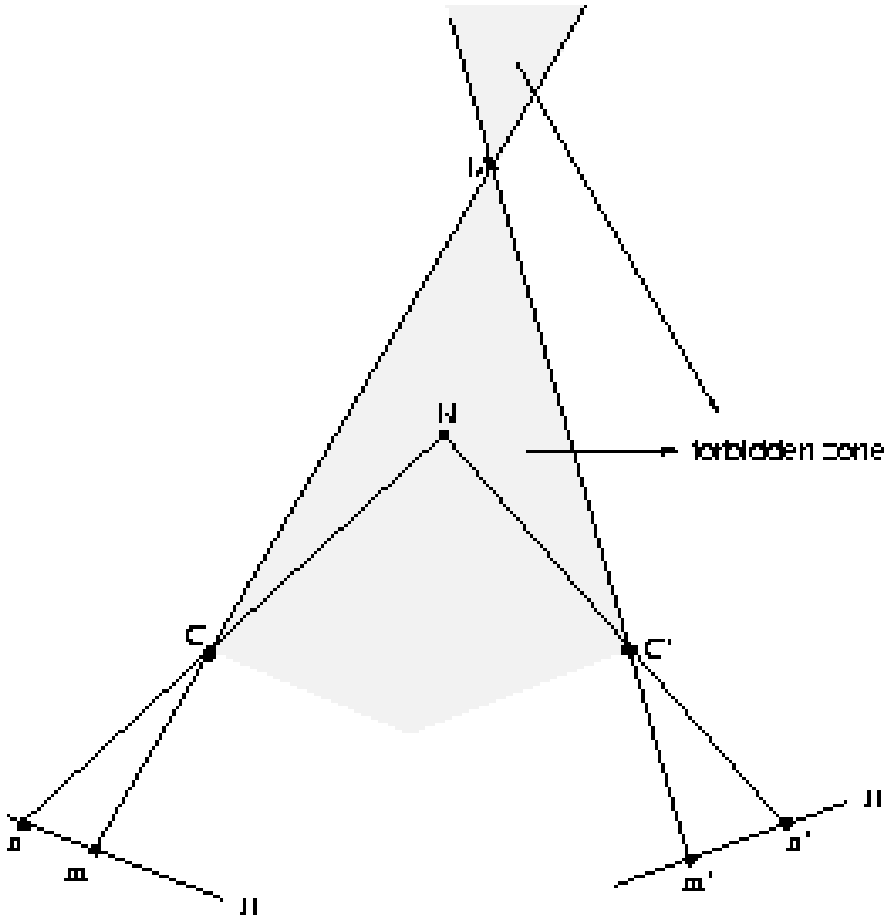
Matching Constraints

- Smooth constraint (surface locally smooth z=f(d))
  - 0 order disparity constraint: two neighboring image points should have close disparity.
  - first order disparity constraint (disparity gradient): disparity gradient is upper-bounded.

Matching Constraints (cont'd)

- Physical and geometric constraints:
  - Physical constraints (monotonic ordering): the 3D point generated by two corresponding image points should be: a) located on the **surface of the same object**; b) visible from both views; c) **not located inside** the object; d) unique (one to one matching); d) spatial ordering is preserved.

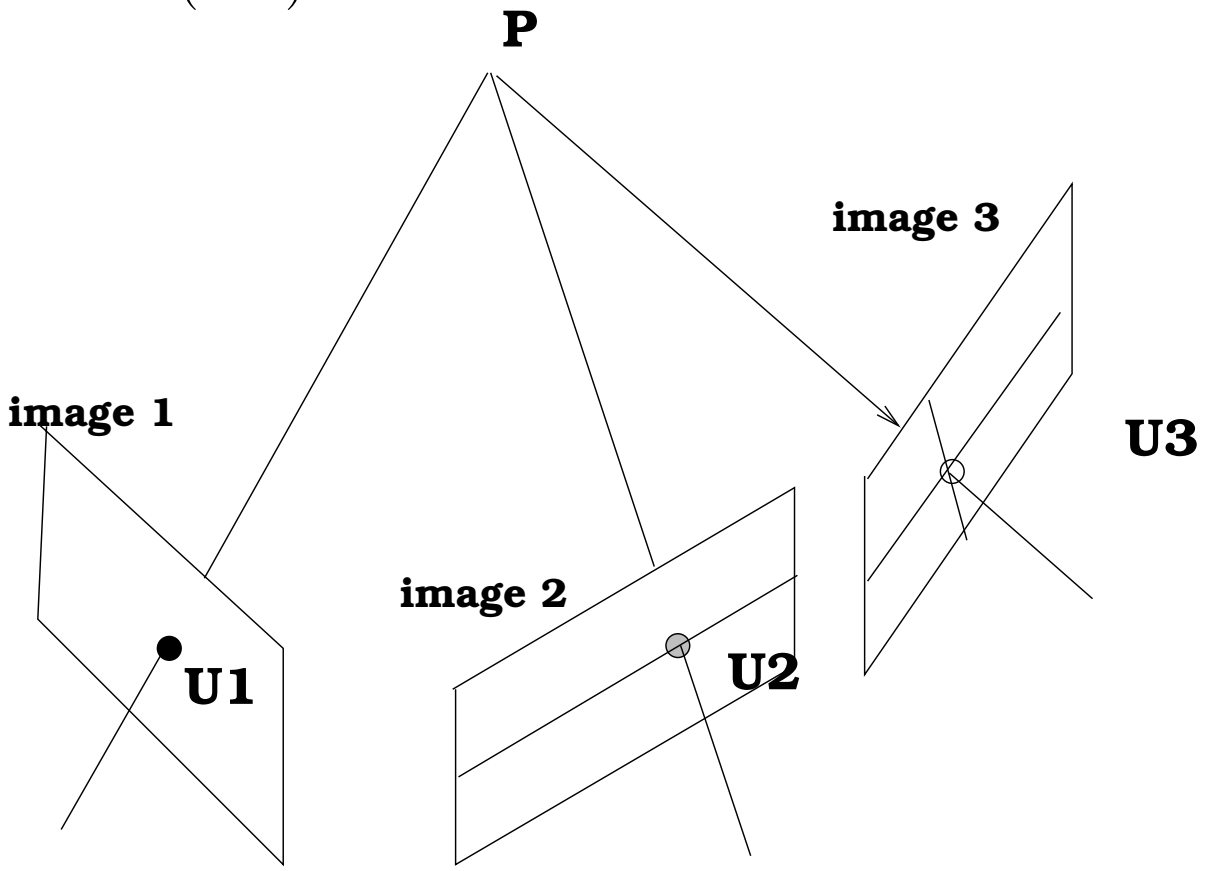
The spatial ordering constraint fails at the *forbidden zone*.



where N is located in the forbidden zone of point M.

Matching Constraints (cont'd)

- Multiple views (> 2)

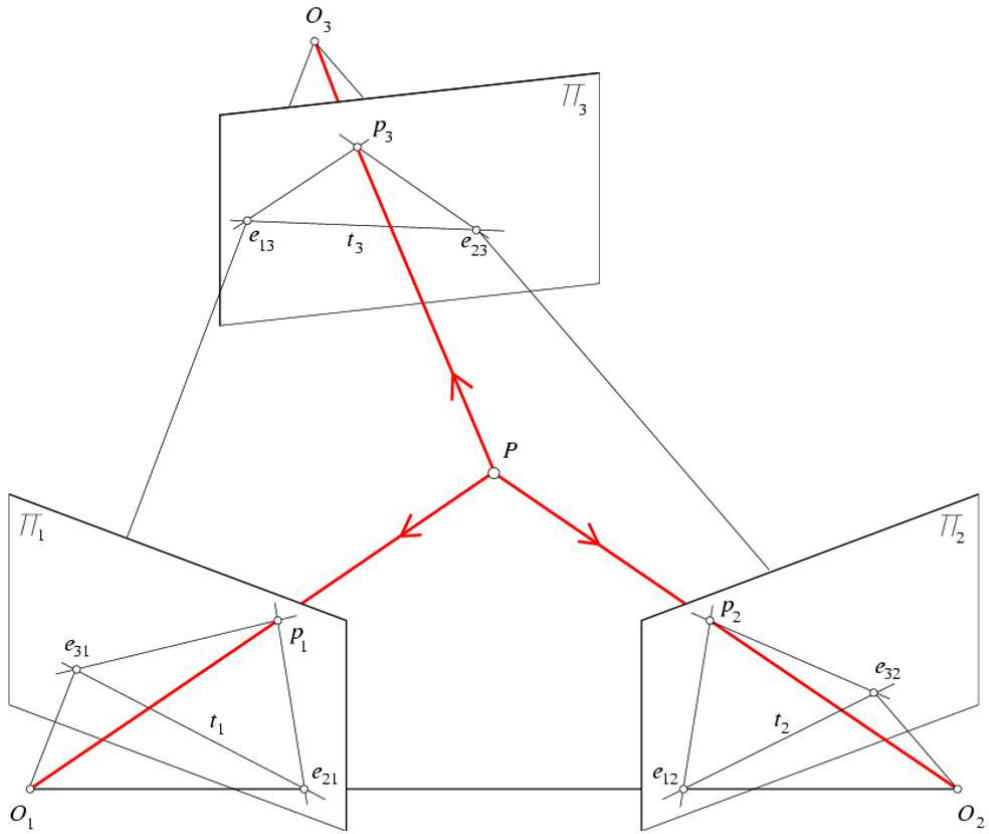


  - predict point in third image (0 order constraint).

- predict tangent in third image (first order constraint)
- predict curvature in third image (second order constraint)

  Check the point in the third image to ensure its geometric properties match the predicted ones.

Like fundamental matrix for two views, **trifocal tensor** for three views. It is a $3 \times 3 \times 3$ tensor that constrain the geometric relationships among three views;
it contains 3 fundamental equations, one for each pair of views.



$p_1^T F_{12} p_2 = 0$, $p_1^T F_{13} p_3 = 0$, $p_2^T F_{23} p_3 = 0$. Note they are not independent. See chapters 14 and 15 of Hartley's book.

Matching Constraints (cont'd)

- Locally known surface form (e.g. locally planar or sphere). Given the plane parameters, we can then uniquely determine the corresponding points.

Prediction and verification

- Extract primitive features from the images
- Assign initial correspondences between the points in two images using the extracted features
- Estimate the 3D pose and model based on the initial point correspondences
- Backproject the estimated 3D model onto the image and measure the discrepancies between the projected image and the original image.
- Iteratively refine the current matches to minimize the projection errors subject to the local smoothness constraint, using a relaxation or an optimization technique.

Reconstruction

- Known camera parameters-full reconstruction
  - Linear technique and non-linear techniques
- Unknown camera parameters
  - Unknown extrinsic parameters but known intrinsic parameters-Euclidean reconstruction, i.e.,the reconstruction is up to an euclidian transformation.
    - recover R and T and then perform full reconstruction
  - both intrinsic and extrinsic parameters are unknown-projective reconstruction, i.e., the reconstruction is up to a projective transformation.
    - Simultaneous estimation of camera parameters and 3D reconstruction

Reconstruction: known camera parameters

This means we know R , T, W_l and W_r, where R and T specify the relative orientation between two cameras. Such reconstruction is called **full reconstruction**. Assume the object frame coincide with the left camera frame and let the two image points be (c_l, r_l) and (c_r, r_r), algebraically, we can solve the 3D coordinates (x_l, y_l, z_l) relative to the left camera frame as follows:

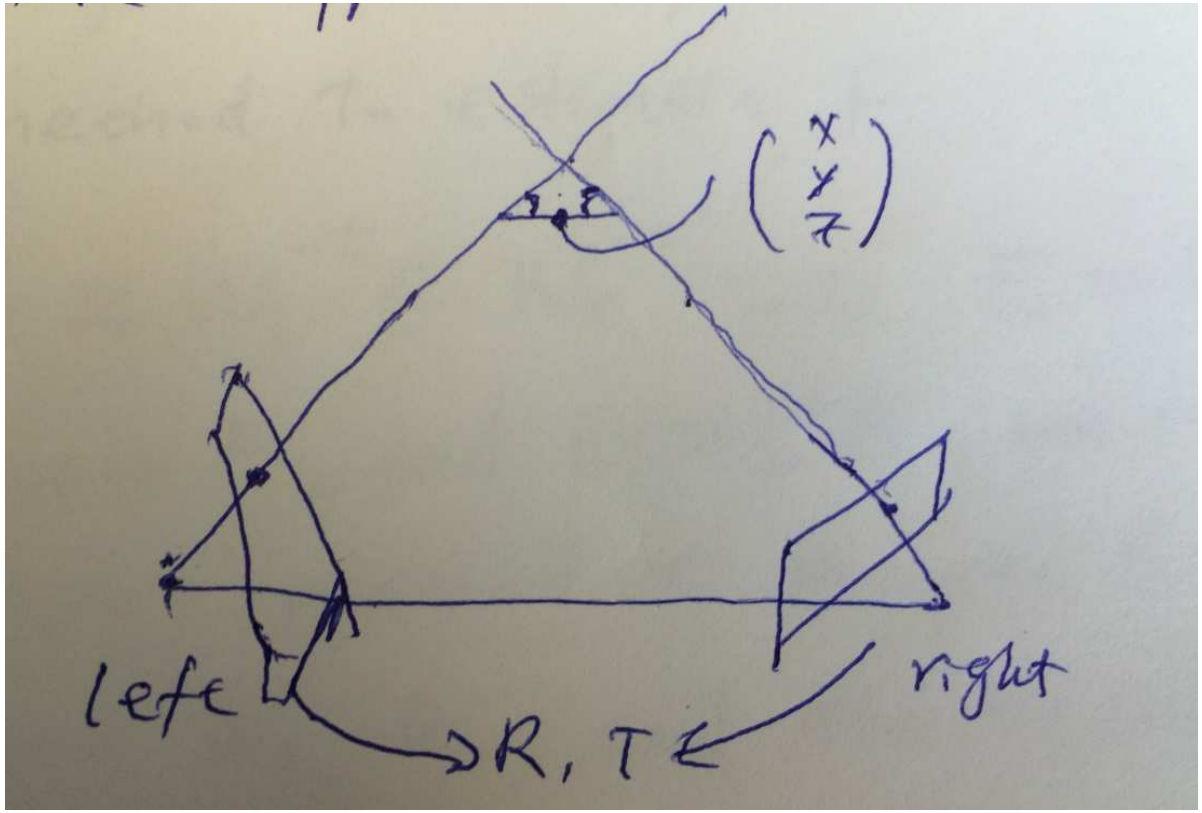
$$\lambda_l \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} = W_l \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix}$$

$$\lambda_r \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix} = W_r \begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = W_r(R^T \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} - R^T T)$$

given a total of 5 unknowns $(x_l, y_l, z_l, \lambda_l, \lambda_r)$ and 6 linear equations of the unknown, we can setup a least-squares linear system to solve for the five unknowns.

Reconstruction: known camera parameters (cont'd)

Alternatively, we can solve (x_l, y_l, z_l) geometrically. See the figure below for the geometric approach solution. Due to image noise, the two rays (representing the two image points) may not intersect. The goal is to identify the line segment that interests with and orthogonal to the two rays. The center of the line segment is the 3D coordinates we need to compute.



The left line equation: $(x_l, y_l, z_l)^T = \lambda_l W_l^{-1}(c_l, r_l, 1)^T$ and the right line equation $(x_r, y_r, z_r)^T = \lambda_l W_r^{-1}(c_r, r_r, 1)^T$. Relative to the left camera frame, the right line equation is changed to $(x_l, y_l, z_l)^T = R\lambda_l W_r^{-1}(c_r, r_r, 1)^T + T$. Their intersection can be

found by combining the first and third equations by solving $\lambda = \lambda_l = \lambda_r$.

Reconstruction: known camera parameters (cont'd)

The linear solutions, while simple, often do not produce accurate solution. An non-linear solution can be used to improve the linear estimates. The non-linear solution can be formulated as minimizing

$$(P_l(X) - \begin{pmatrix} c_l \\ r_l \end{pmatrix})^T (P_l(X) - \begin{pmatrix} c_l \\ r_l \end{pmatrix}) +$$

$$(P_r(X) - \begin{pmatrix} c_r \\ r_r \end{pmatrix})^T (P_r(X) - \begin{pmatrix} c_r \\ r_r \end{pmatrix})$$

where $P_l(X)$ and $P_r(X)$ are the projected image points on the left and the right image for 3D point X, respectively. (c_l, r_l) and

(c_r, r_r) are the corresponding observed image points.

Reconstruction: unknown extrinsic parameters

Without knowing the relative relations between the two cameras, the reconstruction is up to **Euclidean transformation** (rotation, translation, and reflection), i.e., the reconstruction can preserve the object shape but not the scale, position and orientation of the reconstructed. Such reconstruction is called **Metric or Euclidean reconstruction**. Euclidean transformation is a special kind of affine transformation, where the shape, distances, and angles are invariant before and after transformation.

Euclidean reconstruction (i.e., recovering only object shape) can be obtained theoretically by performing a full 3D reconstruction with an arbitrary R and T or performing a simultaneous estimation of the extrinsic camera parameters (R and T) and 3D reconstruction.

Alternatively, to perform a full reconstruction, we can perform a stereo calibration to obtain the fundamental matrix F, based on which we can obtain the essential matrix E. Given E, we can receiver R and T. Specifically, given 8 or more point correspondences on the left and right images, we can use the 8-point method to obtain the fundamental matrix F up to a scale factor. Given F and the intrinsic camera parameters, we can then obtain the essential matrix E, from which we can then recover R and T using the two methods below.

Computing R and T from E: Method 1

Given E, we can solve for (R, T) up to four solutions using two methods. Method 1 is based on Horn [9]. See pages 164 and 165 of Trucco's book or the paper for detail. The four (R, T) solutions lead to four reconstructions, only one of them is geometrically consistent.

Computing R and T (cont'd)

Following Horn's method [9], let S by the skew matrix computed from T. Given $E = S^T R$, the four solutions for T are

$$TT^T = \frac{1}{2} Trace(EE^T) I - EE^T$$

[a] where I is the 3×3 identity matrix. The two solutions for the orientation can be found using

$$R = \frac{Cofactor(E)^T - SE}{T^T T}$$

where $Cofactor(E)$ is the co-factor matrix of E, i.e., $Cofactor(E)_{ij} = (-1)^{i+j} D_{ij}$. D_{ij} is the determinant of the submatrix of E, formed by omitting ith row and jth column.

[a] Note $EE^T = S^T R R^T S = S^T S = I(t_x^2 + t_y^2 + t_z^2) - TT^T$

Computing R and T from E: Method 2

As $T \times S = 0$, we have $T \cdot E = 0$, which leads to $E^T T = 0$

Hence, T is solved up to a scale factor as the null vector of E^T, i.e., the last column vector of V, which results from SVD of E^T, i.e., $E^T = UDV^T$. We can plug the solution to $EE^T = S^T S$ to solve the scale factor as $s = \frac{trace(EE^T)}{2trace(TT^T)}$.

Similarly, R can be solved directly from E, i.e., $R = U'W^{-1}V'^T$, where U' and V' correspond to the SVD matrices of E, and

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Refer to https://en.wikipedia.org/wiki/Essential_matrix for more details.

Reconstruction: unknown camera parameters

Here we are interested in reconstruction with only matched image points and in the absence of any information on both the intrinsic and extrinsic parameters. The reconstruction is called **projective reconstruction**, and it is unique up to an unknown projective transformation matrix H, where H is a function of interior and exterior parameters of the stereo system. Such a reconstruction neither preserves the object shape nor its position and orientation.

Projective reconstruction can be obtained by performing simultaneous estimation of the camera parameters and the 3D reconstruction. Hartley (chapter 9.2, p248) [a] stated the projective reconstruction upgrades to affine reconstruction (up to an affine transformation) if two cameras relate via pure translation.

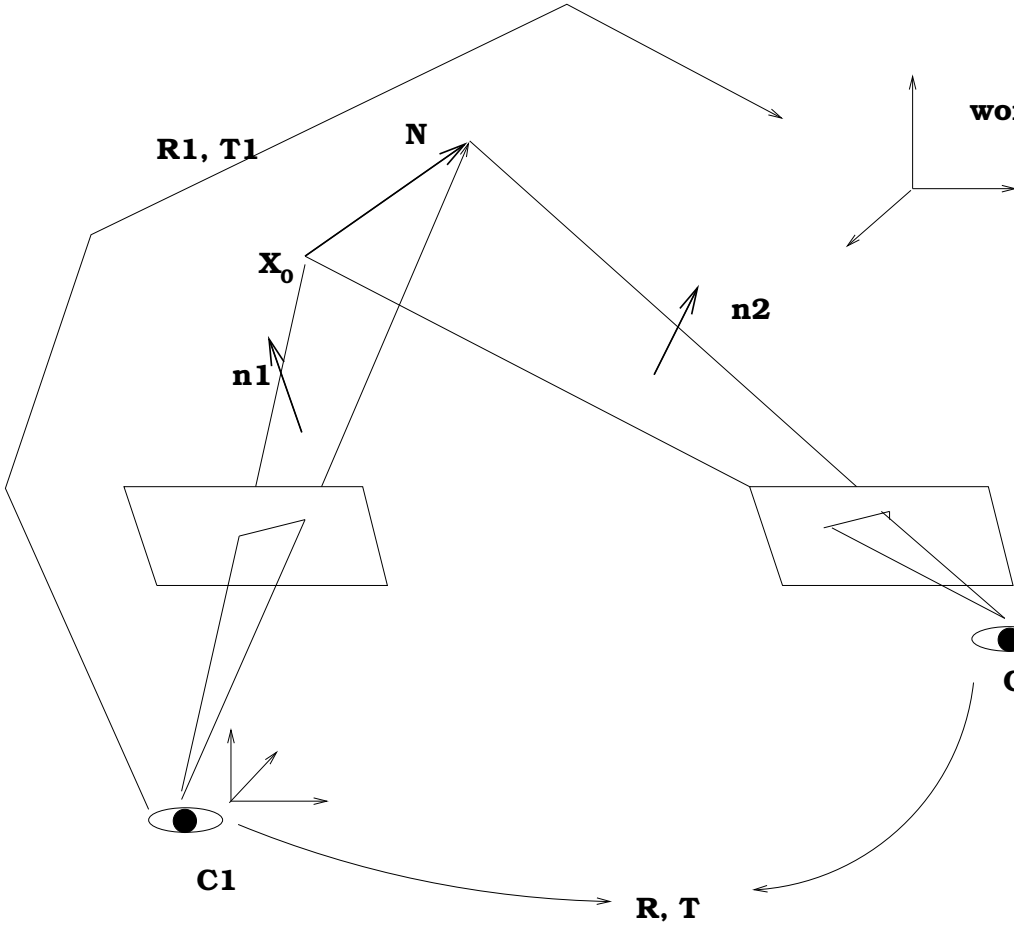
[a] Multiple View Geometry in computer vision, Richard Hartley and Andrew Zisserman

Alternatively, if we are given a minimum of three images from the same camera, then we can use self-calibration techniques to recover the intrinsic camera parameters. We can then obtain the relative orientation from the essential matrix using Horn's method, and eventually obtain the full 3D reconstruction.

Stereo from Lines and Conics

- Stereo from lines
- Stereo from conic curves

Stereo from Lines



Line Matching

If two line segments match, then any two points from the first line segment must match two points on the second lines. The two points on the second line can be found from the intersections between the second line segment and the epipolar lines .

Stereo from Lines (cont'd)

Assume the object frame coincides with the left camera frame and R and T capture the relative orientation of the right camera frame w.r. t. the left camera from. Let the 3D line be parameterized by it's orientation N and a point on the line X_0. Given the corresponding 2D image line on the left row-column frame as $\alpha_1 c_1 + \beta_1 r_1 + \gamma_1 = 0$, the normal of the left backprojection plane is $n_1 = W_1^T \begin{pmatrix} \alpha_1 \\ \beta_1 \\ \gamma_1 \end{pmatrix}$. Simimarly, given the corresponding 2D image line for on the right row-column frame as $\alpha_2 c_2 + \beta_2 r_2 + \gamma_2 = 0$, the normal for the right backrprojection

plane is $n_2 = W_2^T \begin{pmatrix} \alpha_2 \\ \beta_2 \\ \gamma_2 \end{pmatrix}$.

We hence have

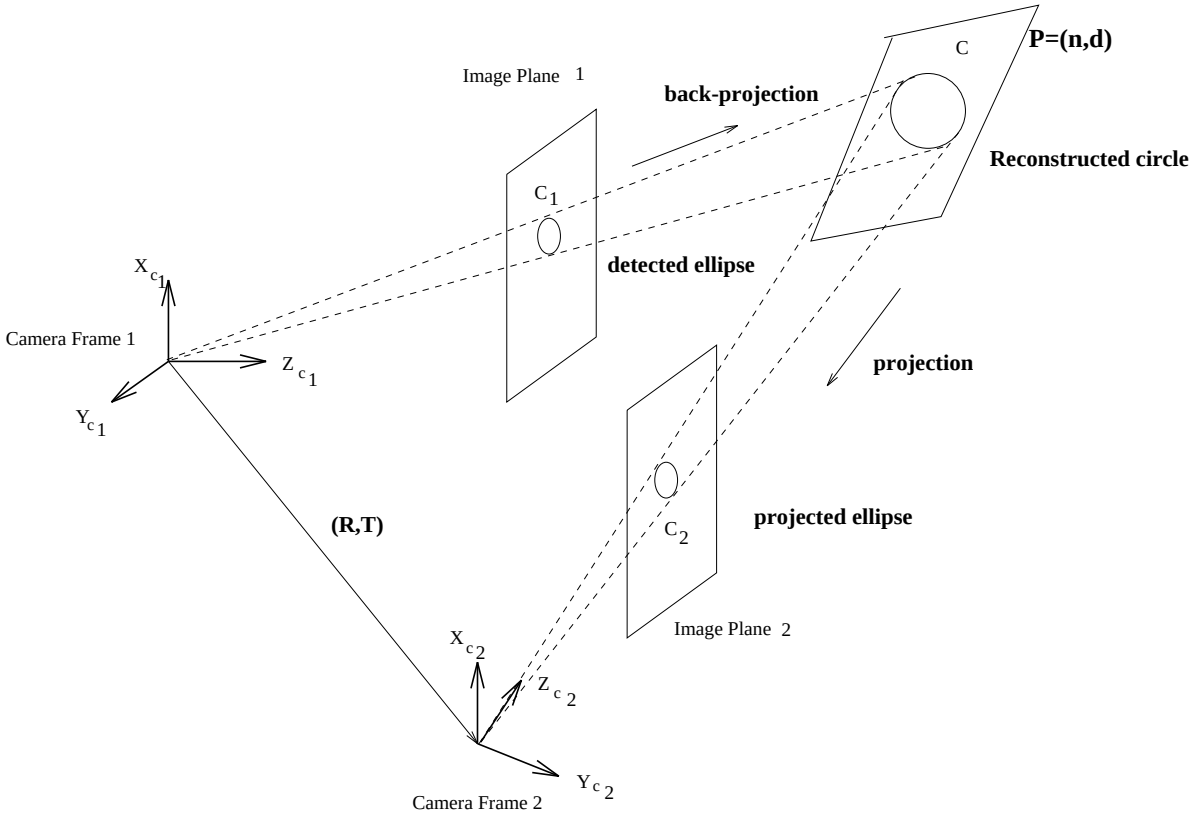
$$\begin{aligned} n_1^T N &= 0 \\ n_2^T R^T N &= 0 \\ n_1^T X_0 &= 0 \\ n_2^T (R^T X_0 - R^T T) &= 0 \end{aligned}$$

The first two equations allow to solve for N. Specifically, based on the first two equations, given n_1 and n_2, the line orientation N w.r.t the left camera can be computed as $N = n_1 \times R n_2$.

The last two equations allow to solve for X_0 up to a scale factor.

Stereo from Conic Curves [16]

- Correspondences between ellipses can be established using epipolar constraint. The corresponding points on the right image are the intersections between the epipolar lines and the ellipse.



- Reconstruction

  Let A and B be the 3×3 symmetric matrices that respectively specify the left and right image and left image ellipses, we hence have

$$\begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix}^T A \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} = 0.$$

$$\begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix}^T B \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix} = 0.$$

Substituting $\lambda_l \begin{pmatrix} c_l \\ r_l \\ 1 \end{pmatrix} = W_l \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix}$ and

$\lambda_r \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix} = W_r \begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix}$ into the ellipse equations above yields the equations for the two cones in the left and right camera frames as shown in the figure

$$\begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix}^T W_l^T A W_l \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} = 0.$$

and

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix}^T W_r^T B W_r \begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = 0.$$

Substituting $\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = R^T \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} - R^T T$ to the right cone equation yields the right cone expressed in the left camera frame

$$[R^T \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} - R^T T]^T W_r^T B W_r [R^T \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} - R^T T] = 0.$$

The two cones are now expressed in left camera frames. As shown in the figure, their surface intersection produces a planar 3D conic curve. Assume the equation of the plane, where the intersecting conic curve is located on, is $ax_l + by_l + cz_l + d$=0 w.r.t the left camera frame. Plugging $z_l = -\frac{ax_l+by_l+d}{c}$ into the two cone equations above gives us

two equations for the conic curve in terms of ratios of the plane parameters $(a/c, b/c, d/c)$. Solve these plane parameter ratios (multiple solutions may exist) by ensuring the two conic equations are identical (with the same coefficients) as they represent the same conic curve. This gives us the equations of the 3D conic curve. Finally, when the 3D conic becomes a circle, one image from one camera is sufficient to recover the orientation of the 3D circle if given the radius of the 3D circle (prove this using above equations).

Limitations with Passive Stereo

A key limitation with the passive stereo approach is the accuracy with the reconstructed 3D points. This lack of accuracy may be due to

- Positional errors (e.g., quantization error) with image coordinates.
- **Inaccuracy in point matching** (e.g. mismatch),
- Errors with camera parameters (like focal length, baseline distance, etc..)

Bayesian Triangulation

Given observed image points $\hat{U}_1$ and $\hat{U}_2$ and their covariance matrices Σ_{U_1} and Σ_{U_2}, the camera parameters $\hat{\Theta}_1$, and $\hat{\Theta}_2$, and their covariance matrices Σ_{Θ_1}, and Σ_{Θ_2}, estimate the corresponding 3D point $\hat{X}$ and it's covariance matrix Σ_X by maximizing

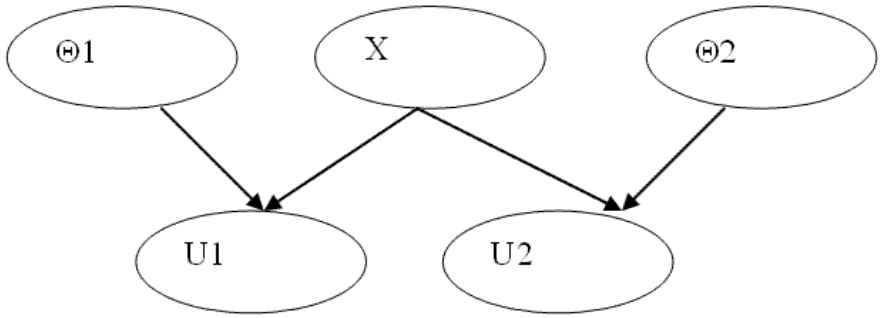
$$p(\hat{X}|\hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2)$$

Bayesian Triangulation (cont'd)

Since $p(\hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2)$ is merely a normalizing constant, maximizing $p(\hat{X}|\hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2)$ is the same as maximizing $p(\hat{X}, \hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2)$.

Bayesian Triangulation (cont'd)

Assuming the relations between $\hat{U}_i$, $\hat{X}$, and $\hat{\Theta}_i$ can be modelled by the following Bayesian Network



We then have

$$
\begin{aligned}
p(\hat{X}|\hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2) &= \frac{P(\hat{X}, \hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2)}{P(\hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2)} \\
&= \alpha P(\hat{X}, \hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2) \\
= \quad & P(\hat{X}) P(\hat{\Theta}_1) P(\hat{\Theta}_2) P(\hat{U}_1|X, \hat{\Theta}_1) P(\hat{U}_2|X, \hat{\Theta}_2) \\
= \quad & \alpha P(\hat{X}) P(\hat{\Theta}_1) P(\hat{U}_1|X, \hat{\Theta}_1) P(\Theta_2) P(\hat{U}_2|X, \hat{\Theta}_2) \\
= \quad & \alpha P(\hat{X}) \prod_{i=1}^{2} P(\hat{\Theta}_i) P(\hat{U}_i|X, \hat{\Theta}_i) \qquad (23)
\end{aligned}
$$

Bayesian Triangulation (cont'd)

Since $\hat{\Theta}_i \sim N(\Theta_i, \Sigma_{\Theta_i})$,

$$P(\hat{\Theta}_i) = \frac{\exp(-\frac{1}{2}(\hat{\Theta}_i - \Theta_i)^T \Sigma_{\Theta_i}^{-1}(\hat{\Theta}_i - \Theta_i))}{2\pi|\Sigma_{\Theta_i}|^{\frac{1}{2}}}$$

Bayesian Triangulation (cont'd)

Assume given X and $\hat{\Theta}_i$, $\hat{U}_i \sim N(U_i, \Sigma_i)$.

Hence,

$$P(\hat{U}_i|X, \hat{\Theta}_i) = \frac{\exp(-\frac{1}{2}(\hat{U}_i - U_i)^T \Sigma_i^{-1}(\hat{U}_i - U_i))}{2\pi|\Sigma_i|^{\frac{1}{2}}}$$

We need Σ_i and U_i.

Bayesian Triangulation (cont'd)

By perspective projection, we have $\hat{U}_i = M(\hat{\Theta}_i, X)$, where M is the projection matrix. Assume the noise with camera parameters are small, by a first order Taylor series approximation of $M(\hat{\Theta}_i, X)$, we have

$$\begin{aligned} \hat{U}_i &= M(\hat{\Theta}_i, X) \\ &= M(\Theta, X) + \frac{\partial M}{\partial \Theta_i} \Delta\Theta_i \end{aligned}$$

Bayesian Triangulation (cont'd)

Since $M(\Theta_i, X) \sim N(U_i, \Sigma_{U_i})$ and $\Delta\Theta_i \sim N(0, \Sigma_{\Theta_i})$, we have

$$\Sigma_i = \Sigma_{\Theta_i} + (\frac{\partial M}{\partial \Theta_i})\Sigma_{\Theta_i}(\frac{\partial M}{\partial \Theta_i})^T$$

Since U_i represents the ideal yet unobserved image projection given Θ_i and X. Assume small perturbations with image and camera, U_i may be approximated by $M(\hat{\Theta}_i, X)$.

Bayesian Triangulation (cont'd)

To solve for $p(\hat{X}, \hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2)$, we still need to compute $p(\hat{X})$.

Assume $\hat{X}$ is uniformly distributed, then maximizing $p(\hat{X}, \hat{U}_1, \hat{U}_2, \hat{\Theta}_1, \hat{\Theta}_2)$ is equivalent to maximizing $E = \prod_{i=1}^{2} p(\hat{U}_i|\hat{\Theta}_i, X)p(\hat{\Theta}_i)$.

Bayesian Triangulation (cont'd)

$$
\begin{aligned}
\log E &= \sum_{i=1}^{2} \{\log p(\hat{U}_i|\hat{\Theta}_i, X) + \log p(\hat{\Theta}_i)\} \\
&= \sum_{i=1}^{2} \{-\frac{1}{2}(\hat{U}_i - U_i)^T \Sigma_i^{-1} (\hat{U}_i - U_i) - \\
&\quad \log(2\pi) - \frac{1}{2}\log|\Sigma_i| - \frac{1}{2}(\hat{\Theta}_i - \Theta_i)^T \Sigma_{\Theta_i}^{-1} (\hat{\Theta}_i - \Theta_i) - \\
&\quad \log(2\pi) - \frac{1}{2}\log|\Sigma_{\Theta_i}|\}
\end{aligned}
$$

Bayesian Triangulation (cont'd)

Maximizing $\log E$ is equivalent to minimizing $-\log E$. Removing the constant terms in $-\log E$ and terms independent of $\hat{X}$ yields

$$\epsilon^2 = \sum_{i=1}^{2} (\hat{U}_i - M(\hat{X}, \Theta_i))^T \Sigma_i^{-1} (\hat{U}_i - M(\hat{X}, \Theta_i)) + \log |\Sigma_i|$$

where

$$\Sigma_i = \Sigma_{U_i} + [\frac{\partial M}{\partial \Theta_i}(\hat{X}, \Theta_i)] \Sigma_{\Theta_i} [\frac{\partial M}{\partial \Theta_i}(\hat{X}, \Theta_i)]^T$$

$$\hat{X}^* = \arg\min_{\hat{X}} \epsilon^2$$

Deep Learning Based Passive Stereo

Deep learning based methods have been applied to passive stereo problems, including feature learning for point matching and depth estimation. A recent survey of deep learning methods can be found in [11, 5, 13].

Deep Learning for Depth Estimation

Deep learning methods for depth estimation can be grouped into two categories: matching feature learning and end-to-end learning.

Matching feature learning employs deep learning based methods to learn features that best match corresponding patches.

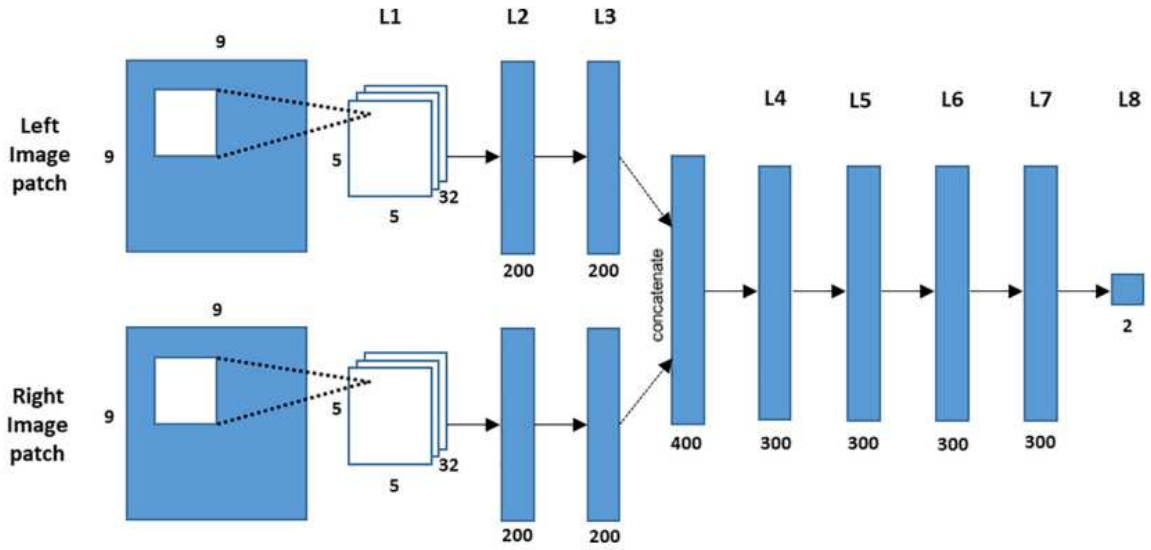


Fig. 5. MC-CNN-acrt network.

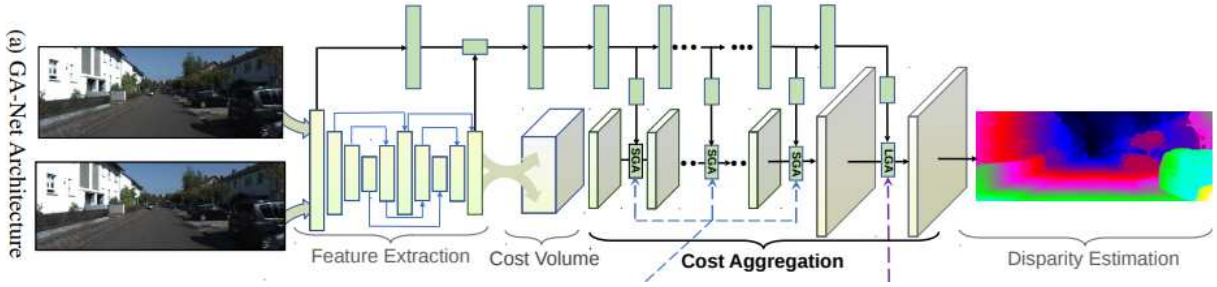
It includes 3 steps. 1) Generate groundtruth dataset. For every reference patch on the left image, positive and negative matching patches from the right image are given; 2) construct the loss function. The loss function includes the main loss term derived from given positive and negative patch pairs as well as additional regularization terms that represent matching constraints,

including epipolar constraint, the smoothness constraint, disparity range constraints, the spatial ordering constraint, the uniqueness constraint, etc.; 3) Train the model.

Once trained, the model can compute the matching cost for each query patch pair and identify the pair with the minimal matching cost, based on which the disparity (depth) can be derived from the matched pairs.

Deep Learning for Depth Estimation (cont'd)

Given a pair of left and right image, the end-to-end learning method [20] below directly outputs the disparity map through regression, without explicitly performing pixel matching. It includes 4 steps: learn features for the entire image, use current features to predict the cost for each pixel, producing the cost volume, aggregate the cost volume to produce the matching map, and derive the disparity map from the matching map. This method, however, requires a lot of training data with groundtruth disparity maps.



Benchmark stereo datasets [17]

Name/URL	Contents/Reference
Middlebury stereo https://vision.middlebury.edu/stereo	33 high-resolution stereo pairs (Scharstein, Hirschmüller *et al.* 2014)
Middlebury multi-view https://vision.middlebury.edu/mview	6 3D objects scanned from 300+ views (Seitz, Curless *et al.* 2006)
EPFL (no longer active)	6 outdoor multi-view sets of images (Strecha, von Hansen *et al.* 2008)
KITTI 2015 http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php	200 train + 200 test stereo pairs (Menze and Geiger 2015)
DTU https://roboimagedata.compute.dtu.dk/?page_id=36	124 toy scenes with 49–64 images each (Jensen, Dahl *et al.* 2014)
Freiburg Scene Flow https://lmb.informatik.uni-freiburg.de/resources/datasets	39k synthetic stereo pairs (Mayer, Ilg *et al.* 2018)
ETH3D https://www.eth3d.net	13 training + 12 test high-res scenes (Schöps, Schönberger *et al.* 2017)
Tanks and Temples https://www.tanksandtemples.org	7 training + 14 test 4K video scenes (Knapitsch, Park *et al.* 2017)
BlendedMVS https://github.com/YoYo000/BlendedMVS	17k MVS images covering 113 scenes (Yao, Luo *et al.* 2020)

Table 12.1 *Widely used stereo datasets and benchmarks.*

Active Stereo Vision

Active stereo vision is a technique for the reconstruction of the three-dimensional description of a scene from images observed from one camera and one light projector.

See the Active Stereo slides at the link below for details.

http://www.ecse.rpi.edu/ qji/CV/ActiveStereo.pdf

Monocular Approach

Problem Statement : *Given a single 2D image of an object, reconstruct the geometry of the visible surface of the object.*

A monocular image alone does not contain sufficient information to uniquely reconstruct 3D information. 3D information, however, can be recovered from monocular images in conjunction with certain visual cues or prior knowledge of certain geometric properties of the object.

Visual Cues

What are the visual cues used by human to infer depth or shape?

- Geometric properties
- Shade
- Distortion
- Vanishing points
- Blurriness

Geometric Properties

- Euclidean distance relationships between geometric entities
- Angular relationships between geometric entities

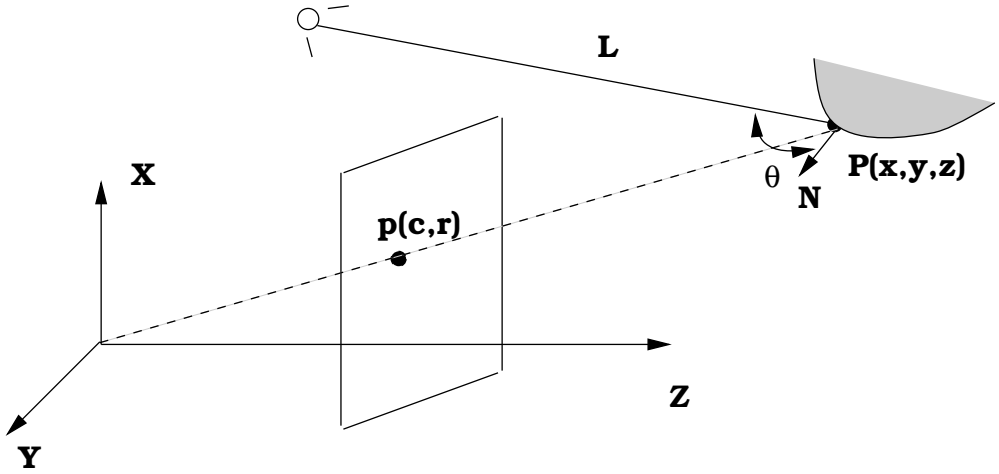
Shape from X

- Shape from shading.
- Shape from texture.
- Shape from geometry.
- Shape from illumination (photometric stereo)
- Shape from (de-)focus.

Shape from Shading

Shape-from-shading technique infers the shape (surface normals) and depth of an image pixel based on its shade (or intensity). The underlying theory of the technique is that the intensity of a pixel is determined in part by, among many other factors, the angle between surface normal (slope) and the illumination direction. See fig. 9.1.

Shape from Shading Geometry



Surface radiance and Lambertian Model

Let $S(x, y, z)$ be the surface radiance at point (x, y, z) relative to the camera frame, $\mathbf{n}$(x,y,z) be the surface normal at this point relative to the camera frame, and $\mathbf{L}$ be a vector representing the incident light also relative to the camera frame,

$$S(x, y, z) = \rho \mathbf{L}^T \mathbf{n} = \rho ||L|| \cos\theta \tag{24}$$

where ρ is the surface *albedo* and θ is the angle between L and n. A surface reflectance model that follows the radiance equation 24 is called *Lambertian Reflectance Model*.

The above equation assumes distant and single illumination source.

The Image irradiance Equation

Let $E(c,r)$ and $I(c,r)$ be the image irradiance and intensity at pixel (c,r) respectively. Then surface radiance $S(x,y,z)$ relates to E(c,r) via the fundamental radiometric equation, i.e.,

$$E(c,r) = S(x,y,z)\frac{\pi}{4}(\frac{d}{f})^2\cos^4\theta \tag{25}$$

where d, f, and θ represent the lens diameter, focal length, and the angle formed by the principal ray through the point at (x,y,z) and the optical axis. In the case of small angular aperture, this effect can be ignored; therefore the image irradiance can be regarded as proportional to the scene radiance, i.e.,

$$E(c,r) = \alpha S(x,y) = \alpha\rho L^T n$$

Since $E(c,r)$ is proportional to $I(c,r)$, so for Lambertian model

and a single distant light source, we have

$$I(c,r) = \alpha\beta\rho L^T n$$

where β is the coefficient that relates image irradiance to intensity. Assuming the optical system has been calibrated, the constant terms α and β can be dropped from the above equation, leading to

$$I(c,r) = \rho L^T n(x,y,z) \tag{26}$$

Equation 26 is the fundamental equation for shape from shading, assuming Lambertian illumination model, a single distant, and point light source. The goal of SFS is to recover surface normal n(x,y,z) for each 3D point (x,y,z) from its image intensity I(c,r).

Surface Normal Computation

Assume the surface depth z may be thought as a function of (x, y), i.e., z=f(x,y). A 3D point P can therefore be represented as $P(x, y) = (x, y, f(x, y))$. Hence the normal vector at (x,y,z) is

$$n(x, y, z) = \frac{\frac{\partial P}{\partial x} \times \frac{\partial P}{\partial y}}{||\frac{\partial P}{\partial x} \times \frac{\partial P}{\partial y}||} = \frac{[-p(x, y), -q(x, y), 1]^T}{\sqrt{1 + p^2(x, y) + q^2(x, y)}}$$

where $p = \frac{\partial f}{\partial x}$ and $q = \frac{\partial f}{\partial y}$ Note (x, y, z) is the coordinates of 3D point relative to the camera frame. As a result, p and q are specified with respect to the camera coordinate frame.

Therefore for surface that follows the Lambertian Model, ideally we have

$$I(c,r) = \rho L^T \frac{[-p(x,y), -q(x,y), 1]^T}{\sqrt{1 + p^2(x,y) + q^2(x,y)}} \tag{27}$$

As we don't know (x,y,z), we need compute p and q as a function of (c,r).

Weak Perspective Projection Assumption

Since the object is assumed to be far away from the viewer (camera), we can assume weak perspective projection. As a result, we have

$$\begin{aligned} c &= \frac{f s_x}{\bar{z}} x + c_0 \\ r &= \frac{f s_y}{\bar{z}} y + r_0 \end{aligned}$$

which can translate to

$$
\begin{aligned}
x &= \frac{(c - c_0)\bar{z}}{f s_x} \\
y &= \frac{(r - r_0)\bar{z}}{f s_y}
\end{aligned}
$$

$\bar{z}$ is the average object z distance to the camera. Given $P(x, y, f(x, y))$ and P is also a function of c and r, hence

$$
\begin{aligned}
\frac{\partial P}{\partial c} &= (\frac{\partial x}{\partial c}, 0, \frac{\partial f}{\partial x}\frac{\partial x}{\partial c}) = (\frac{\bar{z}}{f s_x}, 0, p(x, y)\frac{\bar{z}}{f s_x}) \\
\frac{\partial P}{\partial r} &= (0, \frac{\partial y}{\partial r}, \frac{\partial f}{\partial y}\frac{\partial y}{\partial r}) = (0, \frac{\bar{z}}{f s_y}, q(x, y)\frac{\bar{z}}{f s_y})
\end{aligned}
$$

Hence, we can easily prove that

$$
\frac{\frac{\partial P}{\partial c} \times \frac{\partial P}{\partial r}}{||\frac{\partial P}{\partial c} \times \frac{\partial P}{\partial r}||} = \frac{\frac{\partial P}{\partial x} \times \frac{\partial P}{\partial y}}{||\frac{\partial P}{\partial x} \times \frac{\partial P}{\partial y}||}
$$

Hence,

$$\begin{aligned} n(x,y,z) &= \frac{[-p(x,y),-q(x,y),1]^T}{\sqrt{1+p^2(x,y)+q^2(x,y)}} \\ &= \frac{[-p(c,r),-q(c,r),1]^T}{\sqrt{1+p^2(c,r)+q^2(c,r)}} \end{aligned} \tag{28}$$

Problem Statement

Given ρ and L, the problem is to reconstruct surface slopes p and q and then surface height z for which

$$I(c,r) = \rho R(L, p(c,r), q(c,r)) \tag{29}$$

where $R(L, p(c,r), q(c,r))$ is the reflectance map of a surface. For Lambertian Model, we have

$$R = L^T \frac{[-p(c,r), -q(c,r), 1]^T}{\sqrt{1 + p^2(c,r) + q^2(c,r)}}$$

In general, R is complicate or is estimated numerically via experiments. For each image point, we have one equation for two unknowns (p and q). The problem is under-determined.

Shape from Shading: A Local Approach

To uniquely solve p and q, we introduce a local approach, where we find the normal for each pixel locally. Given a pixel (c, r), let (p, q) be its normal. Assume all pixels in a local patch centered $N_{c,r}$ on (c, r) have the same p and q. Following Eq. 29, we have

$$I(c_i, r_i) = \rho L^T \frac{[-p, -q, 1]^T}{1 + p^2 + q^2}, \text{ where } (c_i, r_i) \in N_{c,r}.$$

Given L and ρ, we can hence solve for p and q by minimizing the following objective function

$$\sum_{(c_i, r_i) \in N_{c,r}} \{I(c_i, r_i) - \rho L^T \frac{[-p, -q, 1]^T}{1 + p^2 + q^2}\}^2$$

This approach is similar to Lucas-Kanade method for optical flow estimation. But the solution is non-linear.

A Calculus of Variational: a global approach

Different from the local approach, here we finds p and q for all pixels simultaneously. Find p and q for each point by minimizing

$$\epsilon = \sum_c \sum_r [(I(c,r) - \rho R(p,q)]^2 + \lambda(p_c^2 + p_r^2 + q_c^2 + q_r^2) \qquad (30)$$

where $R(p,q)$ is the reflectance map and λ is the Lagrange multiplier for imposing the *normal smoothness constraint*. $p_c = \frac{\partial p}{\partial c}$, $p_r = \frac{\partial p}{\partial r}$, $q_c = \frac{\partial q}{\partial c}$, $q_r = \frac{\partial q}{\partial r}$. λ is always positive and determines the relative importance between two terms (the fundamental equation and smoothness).

Find p and q for each point by minimizing ϵ

A global approach (cont'd)

In digital domain, p_c, p_r, q_c, q_r may be numerically approximated as

$$\begin{aligned} p_c &= p(c+1,r) - p(c,r) \\ p_r &= p(c,r+1) - p(c,r) \\ q_c &= q(c+1,r) - q(c,r) \\ q_r &= q(c,r+1) - q(c,r) \end{aligned}$$

A global approach (cont'd)

As a result, find p and q for each point by minimizing

$$\begin{aligned}\epsilon = \quad & \sum_c \sum_r [(I(c,r) - R(p,q)]^2 + \lambda[(p(c+1,r) - p(c,r))^2 \\ + \quad & (p(c,r+1) - p(c,r))^2 + (q(c+1,r) - q(c,r))^2 + \\ & (q(c,r+1) - q(c,r))^2]\end{aligned}$$

A global approach (cont'd)

Taking partial derivatives with respect to p and q and set them to zeros yield

$$\begin{aligned} p_{c,r} &= \bar{p}_{c,r} + \frac{1}{4\lambda}(I(c,r) - R(c,r))\frac{\partial R(c,r)}{\partial p} \\ q_{c,r} &= \bar{q}_{c,r} + \frac{1}{4\lambda}(I(c,r) - R(c,r))\frac{\partial R(c,r)}{\partial q} \end{aligned}$$

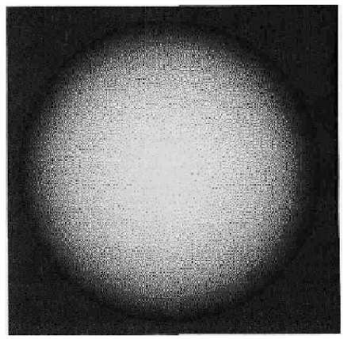
$\bar{p}_{c,r}$ and $\bar{q}_{c,r}$ are the average of p and q over the four nearest neighbors.

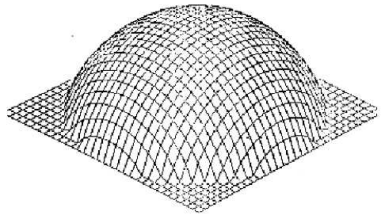
$$
\begin{aligned}
p_{c,r}^{k+1} &= \bar{p}_{c,r}^{k} + \frac{1}{4\lambda}(I(c,r) - R(c,r))\frac{\partial R(c,r)}{\partial p}|^{k} \\
q_{c,r}^{k+1} &= \bar{q}_{c,r}^{k} + \frac{1}{4\lambda}(I(c,r) - R(c,r))\frac{\partial R(c,r)}{\partial q}|^{k}
\end{aligned}
$$

Convergence Properties

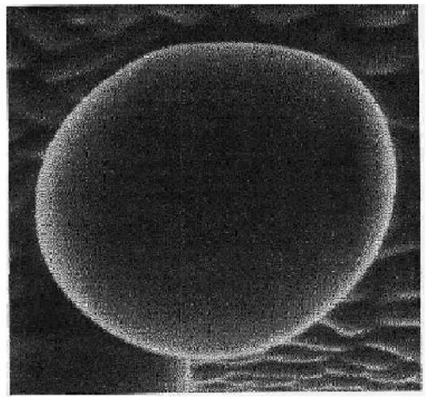
- Convergence-depending on the initial p and q.
- The optimal λ-normally around 1000.
- Stopping condition-residual error may not be always accurate

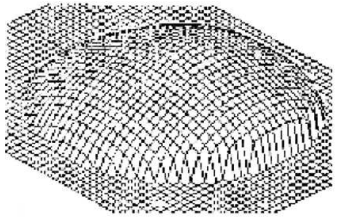
Example of Shape from Shading





Synthetic data

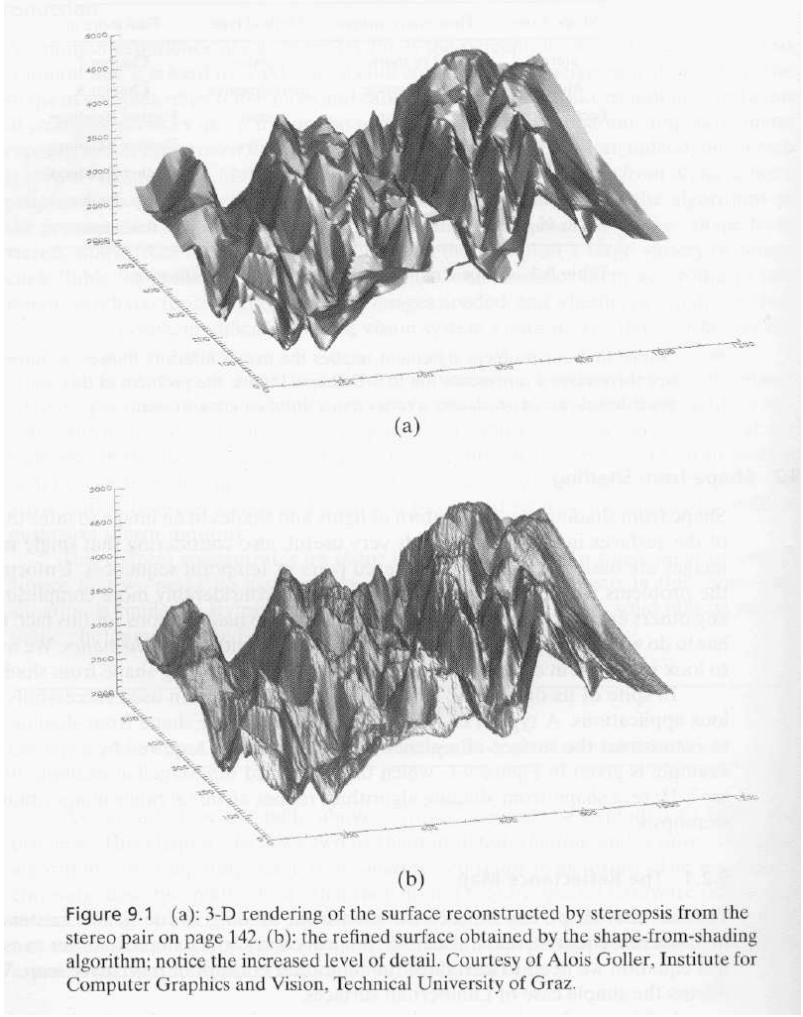




Scanning Electron Microscope image
(inverse intensity)

by Ikeuchi and Horn

Example of Shape from Shading



(a)

(b)

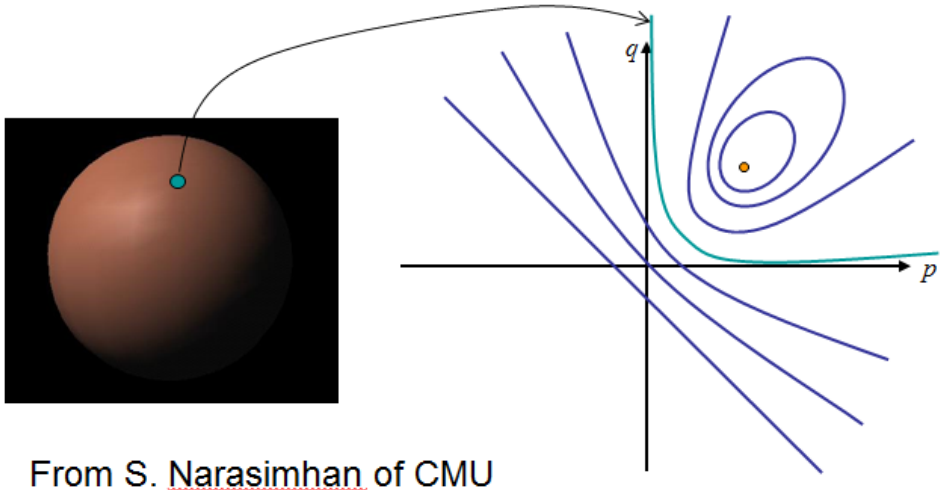
Figure 9.1 (a): 3-D rendering of the surface reconstructed by stereopsis from the stereo pair on page 142. (b): the refined surface obtained by the shape-from-shading algorithm; notice the increased level of detail. Courtesy of Alois Goller, Institute for Computer Graphics and Vision, Technical University of Graz.

Limitations with Shape from Shading Theory

- Assume single point distant illumination source
- Assume richly structued/texture surface
- Assume all visible surface points receive direct illumination
- Assume Lambertian surface.
- Assume weak perspective projection (a paper in ICCV03 to extend it to perspective projection)
- Assume Z=f(x,y)
- Solution is not unique

Shape from a Single Image?

- Given a single image of an object with known surface reflectance taken under a known light source, can we recover the shape of the object?
- Given $R(p,q)$ ((p_S, q_S) and surface reflectance) can we determine (p,q) uniquely for each image point?



From S. Narasimhan of CMU

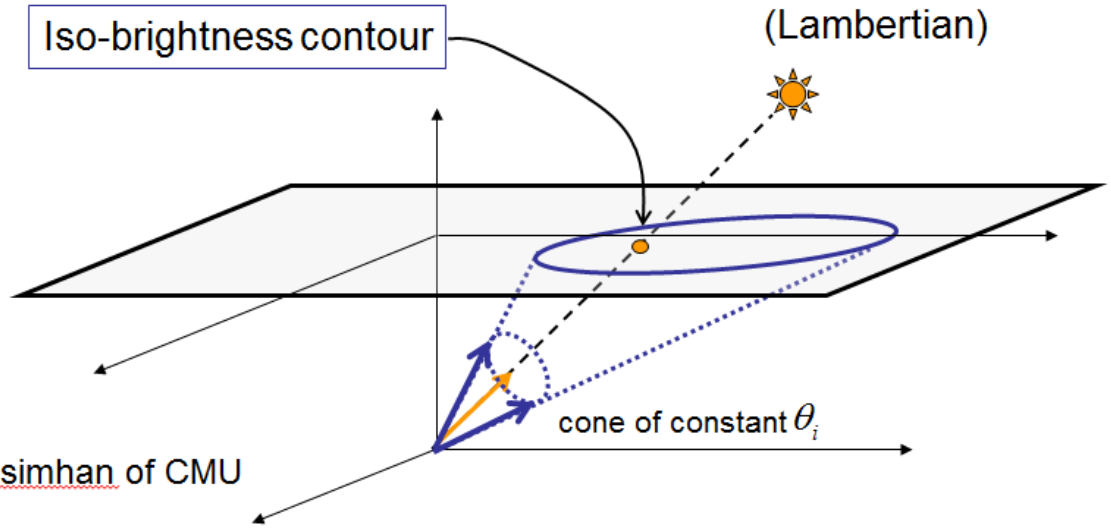
NO

Reflectance Map - RECAP

- Lambertian case

$$I = \cos\theta_i = \mathbf{n}\cdot\mathbf{s} = \frac{(pp_s + qq_s + 1)}{\sqrt{p^2+q^2+1}\sqrt{p_S^2+q_S^2+1}} = R(p,q)$$

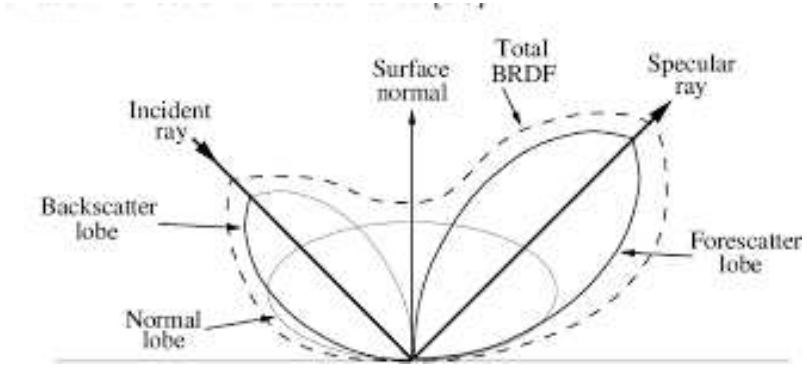
Reflectance Map
(Lambertian)



From S. Narasimhan of CMU

Other issues

- Estimate illumination direction.
- Estimate the reflectance map R(p,q).
  - A more general reflectance model is the **3-lobe model** or the **BRDF** (bidirection reflectance distribution function) model [18].



- Derive depth z from p and q.
  - Given p and q, z may be computed by minimizing $\epsilon = \sum_c \sum_r (z_c - p)^2 + (z_r - q)^2$, where the partial

directives p_c and p_r may be approximated numerically.

- Estimate surface albedo
- Handle specular surface, see Phone's model in [18]
- Deal with inter-reflections.

Estimate illumination direction

- Obtain a ball of known radius, with uniform shading or (color) and hence constant albedo.
- Place a single illuminator through a small hole and place the camera on the same side of the light
- Hang the ball in front of the light and the camera, turn the light on, and take a picture of the ball. Minimize the light interference from other sources.
- Use the image of the ball as well as its radius, recover the center of the ball, hence the equation of the ball w.r.t camera frame. If this is not possible, use two cameras to recover the center of the ball. We can recover the 3D coordinates of each point of the ball, given its image pixel. From the 3D coordinates, we can recover $N(c, r)$, the normal for each point.

Estimate illumination direction (cont'd)

- Given the known shape of the ball and its uniform (but unknown) albedo, for each pixel (c, r) in the image, we have $I(c, r) = \rho L \cdot N(c, r)$. With all image pixels, we can setup a system of linear equations to recover light direction L.

Statistical Shape from Shading

Statistical Shape from Shading incorporates a statistical shape prior to the objective function, allowing to use the global shape prior plus the local smoothness to further regularize the objective function.

Find p and q for each point by minimizing

$$\begin{aligned}\epsilon = \quad & \sum_c \sum_r [(I(c,r) - R(p,q)]^2 + \lambda[(p(c+1,r) - p(c,r))^2 \\ + \quad & (p(c,r+1) - p(c,r))^2 + (q(c+1,r) - q(c,r))^2 + \\ & (q(c,r+1) - q(c,r))]^2 - \gamma \log P(p,q)\end{aligned}$$

$P(p,q)$ is the prior probability of the surface shape. It can be manually specified or learnt from data. See [15].

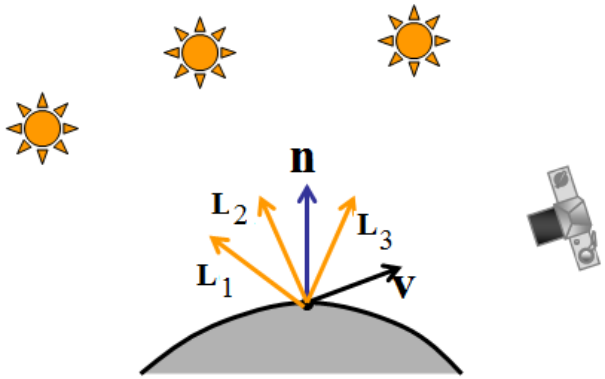
Photometric Stereo

At each point (x,y), we have

$$I_k(c, r) = \rho n^T(x, y, z)L_k,$$

where $k = 1, 2, \ldots N$ and $N > 3$, representing N light sources.

Photometric Stereo



Lambertian case:

$$I = \rho \mathbf{n} \cdot \mathbf{s}$$

Image irradiance:

$$I_1 = \rho \mathbf{n} \cdot \mathbf{L}_1$$
$$I_2 = \rho \mathbf{n} \cdot \mathbf{L}_2$$
$$I_3 = \rho \mathbf{n} \cdot \mathbf{L}_3$$

- We can write this in matrix form:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_2 \end{bmatrix} = \rho \begin{bmatrix} \mathbf{L}_1^T \\ \mathbf{L}_2^T \\ \mathbf{L}_3^T \end{bmatrix} \mathbf{n}$$

From S. Narasimhan of CMU

Photometric Stereo (cont'd)

Given N illumination directions, we can setup a system of linear equations involving

$$A = \rho \begin{pmatrix} L_{1_x} & L_{1_y} & L_{1_z} \\ L_{2_x} & L_{2_y} & L_{2_z} \\ \vdots & & \\ L_{N_x} & L_{N_y} & L_{N_z} \end{pmatrix} \qquad b = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_N \end{pmatrix}$$

n can be solved for by minimizing $||An - b||^2$, which leads to $n = \frac{(A^T A)^{-1} A^T b}{\rho}$, where $\rho = ||(A^T A)^{-1} A^T b||$.

Photometric Stereo (cont'd)

The assumptions of the above solution is that surface points can not be in shadows for any of the light sources and illumination directions are known.

Photometric calibration

For SFS and photometric stereo, the image intensity I needs to be photometrically calibrated to remove the camera gain (a) and offset (b).

Let I be the observed intensity and I' be the intensity without camera gain and offset. We have

$$I = a\rho L^T n + b$$

where a and b are the camera gain and offset.

Camera photometric calibration involves computing I' from I, i.e, $I' = \frac{I-b}{a}$. This may be accomplished via a geometric setup [8]. Via this calibration, we can eliminate the illumination strength and surface albedo from the equation, if we assume the illumination and albedo are the same for every image pixel,

producing the so-called intrinsic image.

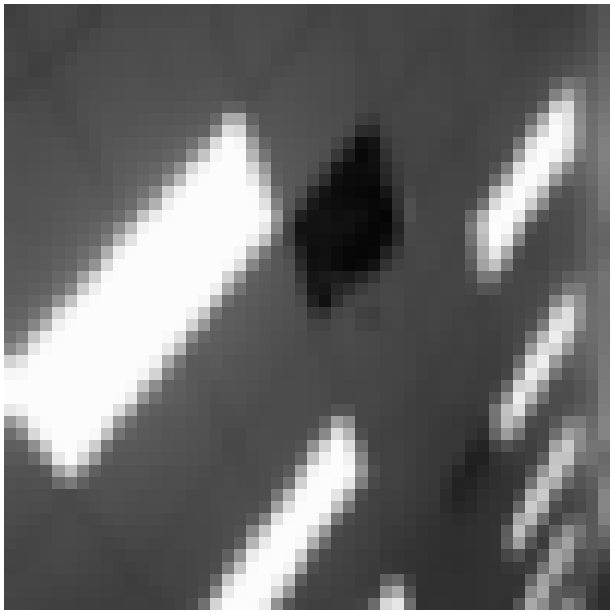
Shape from Texture

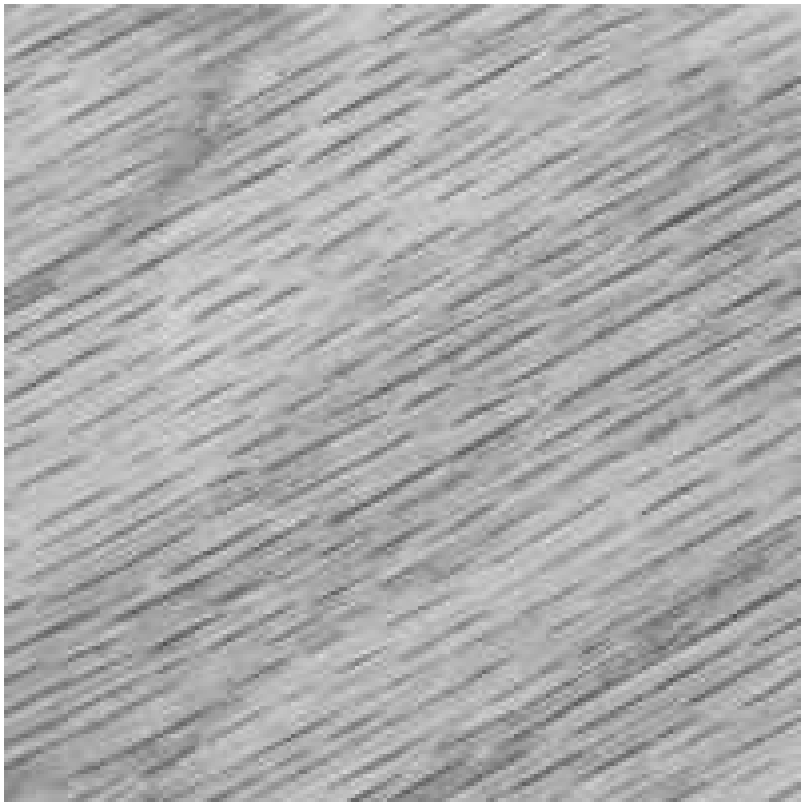
Given a single image of a surface consisting of texture patterns, estimate the shape of the surface from the distortions of the observed image texture patterns.

What is Texture

A texture surface is created by regular repetitions of a basic texture element or regular repetitions of certain statistical properties of surface color. The former may be referred to as the *deterministic* texture while the latter may be referred to as the *statistic* texture.

For deterministic textural elements, they can be characterized by the shape parameters such as parameters for lines or ellipses if they are the basic textural elements.





For statistic texture, they are often characterized by their statistical properties such as entropy, randomness, and spatial frequency over a region.

Sources of Texture Distortions

Texture distortions refer to texture properties change due to imaging process.

A uniformly-textured surfaces undergo two types of projective distortions

- Perspective distortion (distance)
- Foreshortening (orientation)

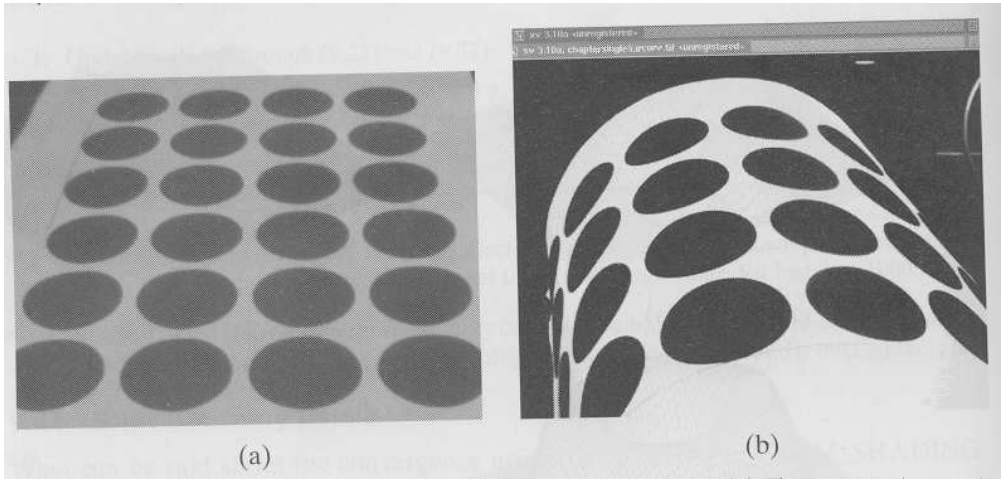
Sources of Texture Distortions (cont'd)

Perspective distortion causes a uniform compression in the area of a texture element as the distance between surface and the camera increases.

Foreshortening causes an anisotropic compression of a texture element.

The texture distortions provide information about the relative distances (perspective distortion) and orientations (foreshortening) of the textured surface in an image.

For deterministic texture, perspective distortion can be quantified by the area variations, while foreshortening distortion can be characterized by the ratio of the major and minor axes.



(a) (b)

Measures of Texture Distortions [1]

- Distortion measures (e.g., area, shape ratio)
- Distortion measure gradients (change of distortion measures)

While the first measure may be used to quantify distortion for a single texture element, the latter may be applicable to a region containing several textural elements.

Steps for Shape Estimation from Texture

- identify a region in the image containing a texture element.
- determine the textural properties to use
- compute texture distortions w.r.t the selected texture properties such as including area, aspect ratio, and density gradients.
- estimate surface normal and distance from the computed texture distortions.

This can be accomplished analytically or numerically with some regression methods such as support vector regression, regression neural network or even deep learning models. A good candidate for final project !.

Shape from Focus (Defocus)

- Object at different distances may appear different degrees of focus (or sharpness) due to depth of field
- Infer the depth from the pixel sharpness or from the degree of blurriness

See slides shape_from_focus.ppt for further information

f

Shape from Geometry [7, 19]

Also referred to as *shape from inverse perspective projection*, *shape from geometry* reconstructs a 3D geometric entity from a single image in conjunction with geometric constraints on the 3D geometric entity.

Active stereo actually exploit this idea.

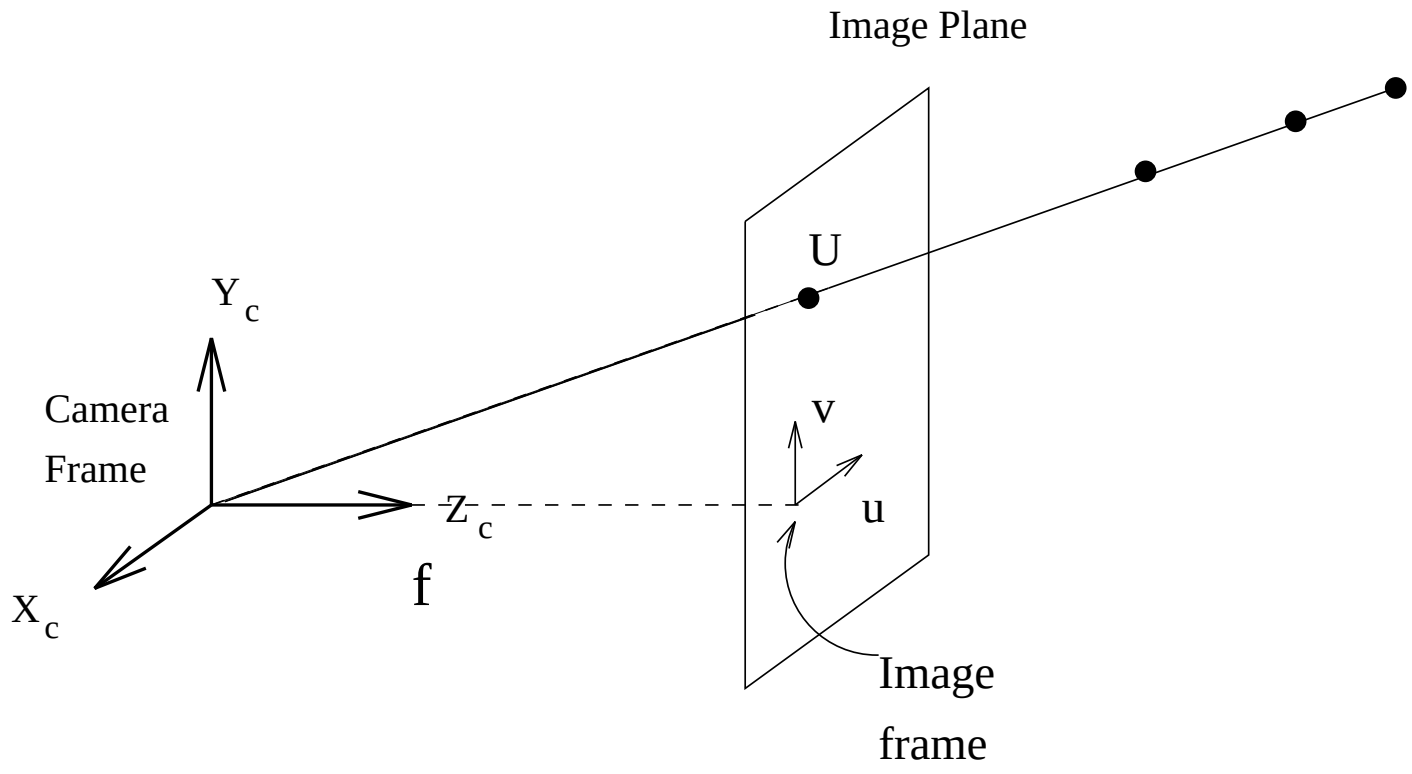
The geometric constraints may come from the world model of the object being viewed in the perspective projection.

More recent work constructs a deformable geometric model for the objects. 3D reconstruction of the objects can be formulated as estimating the parameters of the geometric models from their images.

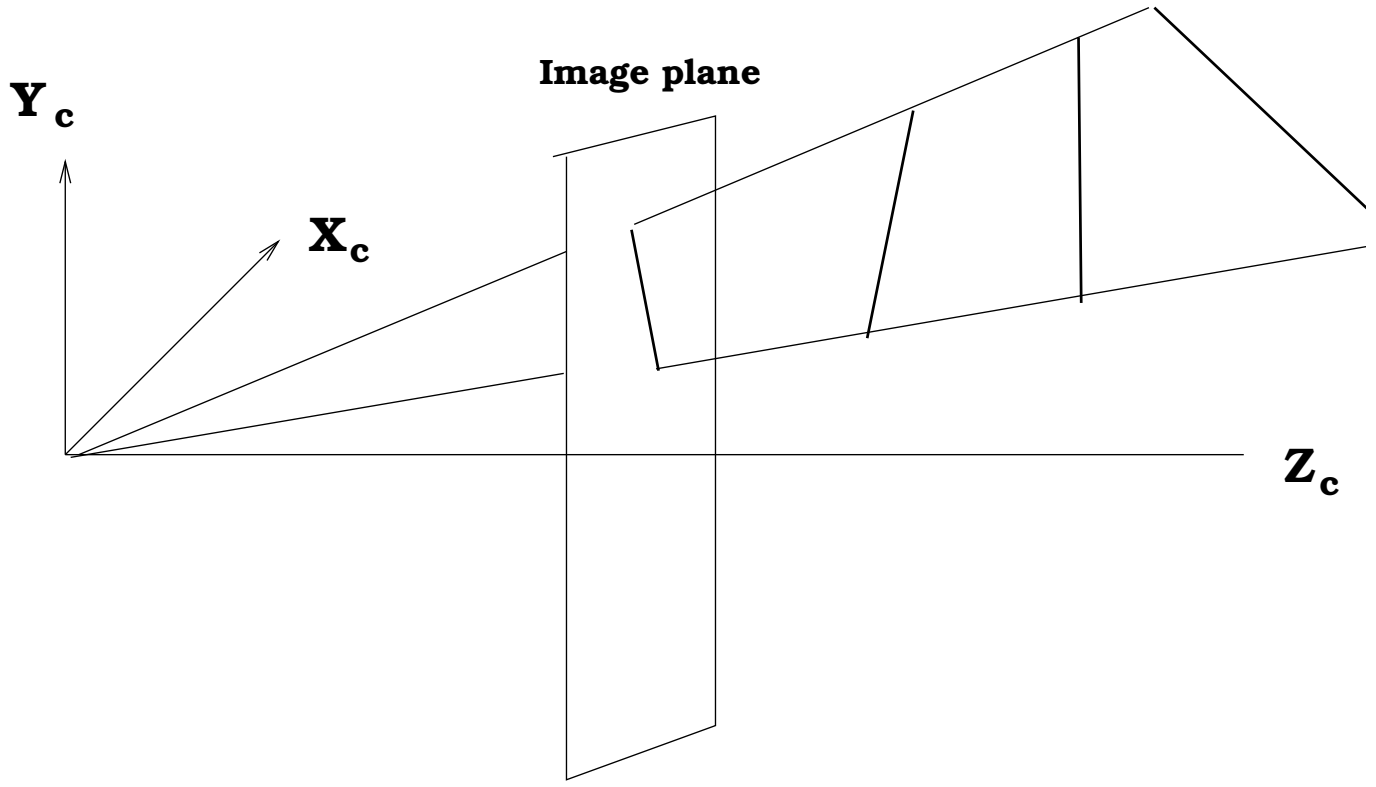
3D Geometric Entities

The 3D geometric entities may include points, lines, planes, and 3D curves like circles or ellipses

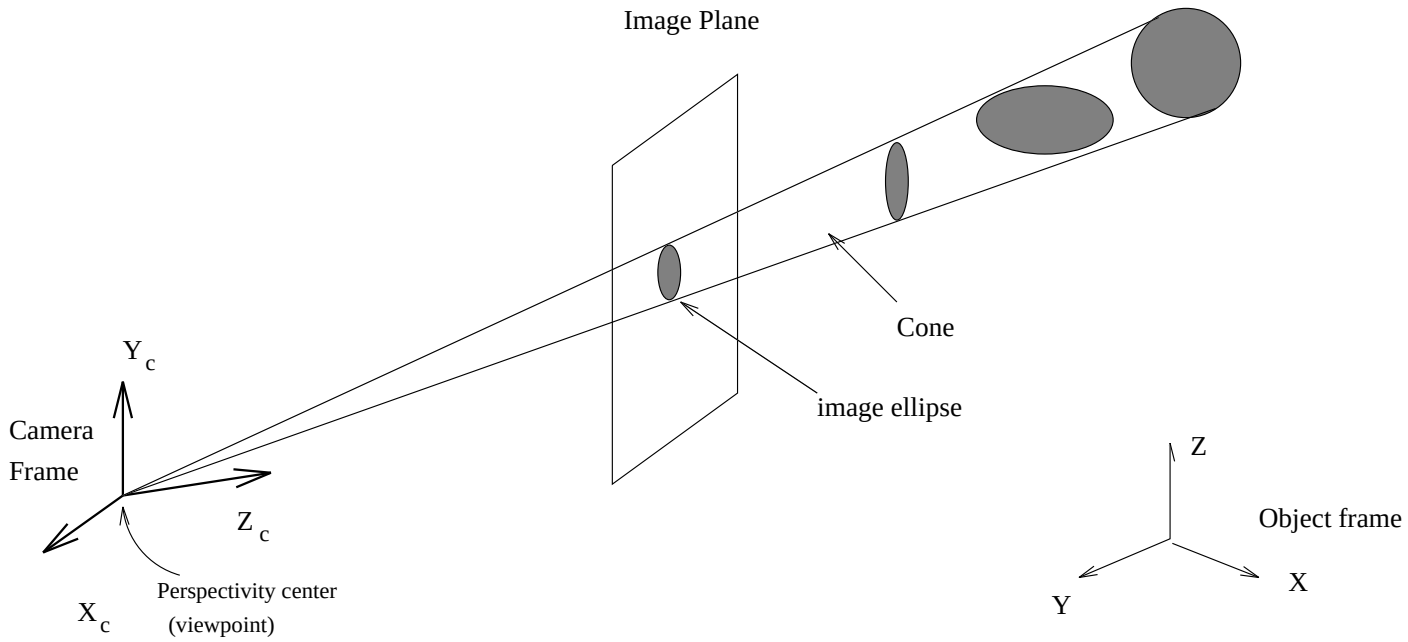
An Ill-posed Problem: Point



An Ill-posed Problem: Line



An Ill-posed Problem: Curves



Geometric Constraints

The geometric constraints employed include

- Euclidean distance constraints.
- Angular constraints

Euclidean Distance Constraints

The distance relationships may include distance among points, lines, and planes

- distance between points
  - two unknown 3D points
  - a known 3D point and an unknown 3D point
- distance between a point and a line
  - an unknown 3D point and a known line
  - an known 3D point and an unknown line
- distance between a point and a plane
  - an unknown 3D point and an unknown plane
  - a known 3D point and an unknown plane

Euclidean Distance Constraints (cont'd)

Distance relationship may also generalize to collinear and coplanar. They include

- three unknown 3D points are collinear
- an unknown 3D point is on a known line
- an unknown 3D point is on an unknown plane
- a known 3D point is on an unknown plane

Euclidean Angular Constraints

In addition to distance relationships, there also exist angular relationships. They may include angle formed by

- two unknown planes.
- three unknown non-collinear points.
- a known line and an unknown plane.

A good candidate for final class project!

Reconstruct 3D Points

A 3D point and its image offers two equations. The 3D point can be reconstructed if a constraint is available about the 3D point.

Reconstruct 3D Points

If an unknown 3D point X is located on a known plane, then X can be solved for.

Reconstruct 3D Points

If the distance between X and a known 3D point Y is known, then X can be solved for.

Reconstruct 3D Points (cont'd)

Given the images of three 3D points and the distances between the 3D points. Then the 3D points can be reconstructed. This is the famous 3-point problem. The solution is not unique however. If we know one of the 3 points, then the other two points can be determined uniquely.

Reconstruct 3D Points (cont'd)

Similarly, for three non-collinear points, can they be reconstructed if the three angles they form are known? Note the three angles are not independent of each other (they sum to 180).

Reconstruct 3D Points

If an unknown 3D point X is located on an known 3D line, then X can be solved for.

Reconstruct 3D Lines

A 3D line may be represented by a point on the line P_0 and its direction cosine N. Hence,

$$P_0 + \lambda N$$

is the equation of a line, where λ is a scaler.

A line therefore has a total of 5 parameters (3 for P_0 and 2 for N). The number of parameters may reduce to 4 if P_0 is chosen such that $P_0^T N = 0$. We can hence say a 3D line has four degrees of freedom.

The relationships between a 3D line and 2D line

Let the 3D line be $P_0 + \lambda N$ and $P_0 = (X_0, Y_0, Z_0)$. Let the corresponding image line be $\alpha c + \beta r + \gamma = 0$. Let n be the normal of the backprojection plane formed by the 3D line and the perspectivity center, and n be derived as $n = W^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$. We then have

$$n^T P_0 = 0 \quad (31)$$

$$n^T N = 0 \quad (32)$$

where the first equation says that P_0 is located on the backprojection plane. It is the same as saying image projection p_0

of P_0 is located on the image line. Note as we do not know where p_0 is, we cannot setup additional equations that relate p_0 to P_0.

The second equation derives from the fact that the 3D line is located on the backprojection plane.

Each line offers two equations for four line parameters, additional constraints are needed.

Lines in a known plane

Let the plane parameters be d (origin distance to plane) and V (plane normal), and the line parameters be P_0 and N, we then have

$$\begin{aligned} P_0^T V &= d \\ N^T V &= 0 \end{aligned}$$

We can then solve for P_0 and N using the above two equations along with the two line projection equations. The 3D line is the intersection of the back-projection plane with the given known plane.

Parallel Lines with known vanishing points

Given the vanishing point of a set of parallel lines, the orientation of the lines can be determined as follows

Let the vanishing point be (u, v) and $N = (N_x, N_y, N_z)$, then we have

$$\begin{aligned} u &= f\frac{N_x}{N_z} \\ v &= f\frac{N_y}{N_z} \end{aligned}$$

Using the above equation and the fact that $N_x^2 + N_y^2 + N_z^2 = 1$, we can solve for N.

Reconstruct 3D Lines: N Pencil Lines

Given $M \geq 3$ lines intersecting at point P_0, we can detect $p_0 = (c_0, r_0)$ in the image and setup equations to relate p_0 to P_0, i.e.,

$$\lambda \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} = W P_0 \tag{33}$$

In addition, we have two line projection equations for each line

$$n_i^T P_0 = 0 \tag{34}$$

$$n_i^T N_i = 0 \tag{35}$$

where i=1,2,..,N . Note combining eq. 33 with eq. 34 seems to provide enough equations to solve for P_0 uniquely. But actually,

their combination can only solve for P_0 up to a scale factor, as they both can only solve P_0 up to a scale factor.

In addition, Eq. 35 only allows solving N_i up to a scale factor.

Reconstruct 3D Lines: N Pencil Lines

Given $M \geq 3$ lines intersecting at a point with known angles α_{ij} (e.g., orthogonal to each other) between each pair of 3D lines as shown in the figure below

We can then use eq. $n_i^T N_i = 0$ plus the equations from the known angles, i.e., $N_i^T N_j = \cos(a_{ij})$, to uniquely solve for N_i. The solution is however non-linear. Unfortunately, given N_i, we can still only solve for P_0 up to a scale factor as N_i is independent of P_0 in the two line projection equations (Eqs. 34 and 35). The

scale factor can be solved if we are given the distance of two unknown points on one side of the wall.

Collinear Points with Known Interpoint Distances

Suppose we observe the perspective projection of a 3D line whose position and orientation are unknown. On this line there are $N(> 2)$ distinguished points with known interpoint distances. We also observe their corresponding image points. This constitutes enough information to determine all the parameters of the line as well as the 3D positions of the points.

Collinear Points with Known Interpoint Distances (cont'd)

Let $P_0 = (P_{0_x}, P_{0_y}, P_{0_z})$ and $N = (N_x, N_y, N_z)$ be the first point on the 3D line and the orientation of the line. Let P_1 be the second point on the line and (c_1, r_1) be the image projection of P_1, then we have

$$P_1 = P_0 + d_1 N$$

where d_1 is the distance between P_0 and P_1.

From the perspective projection equation, we have

$$\begin{aligned} c_1 &= f\frac{P_{0_x} + d_1 N_x}{P_{0_z} + d_1 N_z} \\ r_1 &= f\frac{P_{0_y} + d_1 N_y}{P_{0_z} + d_1 N_z} \end{aligned}$$

Combining with the two line projection equations, we can setup a system of linear equations involving the unknown parameters P_0 and N, which can be solved by a constrained linear least-squares method. Once P_0 and N are solved, we can then use the inter-point distance to solve the 3D coordinates for other points on the line.

Lines with known length and orientation

Given the length of a line segment as d and its orientation N, the coordinates of two endpoints can be reconstructed.

Let the two endpoints be $X_1 = (x_1, y_1, z_1)$ and $X_2 = (x_2, y_2, z_2)$ and their corresponding image points be $U_1 = (c_1, r_1)$ and $U_2 = (c_2, r_2)$. Since

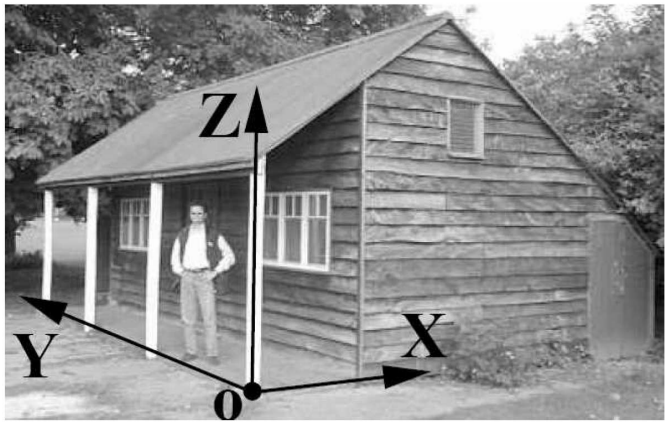
$$X_2 = X_1 + \lambda N$$

$$
\begin{aligned}
c_1 &= \frac{f x_1}{z_1} \\
r_1 &= \frac{f y_1}{z_1} \\
c_2 &= \frac{f(x_1 + dN_x)}{z_1 + dN_z} \\
r_2 &= \frac{f(y_1 + dN_y)}{z_1 + dN_z}
\end{aligned}
$$

Hence the above four equations yield an over-determined system of linear equations involving 3 unknowns x_1, y_1, and z_1. Solving X_1, we can then obtain X_2.

Single View Metrology

Given an image of a shed on the left below, describe what you need to reconstruct the 3D model of the shed on the right [a]. This can be a homework. Additional information about single view metrology can be found in [7, 19, 2].



[a] Image from Fig. 11.4 of [17]

3D Conic Reconstruction

Given a 3D conic (ellipse or circle) and its image, we can not uniquely determine the 3D conic.

Given we know the 3D conic is a circle and the radius of the 3D circle, we can then reconstruct the 3D circle from its image as follows.

Let A be the matrix that specifies the image ellipses, we hence have

$$\begin{pmatrix} c \\ r \\ 1 \end{pmatrix}^T A \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = 0.$$

[a] Substituting $\lambda \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = W \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ into the ellipse equations above yields the equation for the the cone in the camera frames as follows

---

[a]Note the generic equation for a 2D image ellipse is : $ax^2 + bxy + cy^2 + dx + ey + f = 0$. It can be expressed in matrix format as $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}^T A \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$,

where

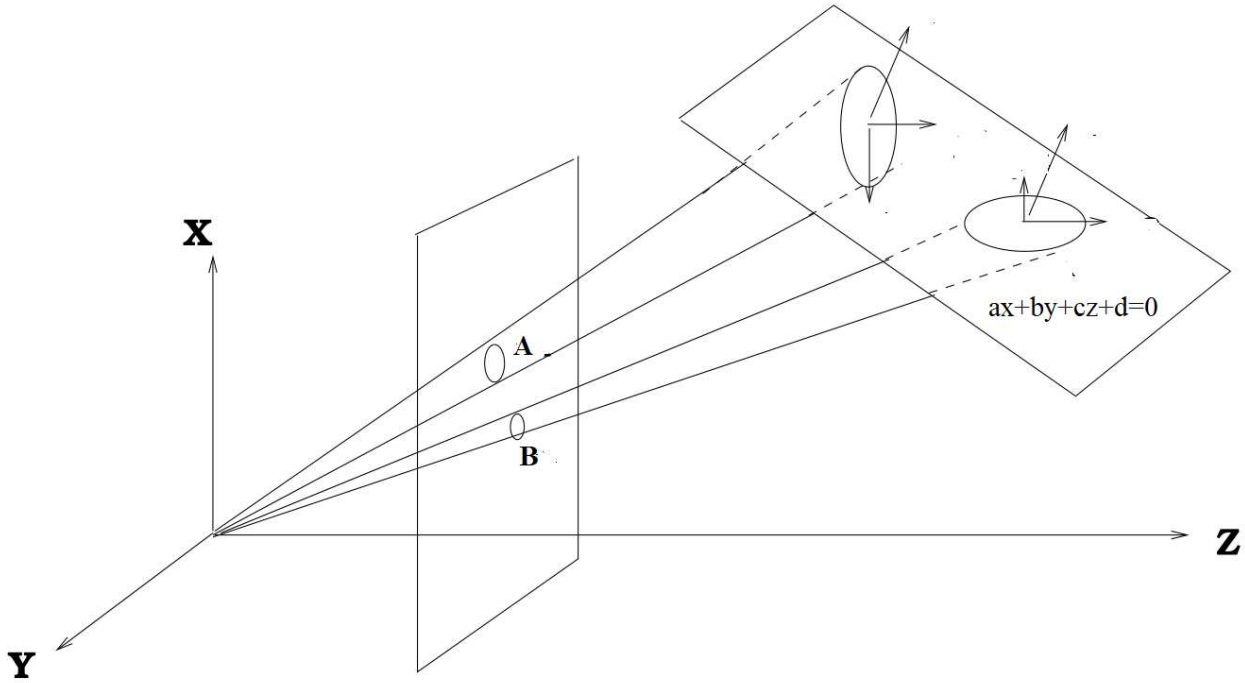
$$A = \begin{pmatrix} a & \frac{1}{2}b & \frac{1}{2}d \\ \frac{1}{2}b & c & \frac{1}{2}e \\ \frac{1}{2}d & \frac{1}{2}e & f \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}^T W^T A W \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0.$$

Assume the equation of the plane, where the 3D conic curve is located on, is $ax + by + cz + d$=0 w.r.t the camera frame. Plugging $z = -\frac{ax+by+d}{c}$ into the conic equation yields the 3D conic equation in terms x and y and the plane parameter ratio $(a/c, b/c, d/c)$. Solving the plane parameter ratios using the information that the 3D conic is a circle with known radius.

3D Conic Reconstruction

If we are given the images of two 3D coplanar conic curves of identical size but different orientations, we can then reconstruct the two 3D conic curves.



Let the A and B be the matrix that respectively specify the two

observed image conics. We can then construct the equations for the two cones that the two image conics form as follows

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}^T W^T A W \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0.$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}^T W^T B W \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0.$$

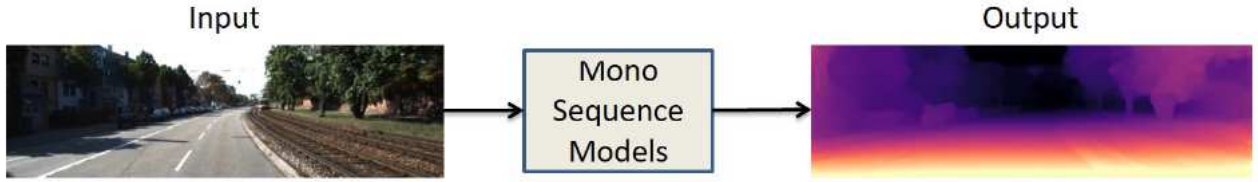
Assume the two 3D conics are located on a 3D plane $ax + by + cz + d{=}0$ [a] w.r.t the camera frame. Plugging $z = -\frac{ax+by+d}{c}$ into the two conic equations, yielding two 3D conic equations in terms of X and Y. We can then solve the plane

[a]Let two planes be: $p_1 = (a, b, c, d)^T$ and $p_2 = (a', b', c', d')^T$. Their intersection produces a line $l = p_1 \times p_2$, where $\times$ represents cross product.

parameters using the two conic equations and an additional equation derived from the fact that the two 3D conics are identical in size. Given the plane parameters, we can then obtain the equations for the two 3D conic curves.

Deep Learning for Monocular Depth Estimation

For a review of recent efforts for monocular depth estimation using deep neural networks, see references [13, 21].



Given the GT depth, the model can be trained in a supervised learning. It can also be trained with weakly supervised learning by using relationships between multiple images. In addition, the adversarial loss and structured loss (e.g. CRF) can also be aded to the loss function as additional regularization terms.

Hierarchy of 3D Reconstruction Techniques

1. Passive approach
   - Passive stereo (two images)
   - Shape from X (shading, texture, geometry, photometric stereo, focus) (one image)
   - Shape from image sequences (multiple images)
2. Active approach
   - Active stereo (one projector and one camera like Kinect)
   - Radar sensor, Lidar (laser imaging, detection, and ranging), and other time of flight sensors

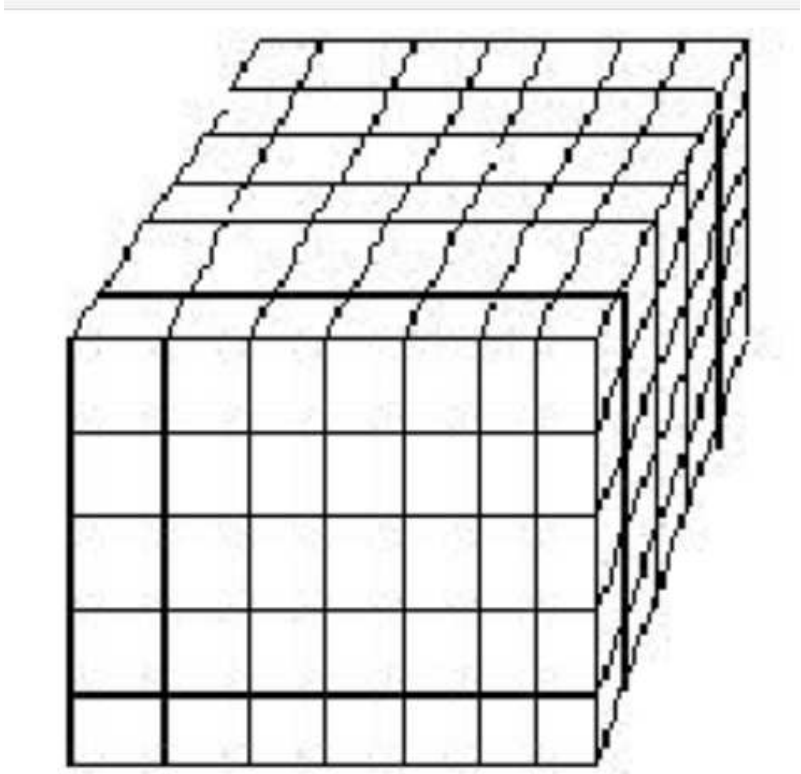
Deep Learning Based 3D Reconstruction

The main idea is to learn a mapping function $f(I, \theta)$ that maps the images I of an object into a 3D representations of the object. A recent survey on image based 3D reconstruction can be found in [6].

The methods can be divided into different categories, depending on the number of input images, including one image (monocular approach), multiple images, and a sequence of images (video).

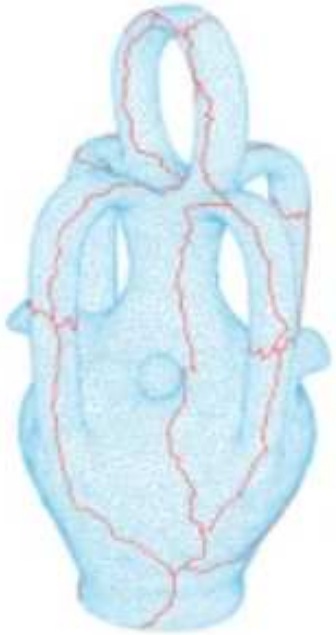
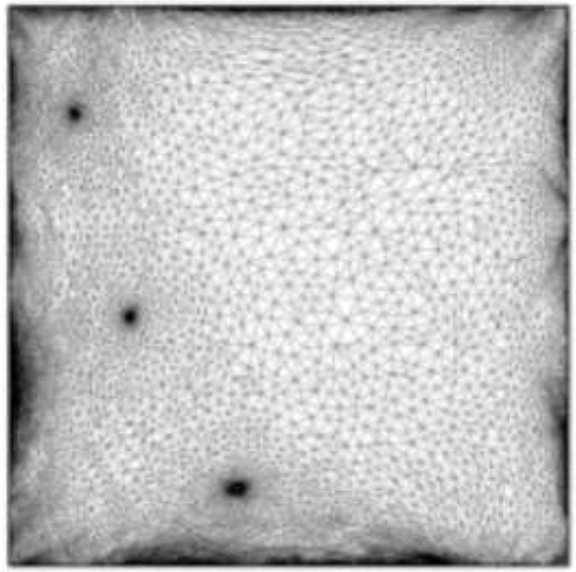
3D representations

The 3D representations can also vary, including volumetric representation and surface representations. Volumetric representation uses a 3D volume to represent a 3D object, where each voxel assumes a binary value to indicate if it is part of the object or not. It captures 3D points of both inside and on the surface of the object. The binary value can also be replaced by a probability value, indicating the probability of the voxel is part of the object.

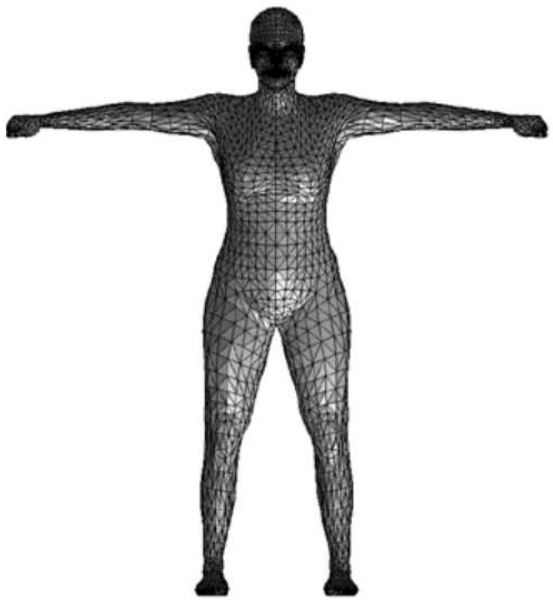


3D representations (cont'd)

Surface representations only capture 3D points on the object surface. They can be further divided into parametric surface representation and non-parametric surface representation. Parametric surface representations represent object surface with a parametric model. One surface parameterizations is the spherical parameterizations, which represent an arbitrary 3D surface by cutting the surface into disk-like or triangle patches and unfolding them into a 2D mesh.

Cutting	Parameterization
	

Another parametric 3D surface representation is the deformation-based representation, which represents the 3D surface with a 3D mesh and assumes the mesh can deform from its mean shape to produce different 3D shape.



Mean body mesh

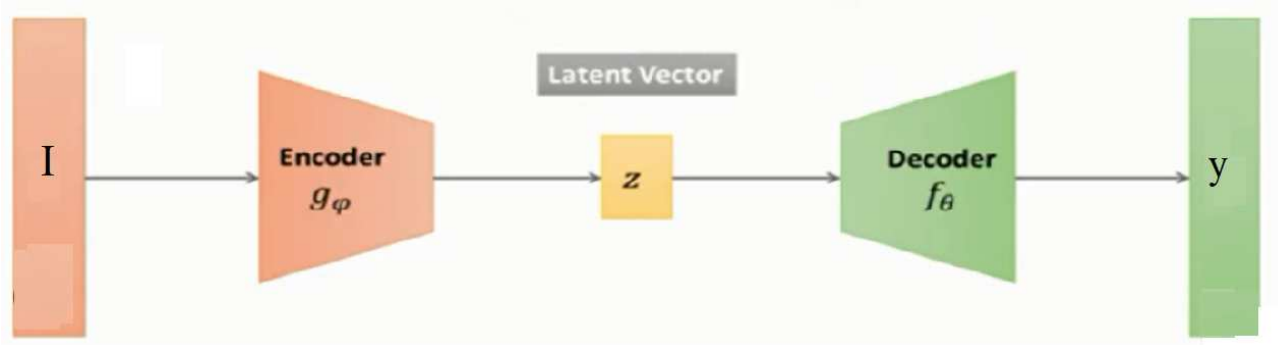
Deformed body mesh

3D representations (cont'd)

The most common non-parametric 3D surface model is 3D point cloud

Deep Model Architectures

The common architecture for 3D reconstruction from images is the encoder and decoder structure, where the encoder takes the input image and produces a latent reprobation Z and the decoder takes Z as input and outputs the 3D representations y in terms of 3D volume, parameters for the parametric surface representations, and 3D coordinates for point cloud representation.



The training loss function includes terms for 3D loss (reconstruction loss-volumetric loss, parameter loss, or 3D point loss) or 2D loss (projection loss).

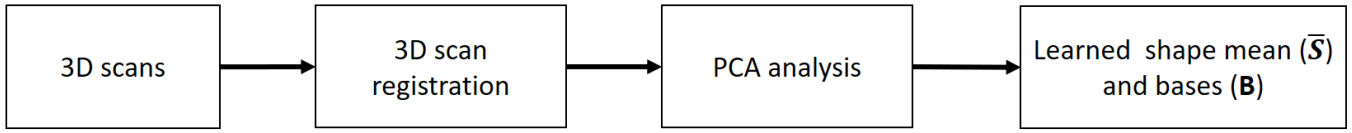
Deformable Mesh based 3D reconstructions

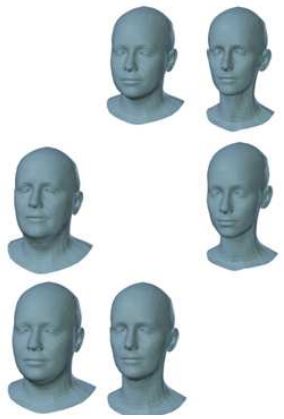
The basic idea is to learn a 3D deformable (morphable) model of 3D objects from their 3D training data and use the deformable model to perform 3D reconstruction of an object by estimating the deformable model parameters.

It includes two steps: deformable model construction and 3D object reconstruction using the learned deformable model.

Deformable model reconstruction: given a large number of 3D scans of different objects from the same category, a deformable model is constructed through a PCA analysis in terms of bases for the 3D object geometry and its mean shape.

For example, to construct a deformable model for human faces, a database of 3D scans of different faces with various facial shapes is used.



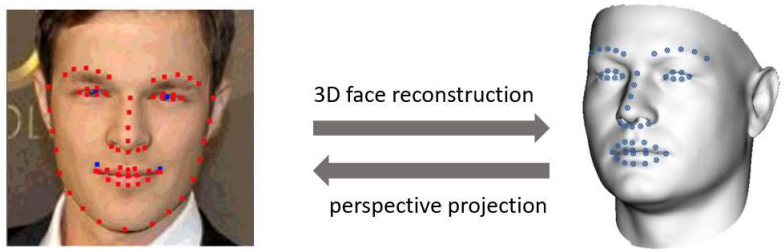


$$S = \bar{S} + \mathbf{B}\boldsymbol{\alpha}$$

where S: target 3D face, $\bar{S}$: mean shape, $\mathbf{B}$: shape basis, and $\boldsymbol{\alpha}$: shape coefficients.

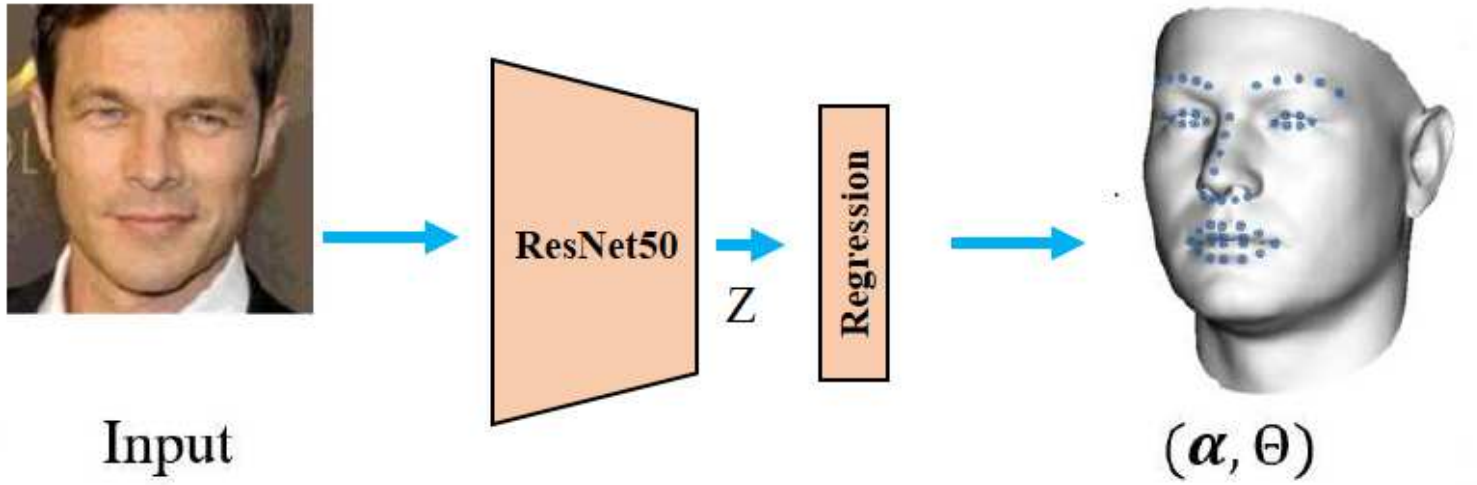
Deformable Mesh based 3D reconstructions (cont'd)

Given the learned deformable model and an image of an 3D object, 3D reconstruction of the object can be formulated as an optimization problem that estimates model parameters $\boldsymbol{\alpha}$ and camera parameters Θ by minimizing the differences between the projected image and the observed image in terms of either positional or intensity differences.



$$\boldsymbol{\alpha}^*, \Theta^* = \arg\min_{\boldsymbol{\alpha},\Theta} \underbrace{|| \overbrace{P}^{\text{projection matrix}} (\bar{S} + \mathbf{B}\boldsymbol{\alpha}, \Theta) - I||_2^2}_{\text{projection/rendering loss}}$$

Recent developments in deep learning allows to efficiently predict the 3D shape $\boldsymbol{\alpha}$ and camera parameters Θ from an image through a regression convolutional neural network. Excellent results have been achieved in 3D deformable model reconstruction.



Datasets for image-based 3D reconstruction

TABLE 5: Some of the datasets that are used to train and evaluate the performance of deep learning-based 3D reconstruction algorithms. "img": image. "obj": object. "Bkg": background. "cats": categories. "GT": ground truth.

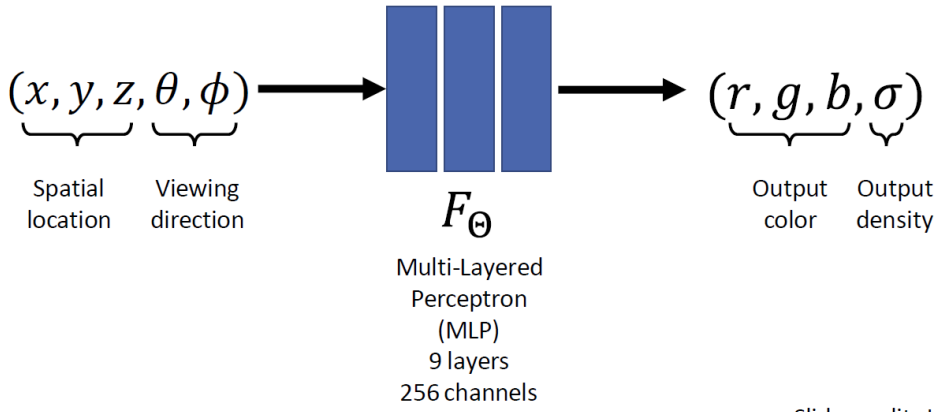
	Year	Images					Objects		3D ground truth			Camera
		No. imgs	Size	objs per img	Type	Bkg	Type	No. cats	No.	Type	Img with 3D GT	params
ShapeNet [139]	2015	–	–	Single	rendered	Uniform	Generic	55	51,300	3D model	51,300	Intrinsic
ModelNet [3]	2015	–	–	Single	Rendered	Uniform	Generic	662	127,915	3D model	127,915	Intrinsic
IKEA [140]	2013	759	Variable	Single	Real, indoor	Cluttered	Generic	7	219	3D model	759	Intrinsic+extrinsic
Pix3D [9]	2018	9,531	110 × 110 to 3264 × 2448	Single	Real, indoor	Cluttered	Generic	9	1015	3D model	9,531	Focal length, extrinsic
PASCAL 3D+ [141]	2014	30,899	Variable	Multiple	Real, indoor, outdoor	Cluttered	Generic	12	36,000	3D model	30,809	Intrinsic+extrinsic
ObjectNet3D [142]	2016	90,127	Variable	Multiple	Real, indoor, outdoor	Cluttered	Generic	100	44,147	3D model	90,127	Intrinsic+extrinsic
KITTI12 [143]	2012	41,778	1240 × 376	Multiple	Real, outdoor	Cluttered	Generic	2	40,000	Point cloud	12,000	Intrinsic+extrinsic
ScanNet [144]	2017, 2018	2,492,518	640 × 480, RGB 1296 × 968	Multiple	Real, indoor	Cluttered	Generic	296	36,123	Dense depth	2,492,518	Intrinsic+extrinsic
Stanford Car [145]	2013	16,185	Variable	Single	Real, outdoor	Cluttered	Cars	196	–	–	–	–
Caltech-UCSD Birds 200 [146], [147]	2010, 2011	6,033	Variable	Single	Real	Cluttered	Birds	200	–	–	–	Extrinsic
SUNCG [148]	2017	130,269	–	Multiple	Synthetic	Cluttered	Generic	84	5,697,217	Depth, voxel grid	130,269	Intrinsic
Stanford 2D-3D-S [149], [150]	2016, 2017	70,496	1080 × 1080	Multiple	Real, indoor	Cluttered	Generic	13	6,005	Point cloud, mesh	70,496	Intrinsic + extrinsic
ETHZ CVL RueMonge [151]	2014	428	–	Multiple	Real, outdoor	Cluttered	Generic	8	–	Point cloud, mesh	428	Intrinsic
NYU V2 [152]	2012	1,449	640 × 480	Multiple	Real, indoor	Cluttered	Generic	894	35,064	Dense depth, 3D points	1,449	Intrinsic

Neural Radiance Fields (NeRF)

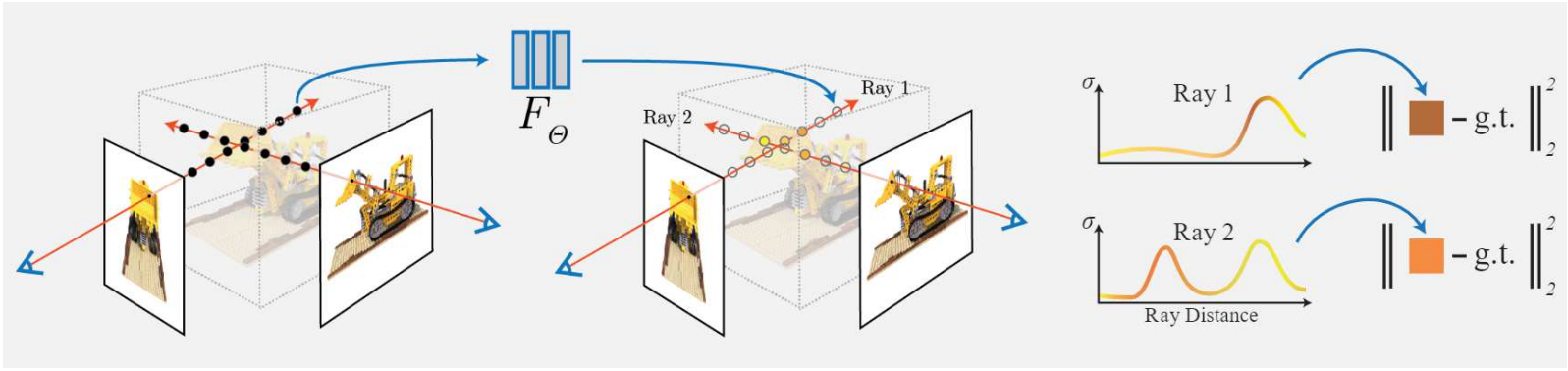
NeRF **learns an implicit continuous volumetric representations** of a complex object or a scene from their 2D images and uses the volumetric representation to **render images for arbitrary views**. The complex geometry of a 3D object/scene is implicitly represented using **a neural network**.

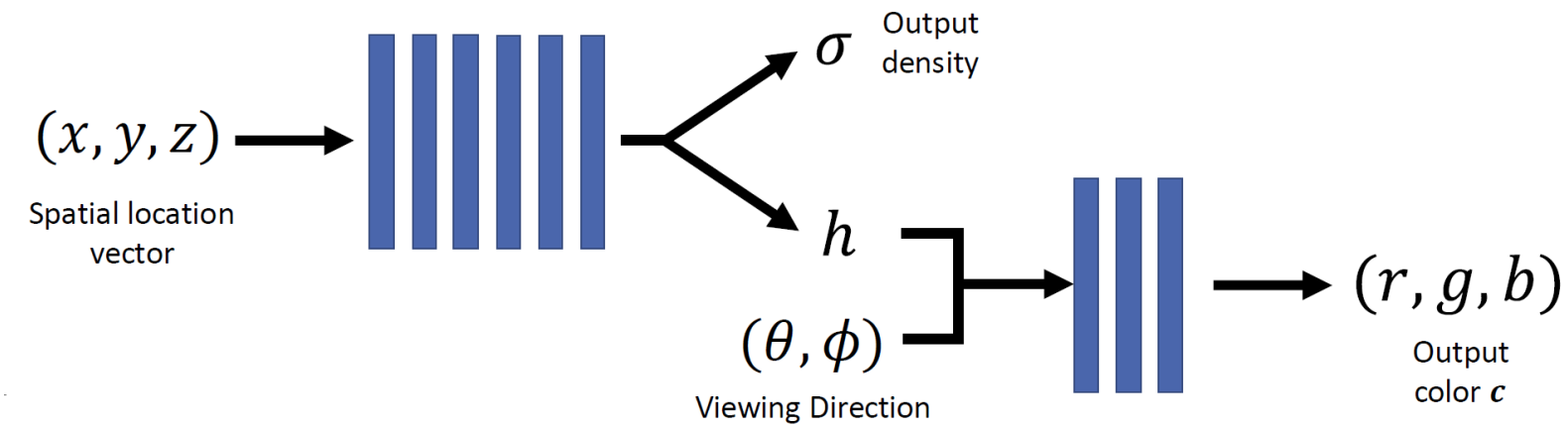
NeRF View Synthesis

Volumetric Rendering: given the volumetric representation of a 3D object with a neural network, produce an image of the 3D object in a given view direction(i.e., given the camera pose).



Input: 3D coordinates of a 3D point and the camera pose (i.e., view direction) Output: the RGB color for the corresponding image pixel and its the density (opacity) (0-visible or 1-invisible).





σ (spatial location)
$\boldsymbol{c}$ (spatial location, viewing direction)

An image can be rendered given all possible 3D points as input to the

neural network. The NeRF neural network can be used to render an image for an arbitrary view for a complex 3D scene.

NeRF Rendering Examples

Inputs: sparsely sampled images of scene

Output: includes new rendered views

matthewtancik.com/nerf

Nearest Input

NeRF

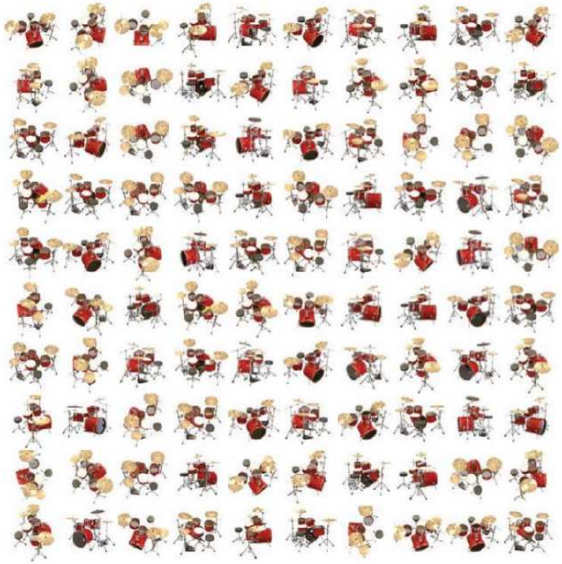
Nearest Input

NeRF

NeRF Training

Training: Collect data from different views with different camera parameters, assuming the camera pose parameters are known.

Multiview Images of a single scene



Camera poses

Learn the parameters of the neural network F_Θ by minimizing the differences between the rendered images and the observed images,

i.e.,

$$\Theta^* = \arg\min_{\Theta} \sum_{x,y,z \in \mathcal{V}} || \underbrace{R(x,y,z,\theta,\phi|\Theta)}_{\text{Image rendering}} - I(\theta,\phi,x,y,z)||_2^2$$

Note the learning process implicitly performs correspondence matching, 3D triangulation of the matched points, and rendering function for each camera view. It hence requires a lot of training data from different view points and under different illumination conditions.

In addition, the explicit volumetric representation of the 3D scene can be extracted from the neural network using the density function for each voxel and its 3D coordinates.

NeRF Limitations

- NeRF only works with static scenes
- A trained NeRF model does not generalize well to other scenes well
- Training is computationally expensive, taking 1-2 days to train a scene on a GPU
- Inference (rendering) is slow: each pixel in the synthesized images requires volume rendering.

The original NeRF paper may be found in [12]. A recent survey on NeRF methods can be found in [3].

References

[1] R. Blostein and N. Ahuja. Shape from texture: integrating texture-element extraction and surface estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(12):1233–1251, 1989.

[2] Antonio Criminisi, Ian Reid, and Andrew Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[3] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022.

[4] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[5] Mohd Saad Hamid, NurulFajar Abd Manap, Rostam Affendi Hamzah, and Ahmad Fauzan Kadmin. Stereo matching algorithm based on deep

learning: A survey. *Journal of King Saud University-Computer and Information Sciences*, 34(5):1663–1673, 2022.

[6] Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun. Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1578–1604, 2019.

[7] R. M. Haralick. Monocular vision using inverse perspective projection geometry: analytic relations. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 370–378, 1989.

[8] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 2. Addison-Wesley Publishing Company, 1993.

[9] B. K. P. Horn. Relative orientation. *International journal of computer vision*, 4(1):59–78, 1990.

[10] Patrick Knobelreiter, Christian Reinbacher, Alexander Shekhovtsov, and Thomas Pock. End-to-end training of hybrid cnn-crf models for stereo.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2339–2348, 2017.

[11] Hamid Laga, Laurent Valentin Jospin, Farid Boussaid, and Mohammed Bennamoun. A survey on deep learning techniques for stereo-based depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):1738–1764, 2020.

[12] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[13] Matteo Poggi, Fabio Tosi, Konstantinos Batsos, Philippos Mordohai, and Stefano Mattoccia. On the synergies between machine learning and stereo: a survey. *https://arxiv.org/abs/2004.08566*, abs/2004.08566, 2020.

[14] G. Poggio and T. Poggio. The analysis of steropsis. *Annu. Rev. Neurosci.*, (7):379–412, 1984.

[15] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. 3-d depth reconstruction from a single still image. *International journal of computer vision*, 76(1):53–69, 2008.

[16] De Ma Song. Conics-based stereo, motion estimation, and pose determination. *International journal of computer vision*, 10(1):7–25, 1993.

[17] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2021.

[18] H. D. Tagare and R. J. P. deFigueiredo. A theory of photometric stereo for a class of diffuse non-lambertian surfaces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(2), 1991.

[19] Y. Wu, S. S. Iyengar, R. Jain, and S. Bose. Shape from perspective trihedral angle constriant. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 261–266, 1993.

[20] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip H.S. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[21] Chaoqiang Zhao, Qiyu Sun, Chongzhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *https://arxiv.org/abs/2003.06620*, abs/2003.06620, 2020.