Construction of Stereo-Camera

As project elective under,

Prof.Dinesh Jayagopi

April 2015

By,

R.Pratesh Kumar Reddy

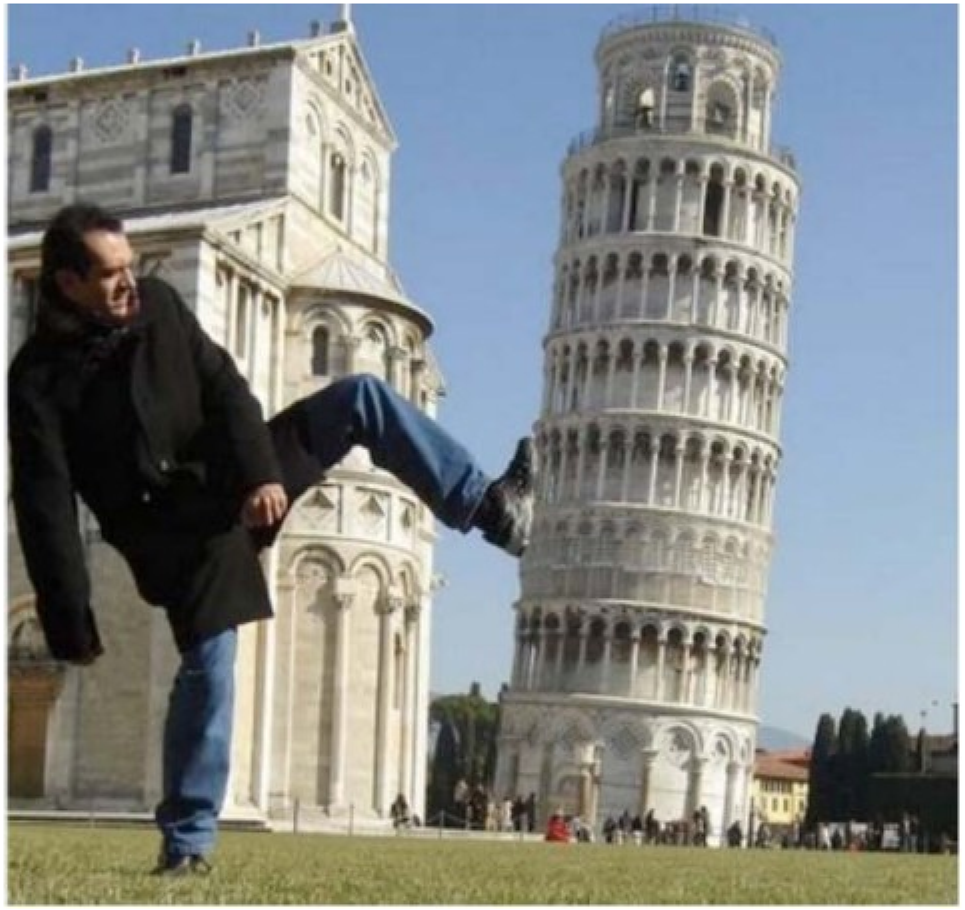
MT2014519

Contents

What is a stereo-camera?

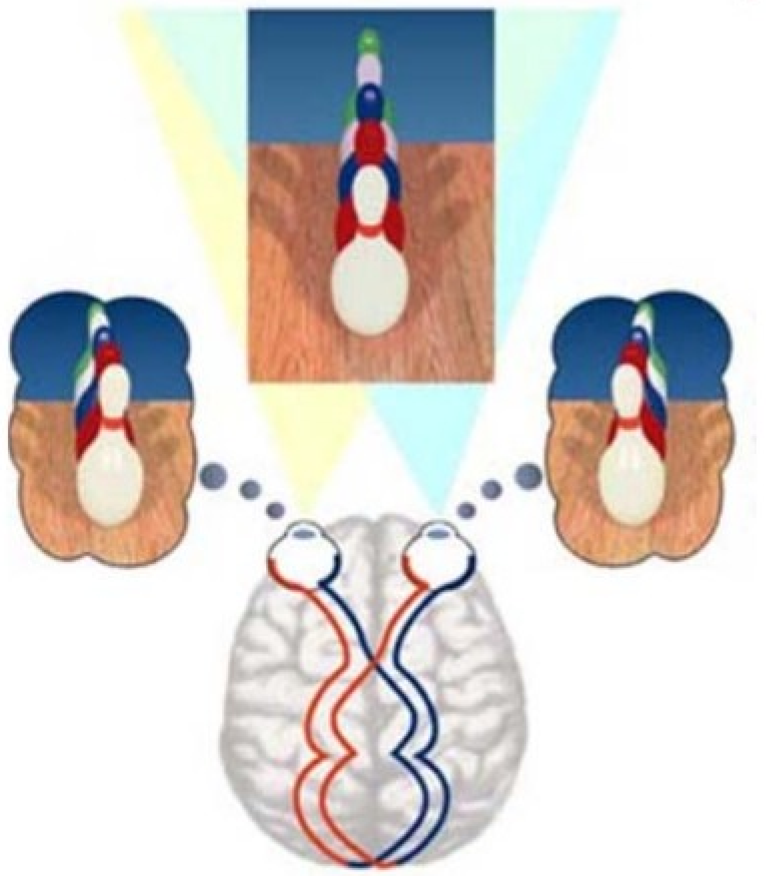
- It’s a camera with two or more lenses with a separate image sensor for each lens.
- The camera simulates human binocular vision and therefore has the ability to perceive depth.
- We can make our own stereo camera using two monocular cameras.

Advantage over monocular cameras

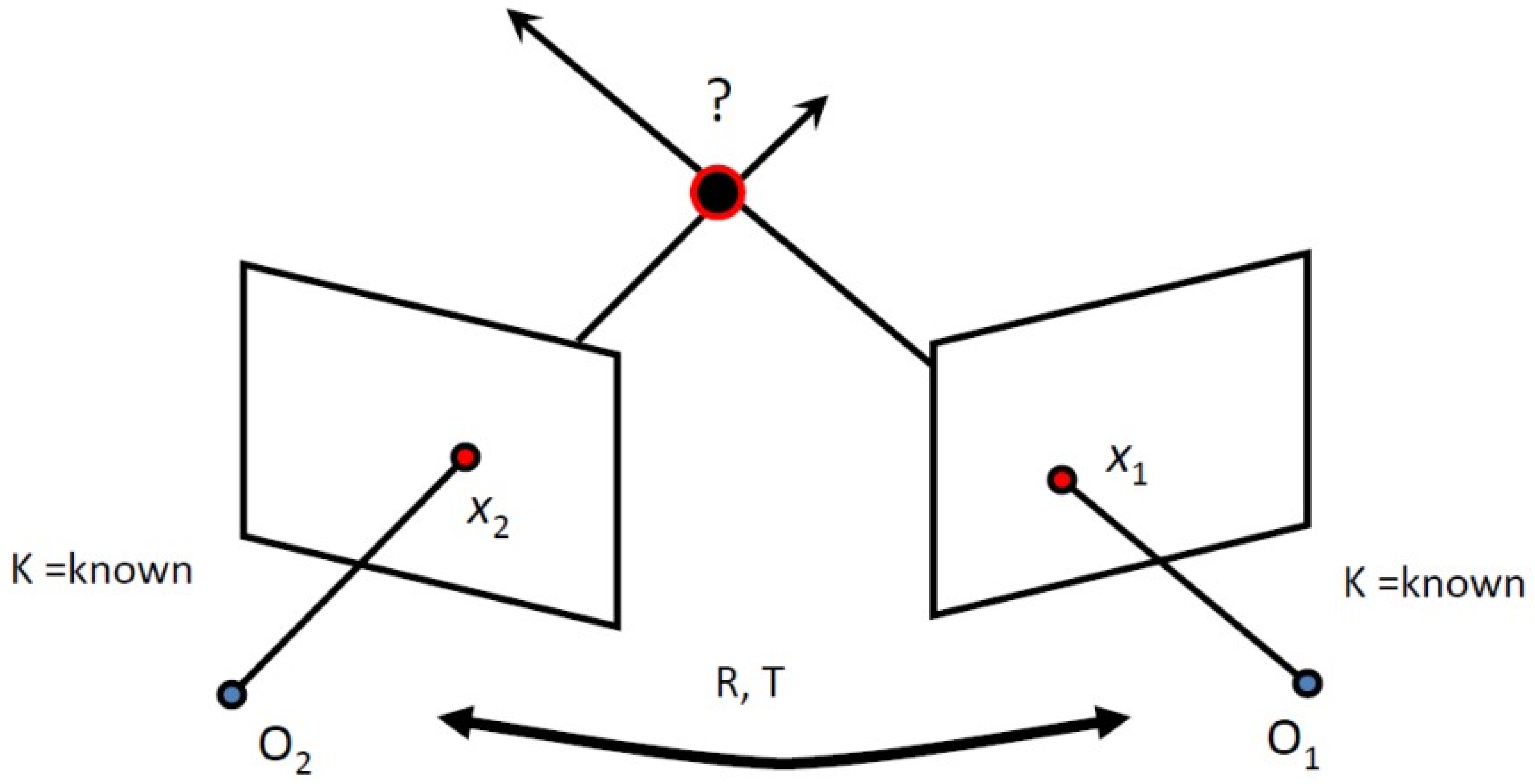


- Using monocular cameras we'll get a 2D image.
- But it will be ambiguous to determine "which object in the image is at what depth?"
- That's when the two eyes help.

Two eyes help!



Basic idea



This is called triangulation

What’s the project about?

- Construct a stereo-vision camera using two monocular cameras.
- Calibrate the camera pair both individually and as a stereo-pair and extract the intrinsic and extrinsic parameters.
- Produce a dense disparity map and thus the depth map.

Motivation behind this project

- To use it any of the future college projects that require depth sensing.
- Mainly to use them in Autonomous Navigation Robots.
- Can be part of Driver Assistance Systems.
- Substitute to Microsoft Kinect because OpenNI is no more open source.
- Can be used to sense human actions and thus model human behavior.

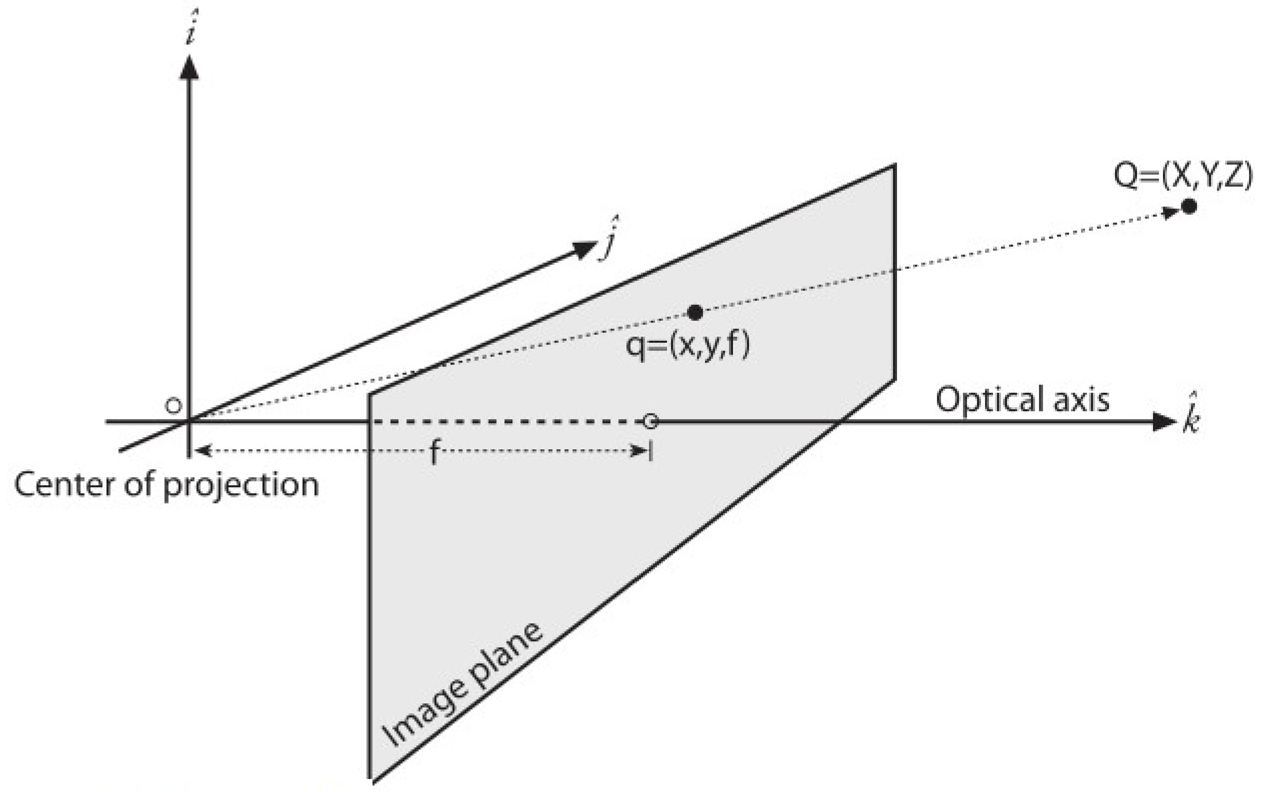
Prerequisites to build a stereo-camera

- Basic Linear Algebra, Rotation and translation
- Pinhole model of camera, Homogenous Coordinates
- Epipolar Geometry
- Image matching methods

- Material required:
  - Two almost similar cameras.
  - Stereo-calibration software tools (OpenCV/MATLAB/LabView can be used)
  - Calibration object

Basics: Image formation

- Camera model (Pin-hole model)



$$x = f * (X/Z) \qquad y = f * (Y/Z)$$

Basics: Image formation

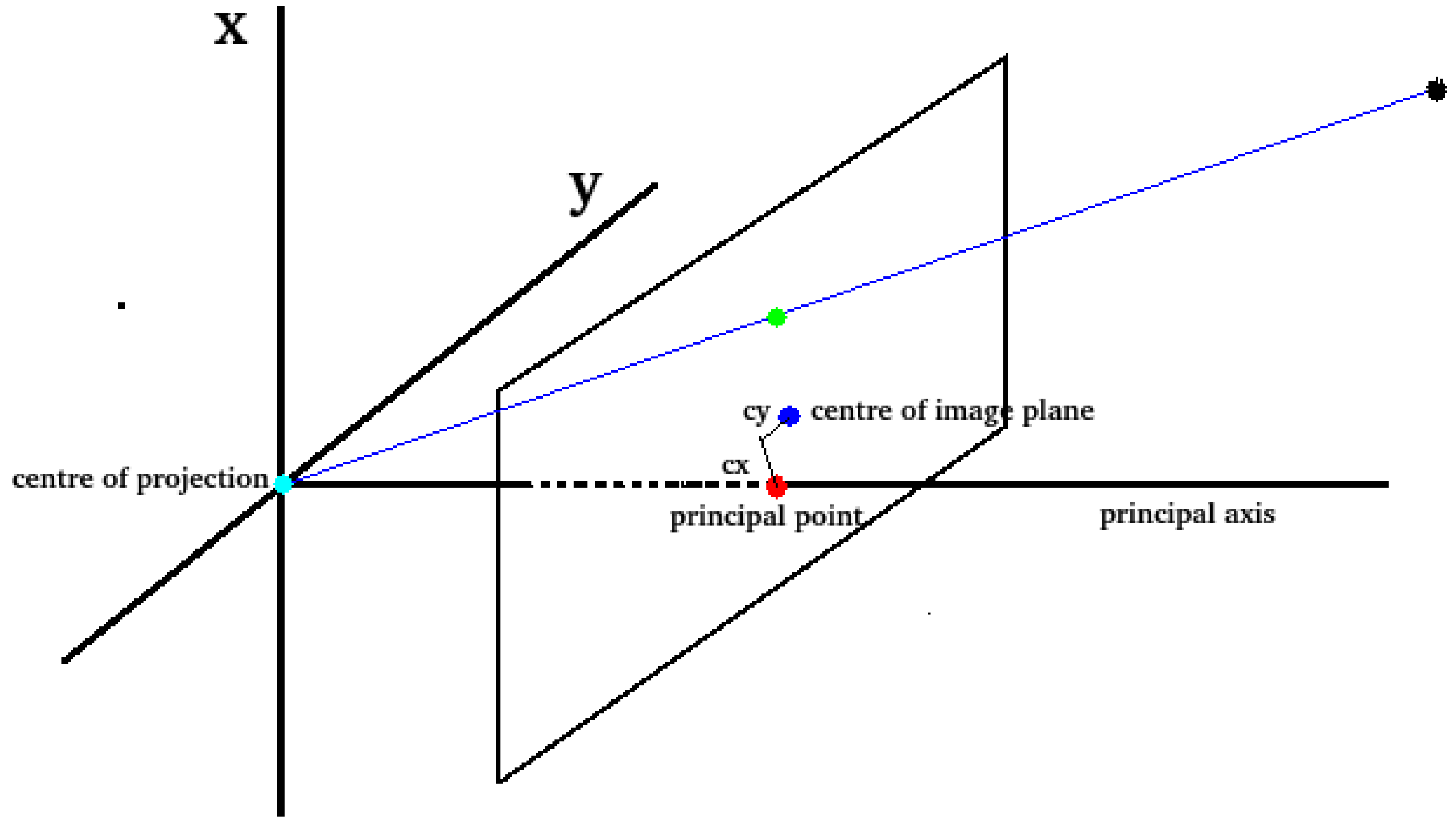
- Camera model
  - Due to displacement in center of image plane w.r.t to principal axis

$$x_{im} = f * (X/Z) + c_x * s_x$$

$$y_{im} = f * (Y/Z) + c_y * s_y$$

$$\mathbf{x_{im} = f_x * (X/Z) + c_x}$$

$$\mathbf{y_{im} = f_y * (Y/Z) + c_y}$$



Basics: Image formation

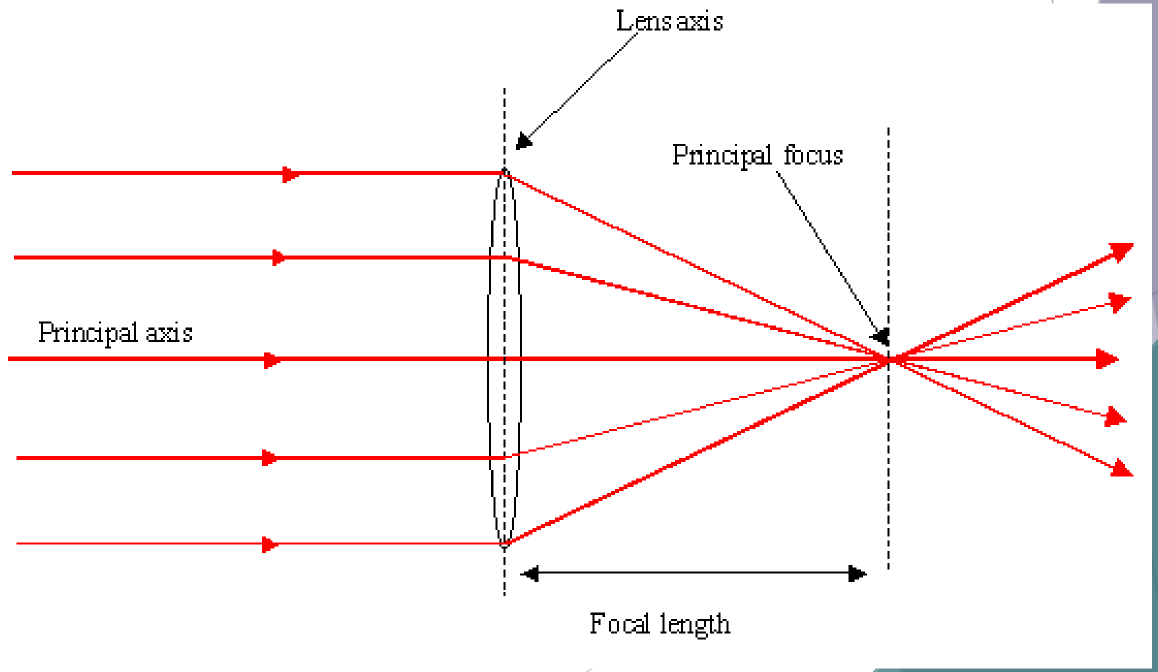
- Camera model
  - Finally in matrix form it is represented as

Intrinsic matrix

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

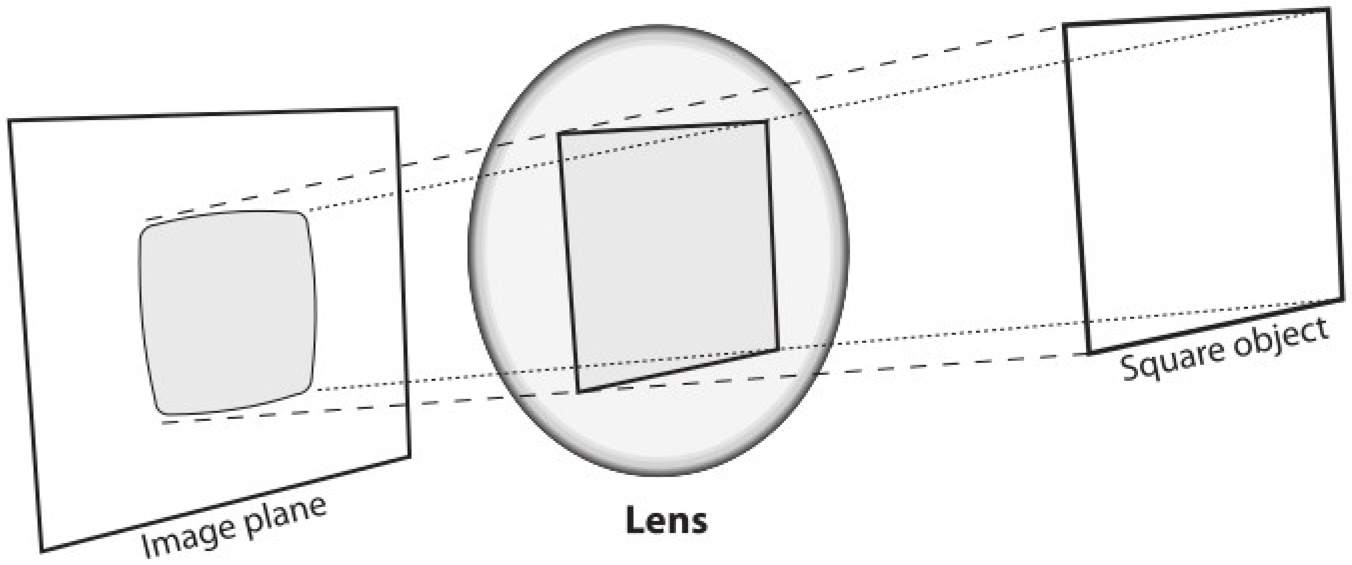
Basics: Distortions

- What ever is explained till now is an ideal pinhole model of camera.
- So such cameras will just form very small and less bright images because very little light goes through the pin-hole.
- To form a good image, we must gather a lot of light over a wider area and focus that light to converge at the point of projection.
- To accomplish this we use the **lens.**
- **So practical camera has both lens and image plane.**
- But since lenses are not perfect they introduce some **distortions.**



Basics: Distortions

- Radial distortion : because of the shape of the lens the pixels near the edges of the imager are more distorted.



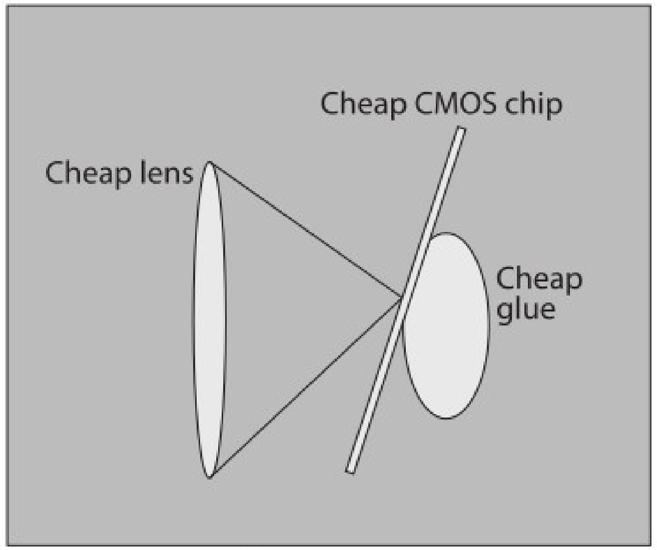
- Modeled as :

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

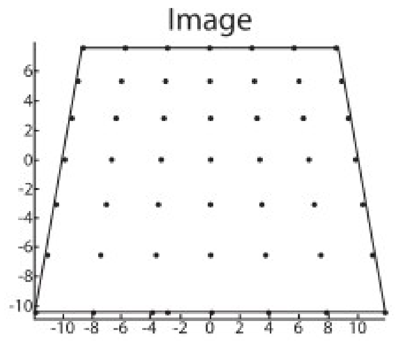
$$y_{\text{corrected}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Basics: Distortions

- Tangential distortion : when lens is not exactly parallel to the image plane.



Cheap camera



- Modeled as :

$$x_{\text{corrected}} = x + [2p_1 y + p_2(r^2 + 2x^2)]$$
$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2 x]$$

[Ref:7]

Basics: Distortions

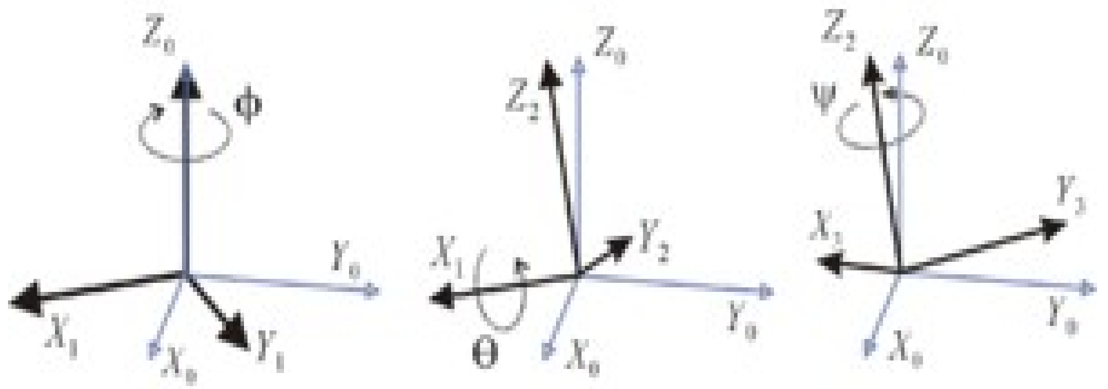
- Final distortion coefficients vector (as used in OpenCV) :

$$(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]])$$

Basics: Extrinsic Parameters

- Rotation Matrix and Translation Vector
  - If we rotate around the x-, y-, and z-axes in sequence with respective rotation angles ψ, φ, and θ, the result is a total rotation matrix R that is given by the product of the three matrices **R x(ψ), Ry(φ), and Rz(θ**), where

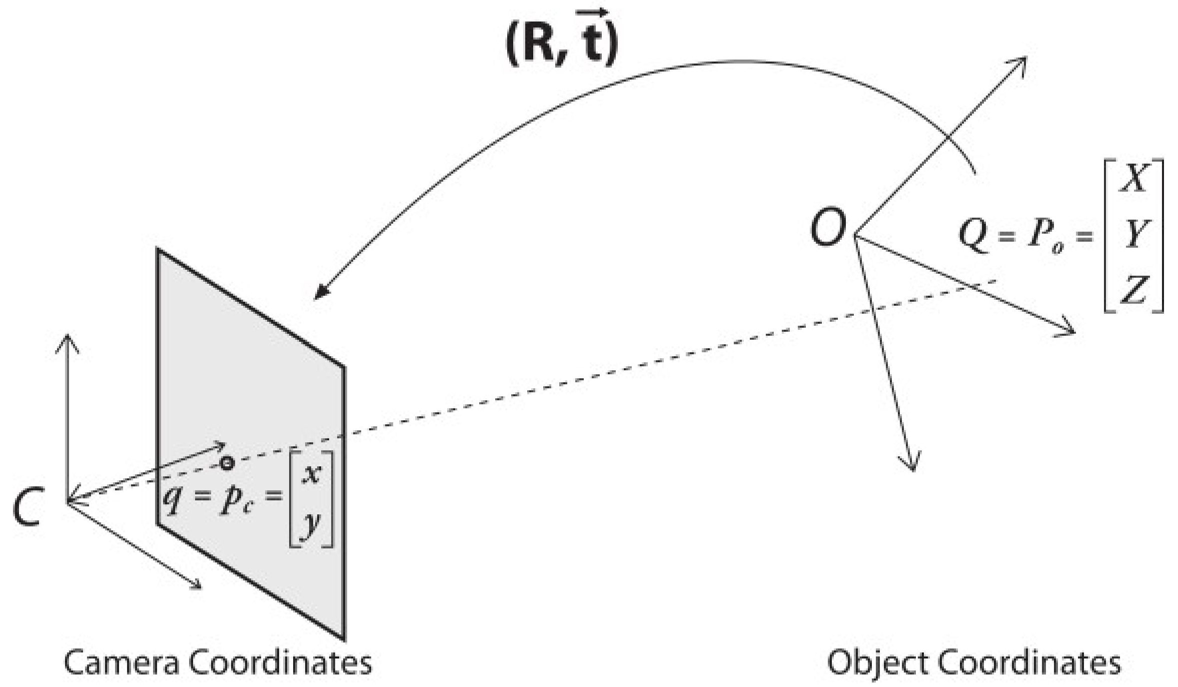
$$R_x(\psi)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\psi & \sin\psi\\ 0 & -\sin\psi & \cos\psi\end{bmatrix} \qquad R_y(\varphi)=\begin{bmatrix}\cos\varphi & 0 & -\sin\varphi\\ 0 & 1 & 0\\ \sin\varphi & 0 & \cos\varphi\end{bmatrix} \qquad R_z(\theta)=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$



Basics: Extrinsic Parameters

- Translation Vector

$$W = [R \mid t]$$



$(\mathbf{R}, \vec{\mathbf{t}})$
O
$Q = P_o = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$
C
$q = p_c = \begin{bmatrix} x \\ y \end{bmatrix}$
Camera Coordinates
Object Coordinates

All Combined together

$$\tilde{q} = sMW\tilde{Q}, \quad \text{where} \quad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad W = [R \mid t]$$

where

$$\tilde{Q} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$$
$$\tilde{q} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$$

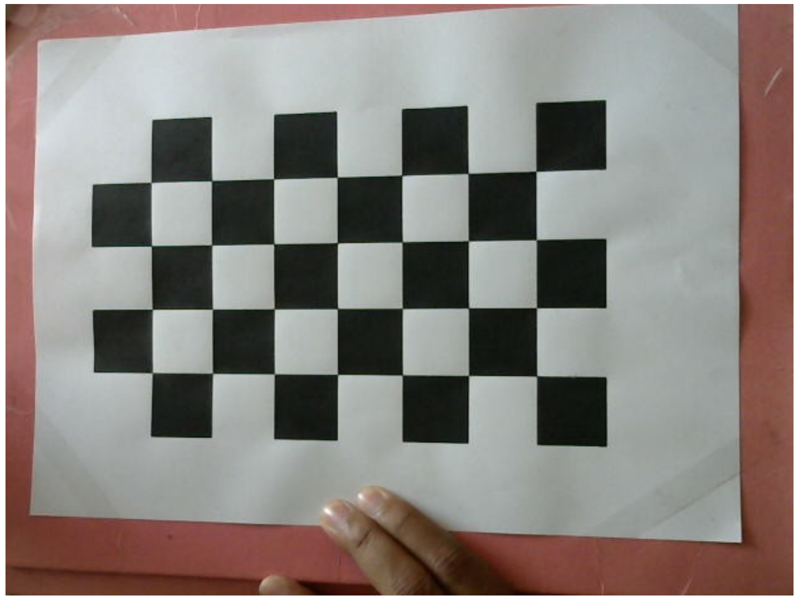
The whole aim of the camera calibration is thus find out this MW matrix and then extract the M and W individually.

Step1: Calibration Object

- Idea:Using a set of known correspondences between point features in the world (X, Y, Z) and their projections on the image (x, y).

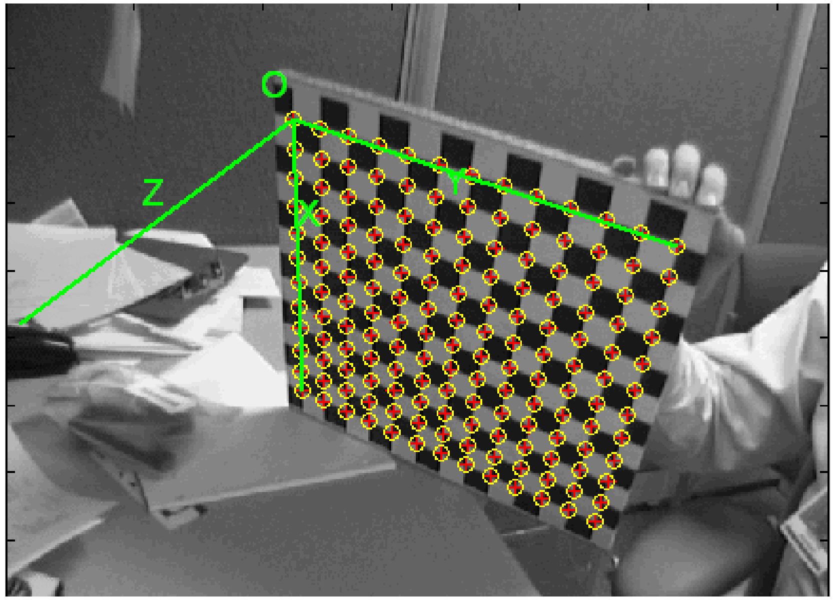
- For that we have to use a calibration object

We have used the planar chess board as the calibration object



Step2:Finding out the world coordinates

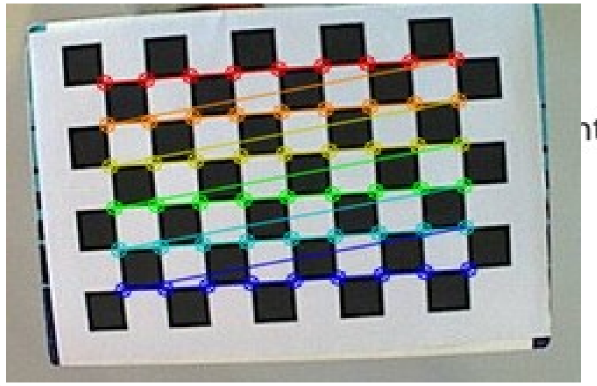
- The world coordinate system will be attached along one of the edges of the chess board.
- The no. of inner corners along the width and height are known.
- The physical size of each block is known.
- So we know the world coordinates of the each of the chessboard corner.
- The z-coordinate can be taken as zero or any constant.
- Thus we know the Q vectors in the equation

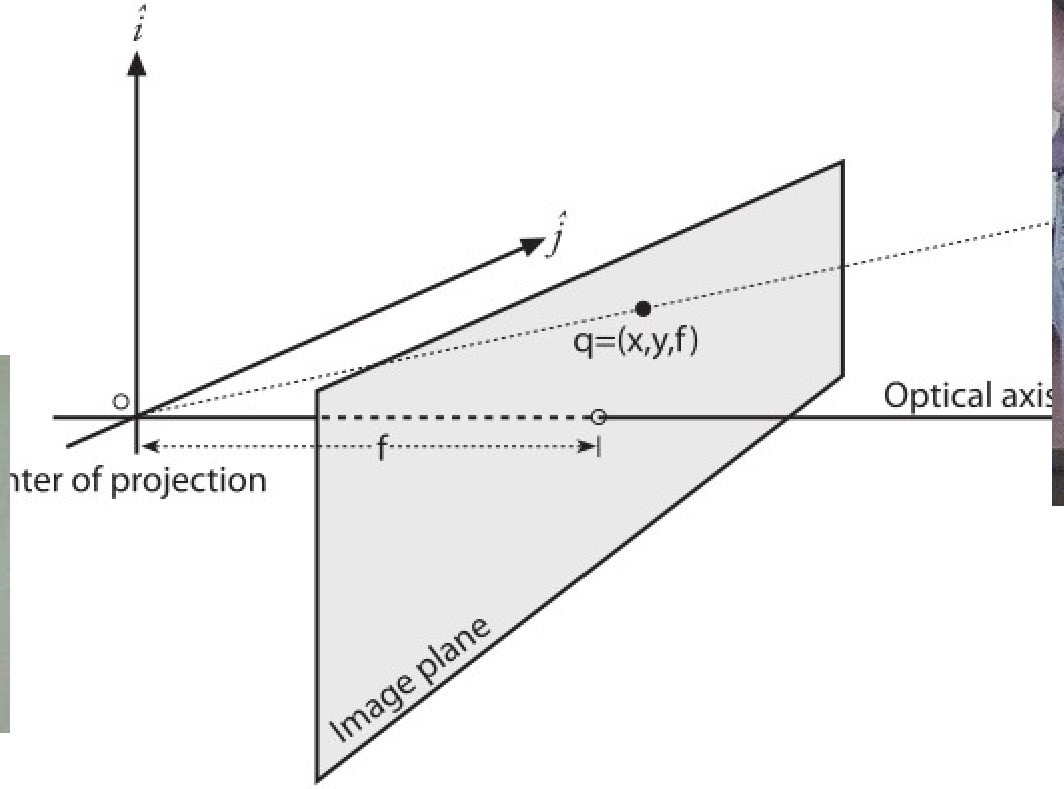


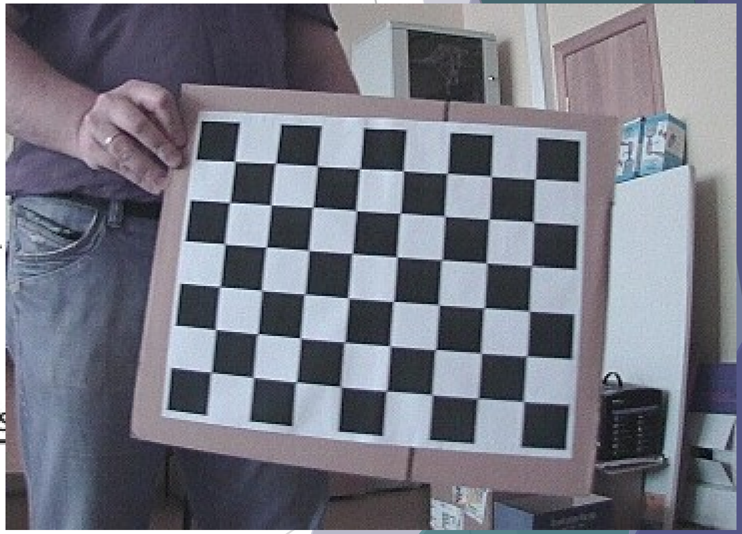
$$\tilde{q} = sMW\tilde{Q};$$

Step3:Finding out the image coordinates

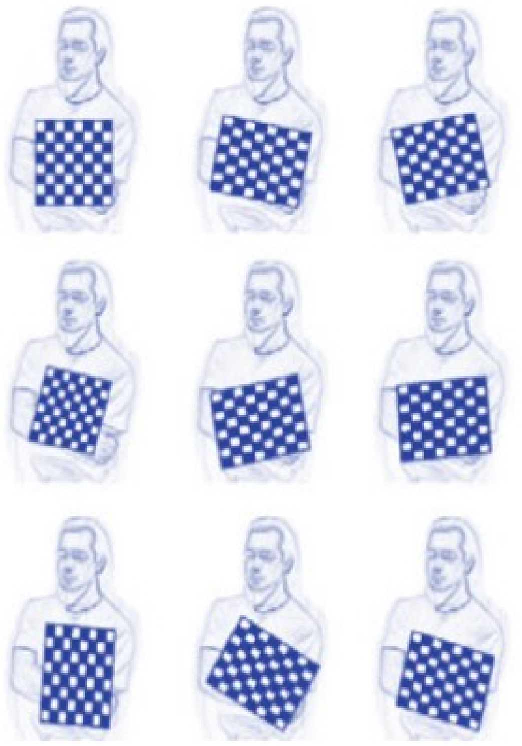
- Use the OpenCV function cvFindChessboardCorners() to locate the corner pixels in the chess board images.
- Now we got the q vectors in equation: $\tilde{q} = sMW\tilde{Q}$;

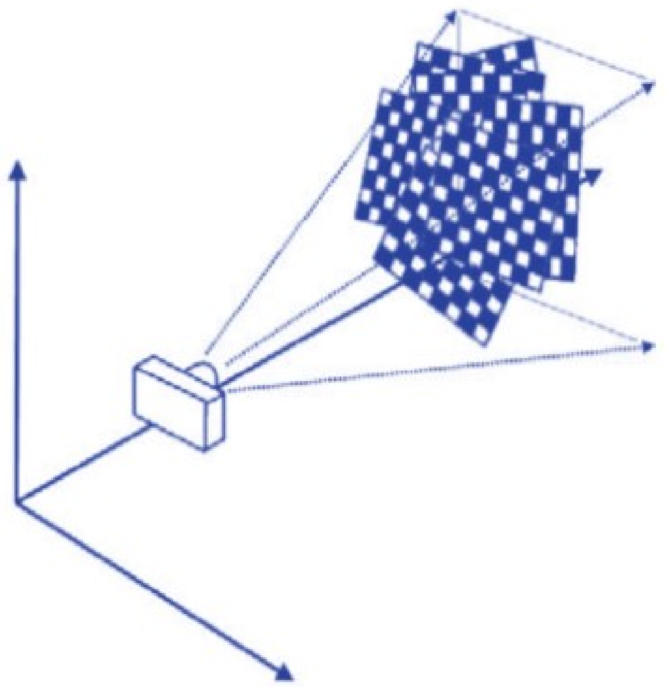






Step5:Chessboards in different orientation





- Even-though 4 points per image and two images in different orientations are enough to calibrate the camera, we usually take 10~15 images to compensate noise and other inconsistencies.

Step6:Final step

- To solve the equations that we got from the world-image correspondences.

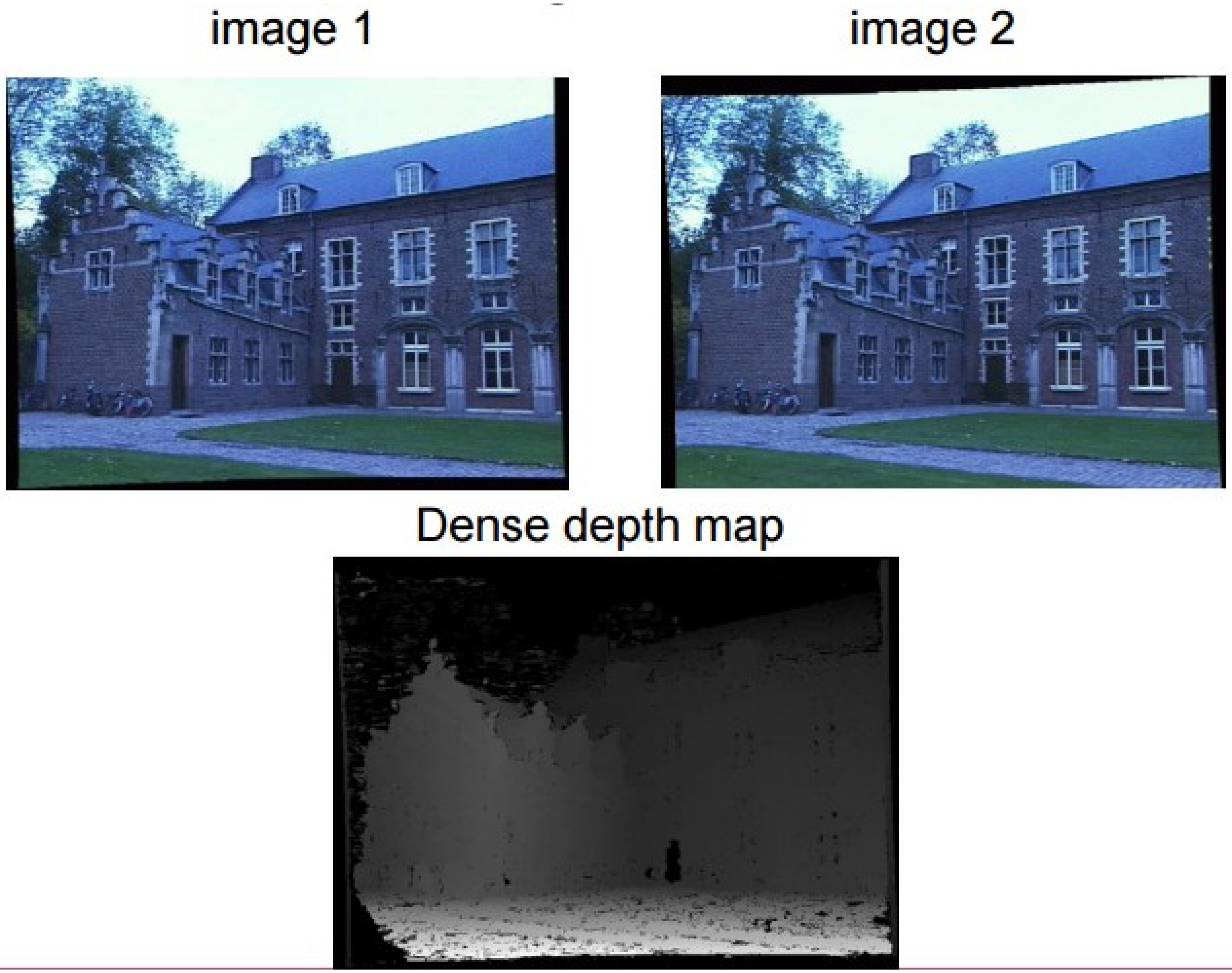
- For this we use cvCalibrateCamera2() function which does all the work we need.

  - This function takes in the object and image coordinates as inputs.
  - Gives out the intrinsic matrix , distortion coefficients of the camera.
  - Gives out the rotation and translation vectors for each orientation.

- Most of the samples codes in OpenCV generally neglect the tangential distortions.
- Refer Learning OpenCV (1) book for mathematics inside this function.

Single camera calibration in over :)

Step 7: Moving to the Stereo-Camera



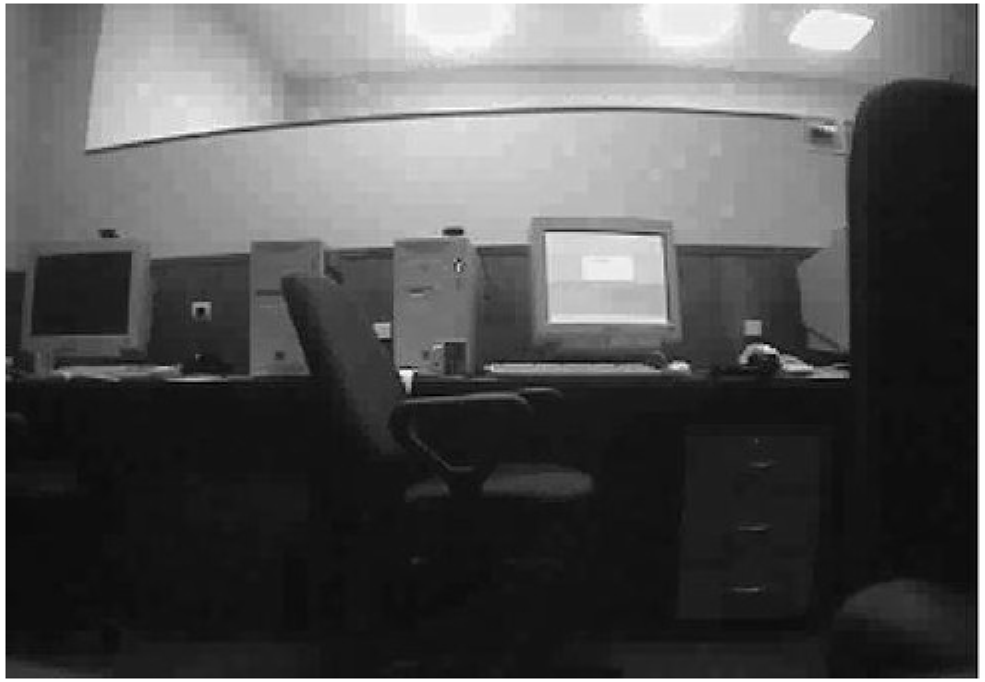
In stereo vision, features in two (or more) images taken at the same time from separate cameras are matched with the corresponding features in the other images, and the differences are analyzed to yield depth information

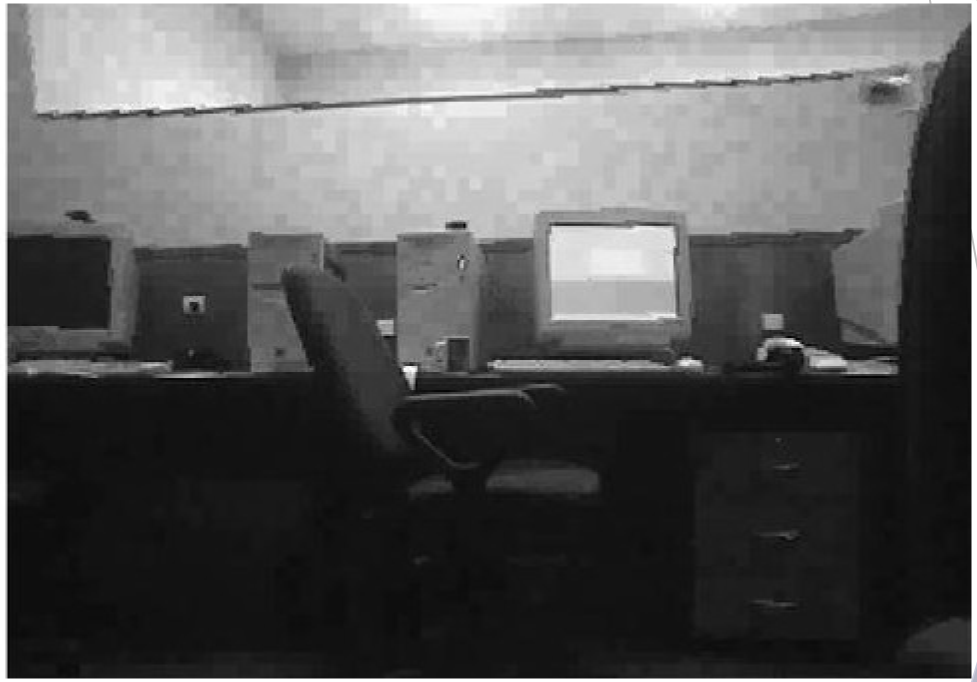
Step 7: Moving to Stereo-Camera

- Here we have two cameras mounted on flat surface and separated by some horizontal displacement called as stereo-rig.
- In practice, stereo-imaging involves following steps:

  - Find out the intrinsic and distortion parameters of each camera.
  - Find out the rotation and translation vector in between the two cameras.
  - Undistortion: To Remove the radial (and tangential) lens distortions.
  - Rectification: Adjust the angles and distances between cameras to get row-alignment.
  - Correspondence: To find the same features in the left and right camera views.
  - Find out the difference in pixels' x-coordinates to get the disparity value.

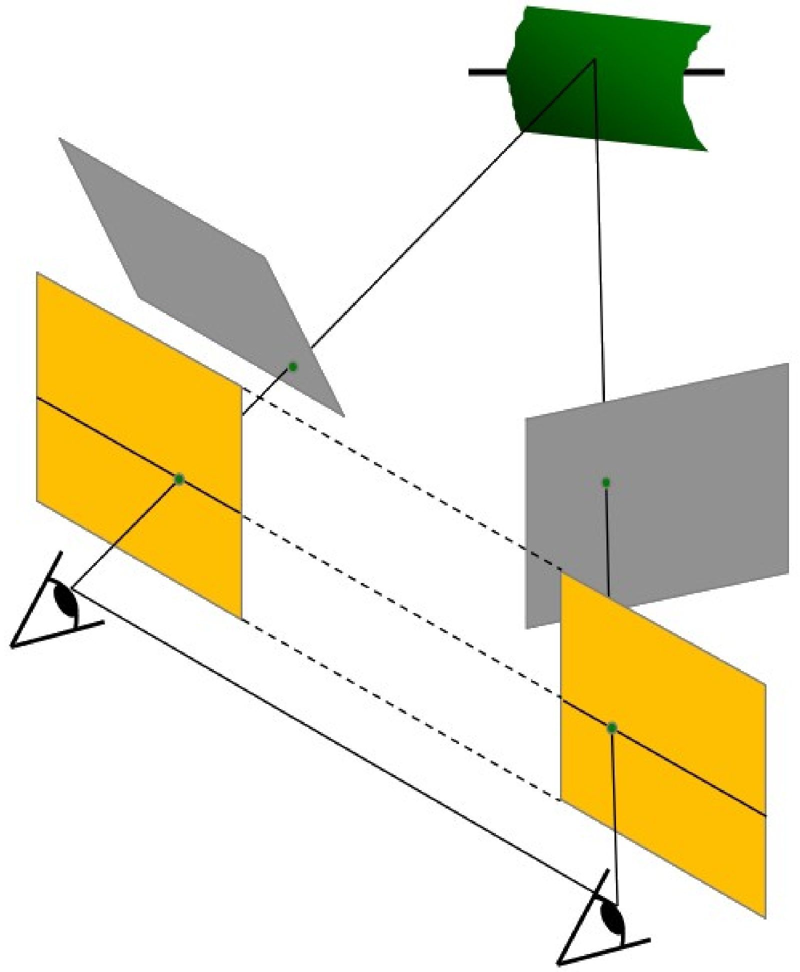
Step8: Undistortion





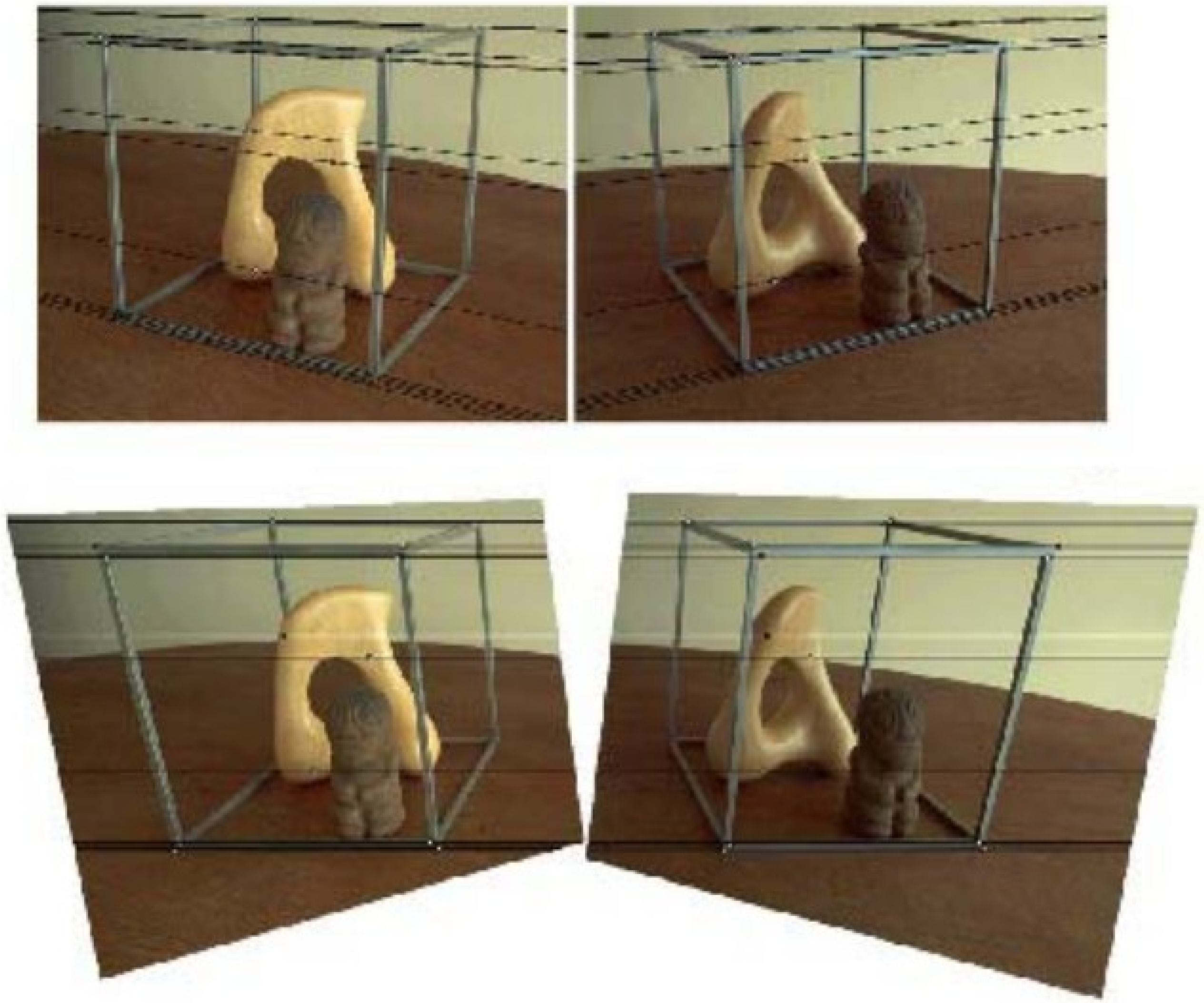
- To compensate the lens distortion effects.
- Can be done usign undistortpoints() function in OpenCV.

Step9: Rectification



The goal will be to mathematically (rather than physically) align the two cameras into one viewing plane so that pixel rows between the cameras are exactly aligned with each other

Step9: Rectification



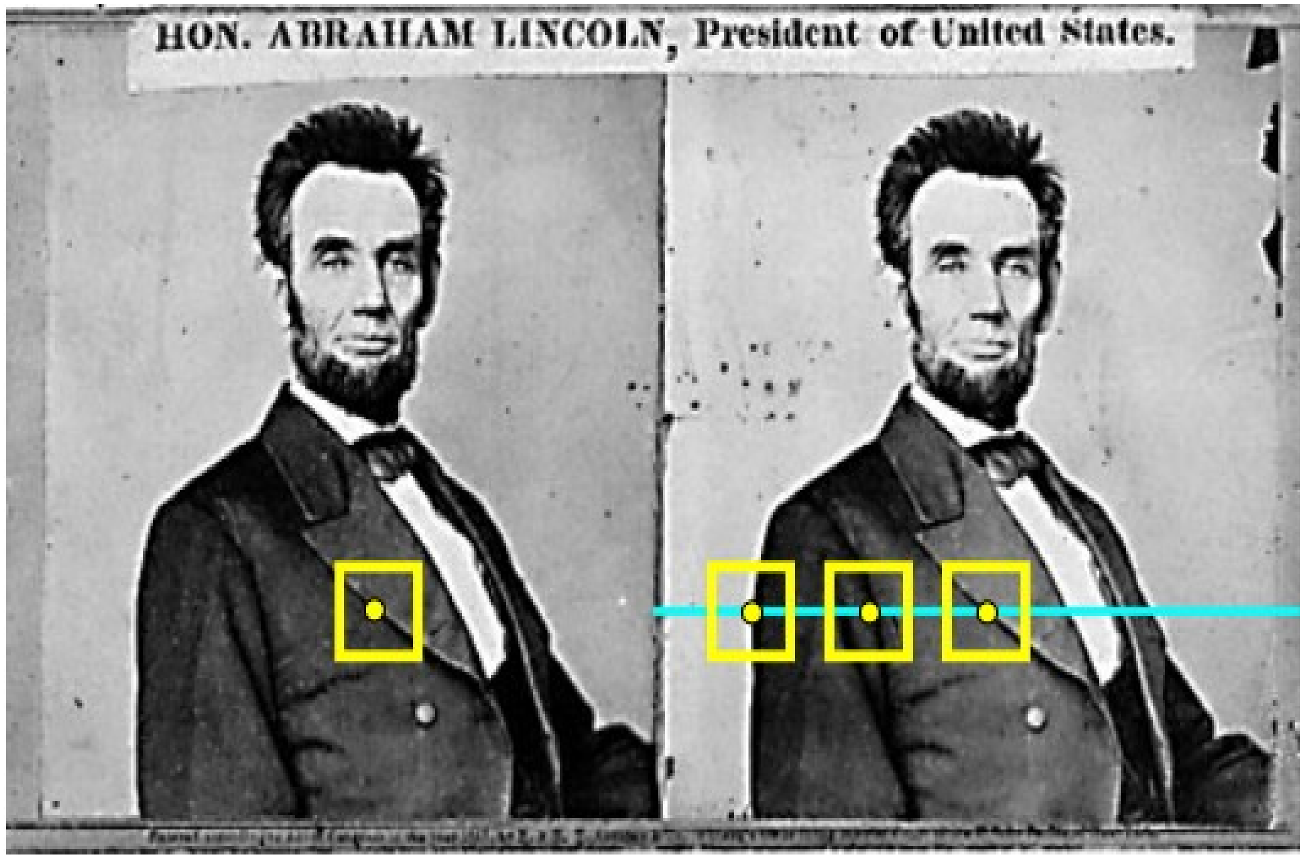
Step9: Rectification

The mathematics behind the stereo-rectification involves the concept of Epipolar Geometry and Constraints.

Using that constraints we get the Fundamental matrix and essential matrix that guide us in “how to rotate the image planes for row alignment”.

Maths under the hood can again be found in “Learning OpenCV” book by Bradski.

Step10: Stereo-matching



For each pixel in the first (left camera's) image

- Find corresponding epipolar line (in this case it's the same row) in the right image
- Examine all pixels on the row and pick the best match.

Step10: Stereo-matching

How to pick the best match ??

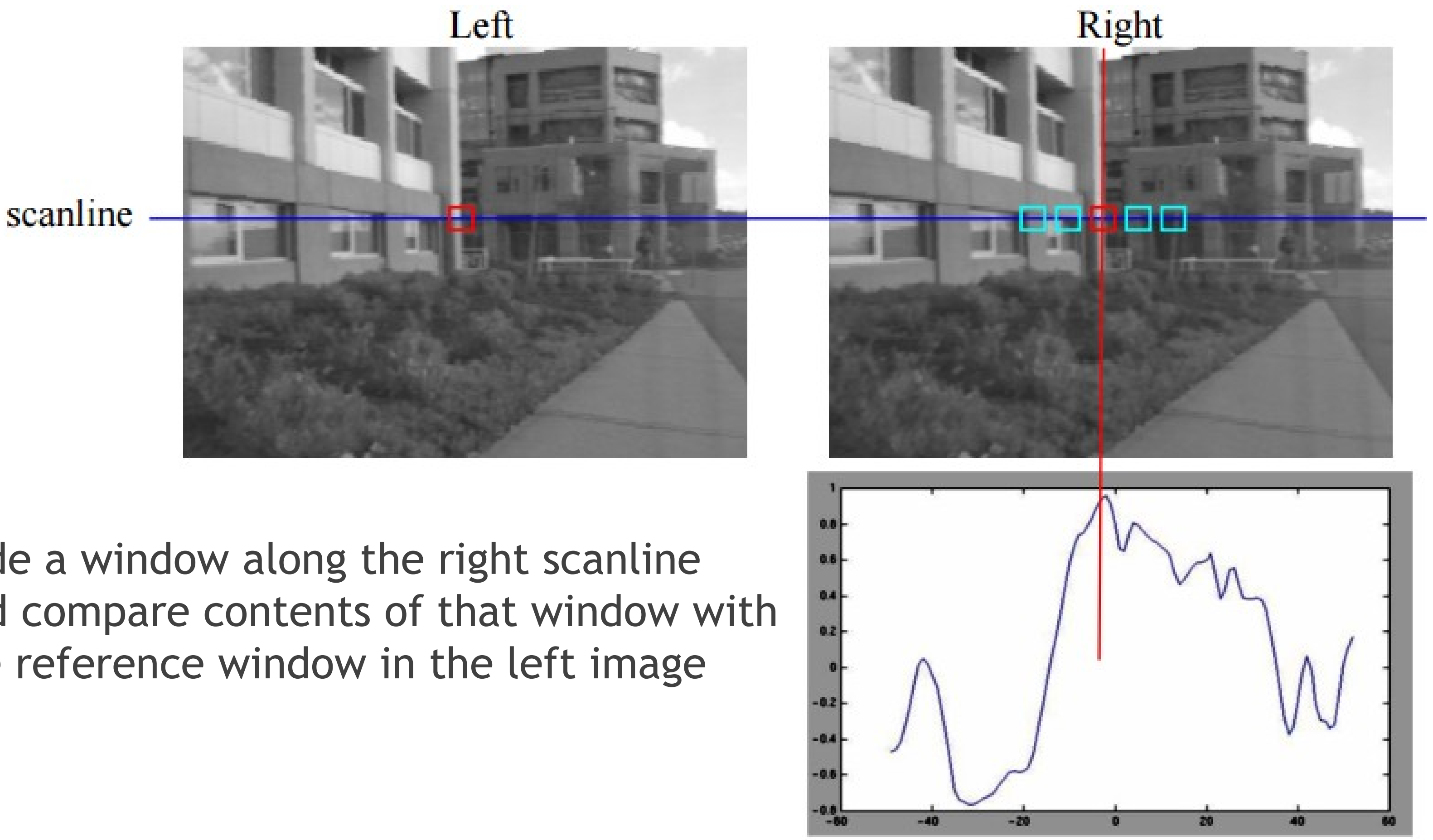
- This is called as **stereo-correspondence** problem.

There are many proposed methods to find the correspondence like

- Block-matching
- Semi-global block matching
- Census transform based matching
- Variance based matching

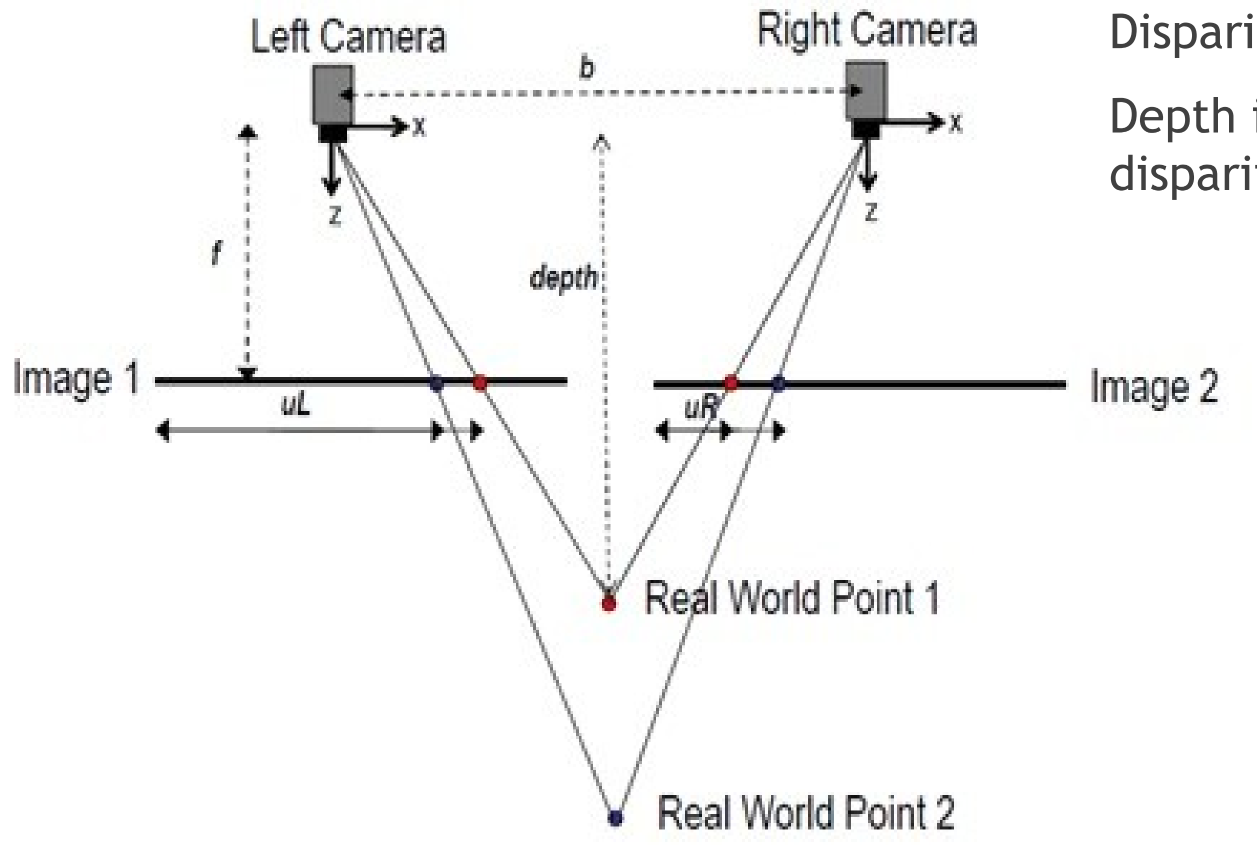
Step10: Stereo-matching

General Algorithm for finding Correspondence



Slide a window along the right scanline and compare contents of that window with the reference window in the left image

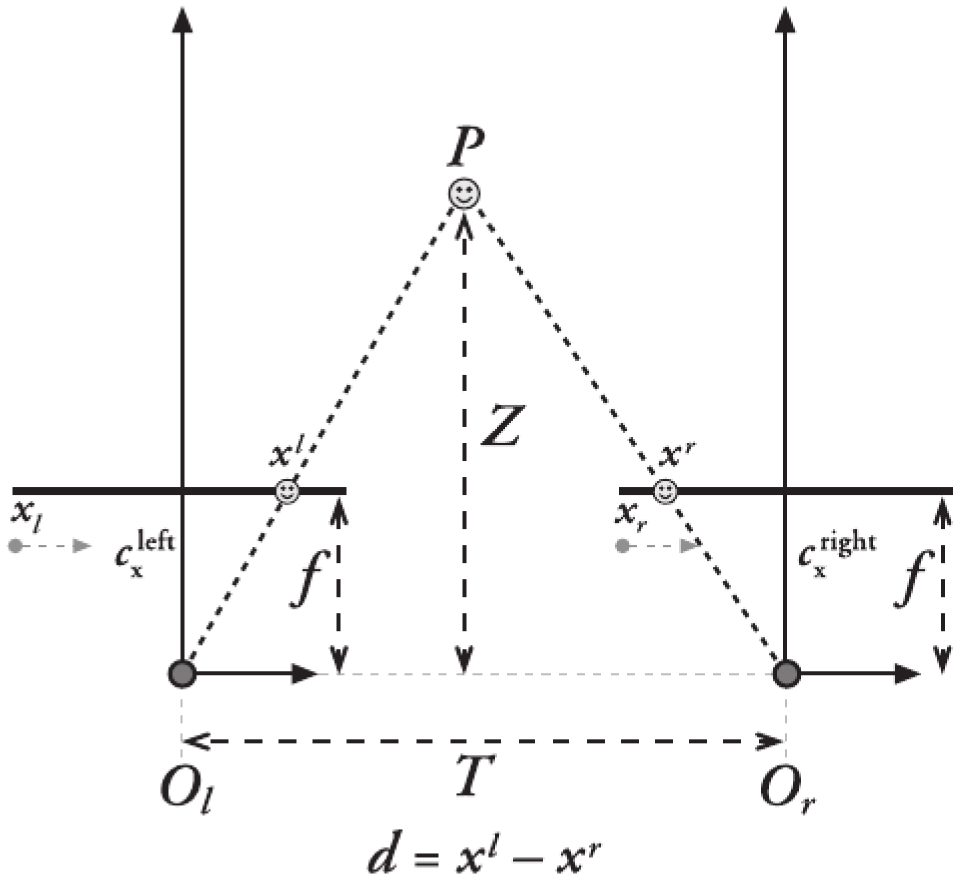
Step11: Calculate Disparity



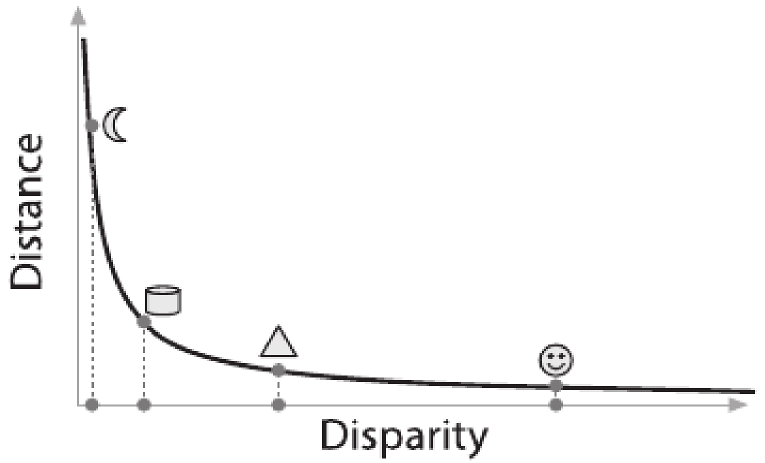
Disparity= uL-uR

Depth is inversely proportional to the disparity between these views

Step12: Depth by triangulation



$$\frac{T-(x^l - x^r)}{Z-f} = \frac{T}{Z} \quad \Rightarrow \quad Z = \frac{fT}{x^l - x^r}$$



Step12: Depth by triangulation

- When disparity is near 0, small disparity differences make for large depth differences.
- When disparity is large, small disparity differences do not change the depth by much.
- The consequence is that stereo vision systems have **high depth resolution only for objects relatively near the camera.**

Implementation

- The whole procedure is being implemented using the OpenCV computer-vision library in a Linux (Ubuntu) machine.
- Two commercial USB cameras (Logitech C270) are used.
- Take and store the pictures of a calibration object (like Chessboard) using both camera.
- Then use the following OpenCV functions:
  - *findChessboardCorners(): to locate the chessboard corners on the images.*
  - *drawChessboardCorners() : to re-draw the located chessboard corners on the chessboard images*
  - *cornerSubPix() : to locate the chessboard corners with sub-pixel accuracy*
  - *stereoCalibrate() : to find out the camera & distortion matrices of both cameras and rotation, translation, Essential, Fundamental matrices between the two cameras.*

Implementation (Contd.)

- The matrices thus obtained are used further to
  - *Transform the image points to compensate for lens distortion by using undistortPoints()*
  - *compute the corresponding epilines in an image for the points in another image of a stereo pair using computeCorrespondEpilines()*
  - *Get the respective rectification (R1 & R2) and projection matrices (P1 & P2) and disparity to depth map (Q) using stereoRectify() function*.
- The rectification and projection matrices thus obtained are passed to
  - *initUndistortRetifyMap() to calculate the undistortion and rectification transformation matrix for each camera head*.
- These transformation matrices are given to remap() function to do inverse mapping and get the final corrected and rectified image.

DEMO

Future work

- Implement the stereo-vision algorithm on hardware using either
  - *Digital Signal Processor*
  - *FPGA*
  - *Other General Purpose processors*
  - *Combination any of the above*
- The aim is to able to use the stereo camera at such a frame rates in real-time that it can be used in Advanced Driver Assistance Systems and autonomous navigation.

References (in the order of their utility)

1) Learning OpenCV by Bradski. (Chapter 11 & 12)
2) Stereo Vision (Lecture 7) by Lin Zhang , Tongji University
3) A flexible new technique for Camera Calibration by Zhang.
4) Emerging Topics in Computer Vision by Midioni and Kang (Chapter 2)
5) Camera Calibration Lectures by Prof.B.K.Biswas from NPTEL (IIT Kgp)
6) FPGA design and implementation of a Real-time Stereo Vision by Jin et.al.
7) De centering Distortion of Lenses by D.C.Brown
8) Wikipedia articles on Image-rectification, fundamental matrix

Questions ?

Thank You

Thankyou

- Just explain the steps followed
- Combine all the fucntuionalieties f Opencv functions without mentioning the fucntions itself.