Computer Vision

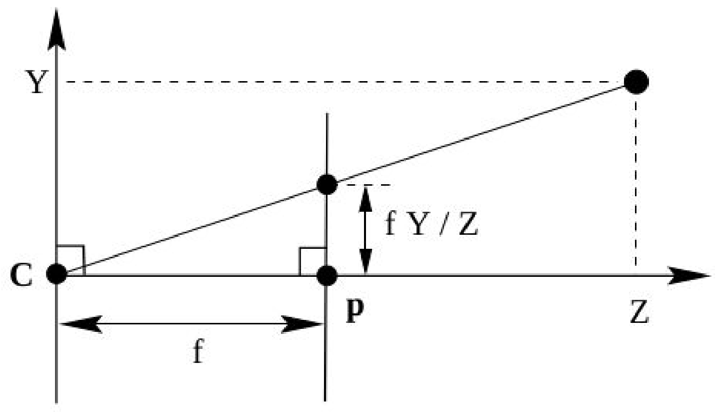
Computer vision is an interdisciplinary scientific field that deals with how **computers** can be made to gain high-level understanding from digital images or videos.

-> Video1

Calculation of the Camera Matrix

$$x = PX$$

Basic Pinhole Model



$$(X,\ Y,\ Z)^T \Rightarrow \left(f\,X/Z,\ f\,Y/Z\right)^T$$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}.$$

$$P = diag(f,\ f,\ 1)\left[I \mid 0\right]$$

Principal Point Offset

$$(X,\ Y,\ Z)^T \Rightarrow \left(fX/Z + p_x,\ fY/Z + p_y\right)^T$$

$$x = K\left[I \mid 0\right] X_{cam}$$

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

Camera Rotation and Translation

$$\tilde{X}_{cam} = R(\tilde{X} - \tilde{C})$$

$$x = K\,R\left[I \mid -\tilde{C}\right] X$$

$P = K\left[R \mid t\right]$, where $t = -R\tilde{C}$

Finite Projective Camera

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

Where $\alpha_x,\ \alpha_y,\ x_0,\ y_0$ are same quantities measured in pixel dimensions.
and *s* is the skew parameter.

Distortion

2 types-

1. **Radial**: Straight lines appear curved.

$$x_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$
$$y_{corrected} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

2. **Tangential**: Occurs because image taking lens is not aligned perfectly parallel to the imaging plane. So some areas in image may look nearer than expected.

$$x_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

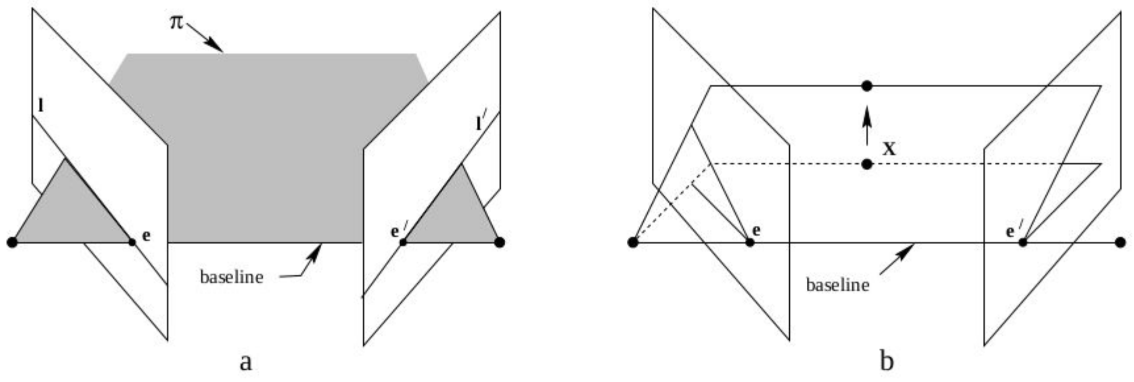
So, 5 parameters are needed, known as the **distortion coefficients**.

$$Distortion\ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

Epipolar Geometry and the Fundamental Matrix

Epipolar Geometry

The epipolar geometry between two views is essentially the geometry of the intersection of the image planes with the pencil of planes having the baseline as axis (the baseline is the line joining the camera centres).



Definitions :-

1. The **epipole** is the point of intersection of the line joining the camera centres (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera centre of the other view.
2. An **epipolar plane** is a plane containing the baseline. There is a one-parameter family (a pencil) of epipolar planes.

3. An **epipolar line** is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

Fundamental Matrix

The fundamental matrix(F) is the algebraic representation of epipolar geometry.

For an image point x and its corresponding point in the other image, x' ,

$$x'^{T}Fx = 0$$

Properties :-

1. F is a rank 2 homogeneous matrix with 7 degrees of freedom.
2. **Epipolar Lines** - $l' = Fx$ is the epipolar line corresponding to x.
   $l = Fx'$ is the epipolar line corresponding to x'.
3. **Epipoles** - $Fe = 0,\ F^{T}e' = 0$
4. $F = [e']_{\times}P'P^{+}$, where P^{+} is the pseudo inverse of P, and $e' = P'C$ with $PC = 0$

Essential Matrix

The essential matrix(E) is the specialization of the fundamental matrix to the case of normalized image coordinates.
The essential matrix has fewer degrees of freedom, and additional properties, compared to the fundamental matrix.

$$\hat{x}'\,E\,\hat{x} = 0$$

Normalized image coordinates : $\hat{x} = [R \mid t]\,X$

Reconstruction of Cameras and Structure

Outline of reconstruction method:

(i) Compute the fundamental matrix from point correspondences.
(ii) Compute the camera matrices from the fundamental matrix.
(iii) For each point correspondence $x_i \leftrightarrow x_i'$, compute the point in space that projects to these two image points.

Computation of the fundamental matrix:

Given a set of correspondences $x_i \leftrightarrow x_i'$ in two images the fundamental matrix F satisfies the condition $x_i' F x_i = 0$ for all i. Withthe x_i *and* x_i' known, this equation is linear in the (unknown) entries of the matrix F. Infact, each point correspondence generates one linear equation in the entries of F. Given at least 8 point correspondences it is possible to solve linearly for the entries of F up to scale (a non-linear solution is available for 7 point correspondences).

Computation of the camera matrices:

A pair of camera matrices P and P' corresponding to the fundamental matrix F is easily computed using the direct formula as shown earlier.

Triangulation:

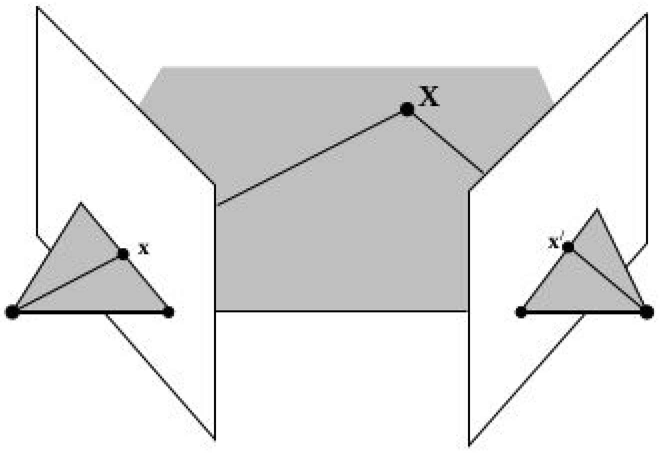
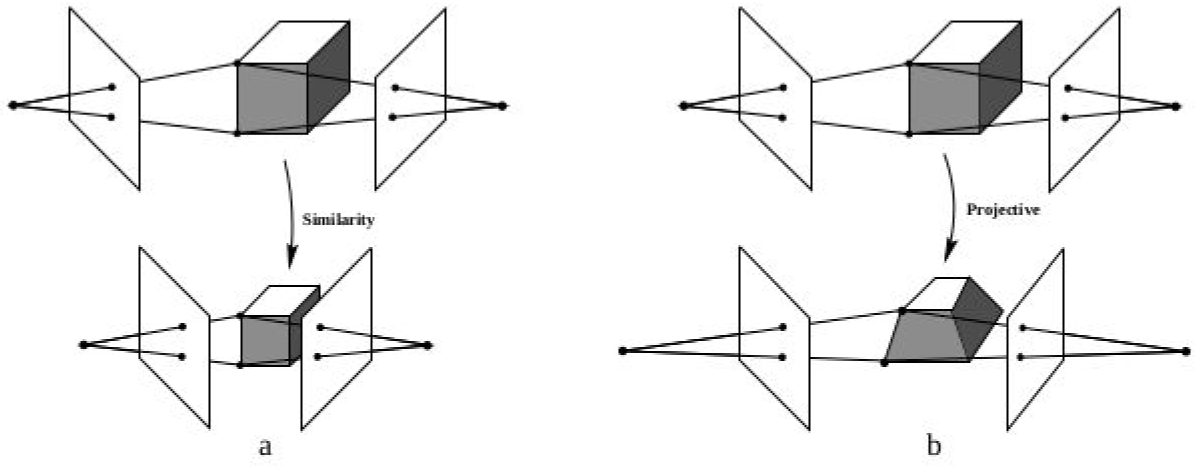


Fig. 1. Triangulation. The image points x and x' back project to rays. If the epipolar constraint $x'^T F x = 0$ is satisfied, then these two rays lie in a plane, and so intersect in a point X in 3-space.
The only points in 3-space that cannot be determined from their images are points on the baseline between the two cameras. In this case, the back-projected rays are collinear

(both being equal to the baseline) and intersect along their whole length.

Reconstruction ambiguity:



a) For calibrated cameras b) For uncalibrated cameras

Fig. 2.

Position ambiguity:

It is impossible based on the images alone to estimate the absolute location and pose of the scene w.r.t. a 3D world coordinate frame.

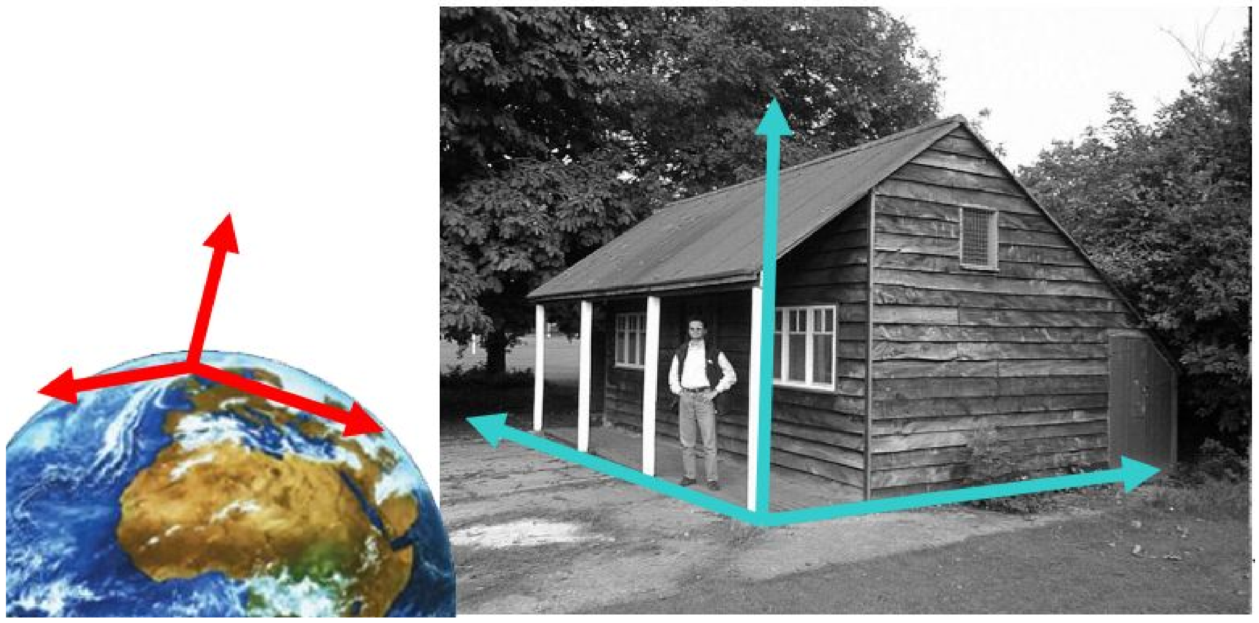


Fig.3. Position ambiguity

Scale ambiguity:

It is impossible based on the images alone to estimate the absolute scale of the scene (i.e. house height).

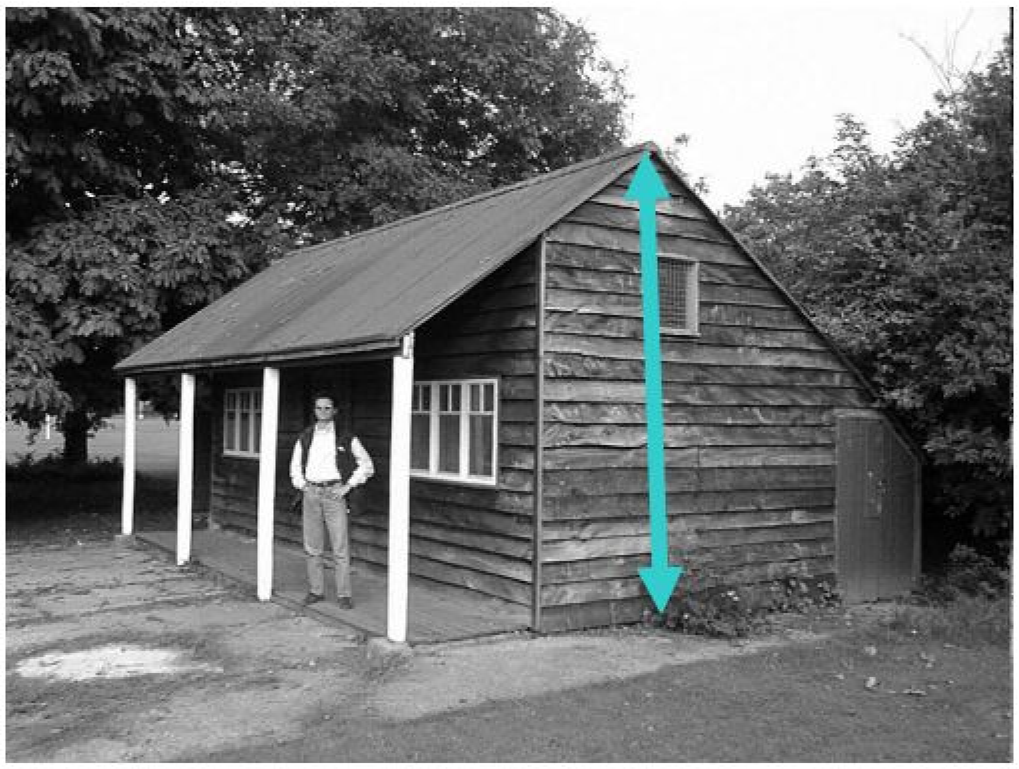


Fig. 4. Scale ambiguity

The projective reconstruction theorem:

We can compute a projective reconstruction of a scene from 2 views based on image correspondences alone.We don't have to know anything about the calibration or poses of the cameras.
Assume we determine matching points xi and x'i. Then we can compute a unique fundamental matrix F. The recovered camera matrices are not unique: (P1, P'1), (P2, P'2), etc. Hence the reconstruction is not unique: $X_{1i}, X_{2i},$ etc. There exists a projective transformation H such that $X_{2i} = HX_{1i},\ P_2 = P_1 H^{-1}, P_2' = P_1' H^{-1}$
and $P_2 X_2 = P_1 H^{-1} X_2 = P_1 H^{-1} H X_1 = P_1 X_1 = x$

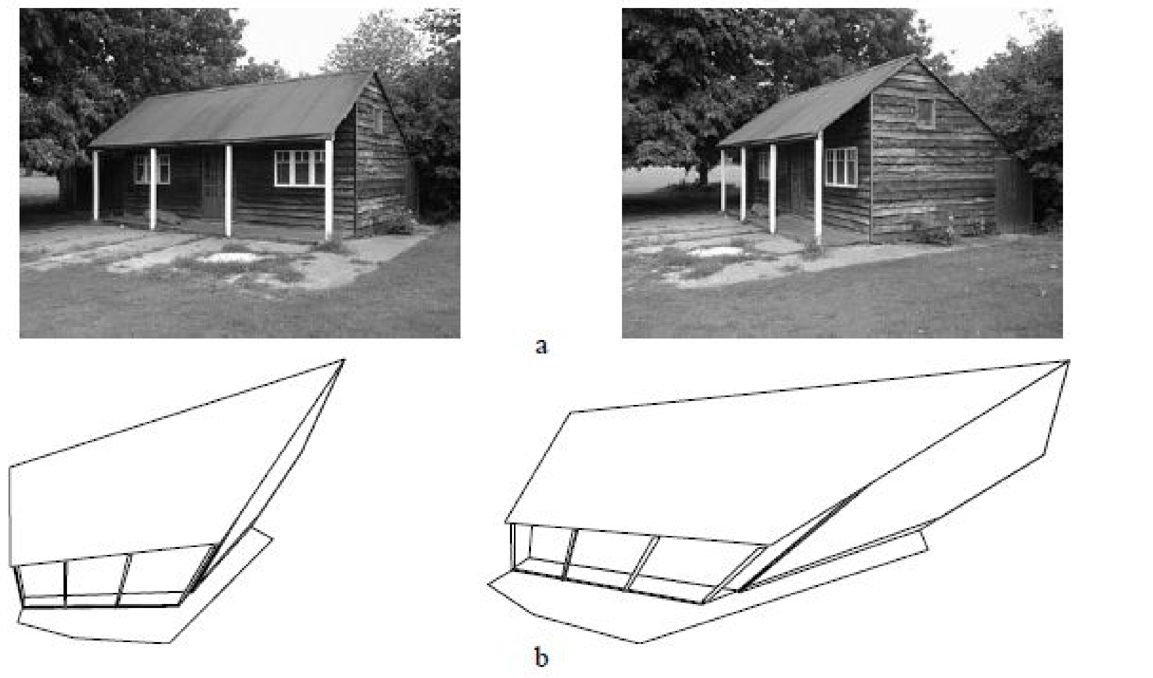


Fig. 10.3. **Projective reconstruction.** *(a) Original image pair. (b) 2 views of a 3D projective reconstruction of the scene. The reconstruction requires no information about the camera matrices, or information about the scene geometry. The fundamental matrix F is computed from point correspondences between the images, camera matrices are retrieved from F, and then 3D points are computed by triangulation from the correspondences. The lines of the wireframe link the computed 3D points*

If the reconstruction is derived from real images,there is a true reconstruction that can produce the actual points Xi of the scene.
->Video2

Calibration Techniques:

Intrinsic:

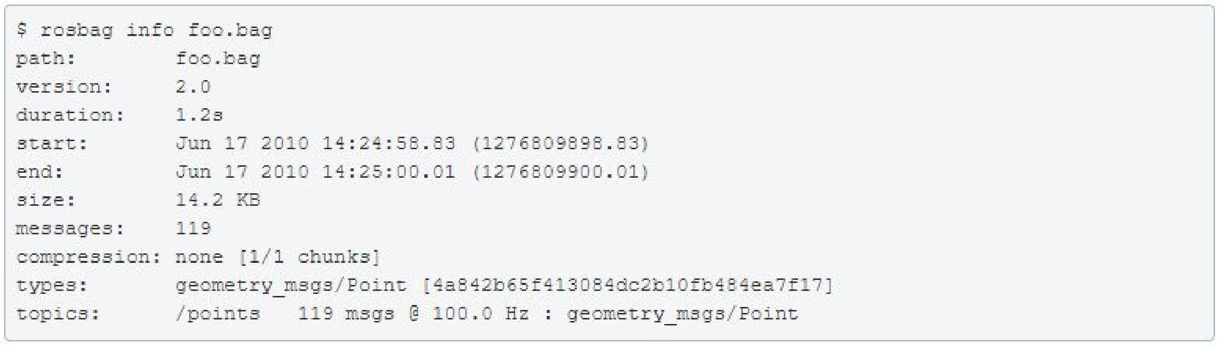
BAG FILES

A bag is a file format in ROS for storing ROS message data. Bags -- so named because of their .bag extension -- have an important role in ROS, and a variety of tools have been written to allow you to store, process, analyze, and visualize them.
Bags are typically created by a tool like rosbag, which subscribe to one or more ROS topics, and store the serialized message data in a file as it is received. These bag files can also be played back in ROS to the same topics they were recorded from or even remapped to new topics.

We can use the following command to get the details of a bag file:

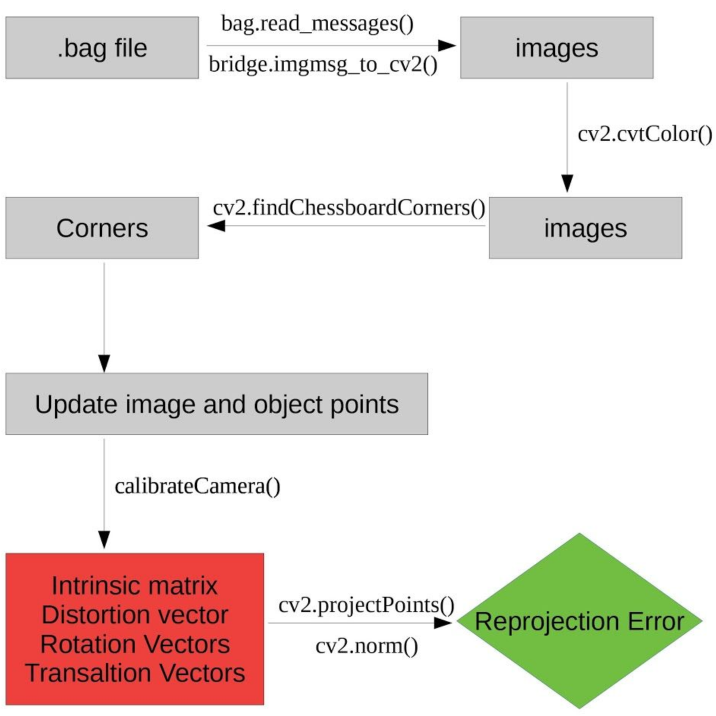
rosbag info foo.bag

The output might look something like this



```
$ rosbag info foo.bag
path:        foo.bag
version:     2.0
duration:    1.2s
start:       Jun 17 2010 14:24:58.83 (1276809898.83)
end:         Jun 17 2010 14:25:00.01 (1276809900.01)
size:        14.2 KB
messages:    119
compression: none [1/1 chunks]
types:       geometry_msgs/Point [4a842b65f413084dc2b10fb484ea7f17]
topics:      /points   119 msgs @ 100.0 Hz : geometry_msgs/Point
```

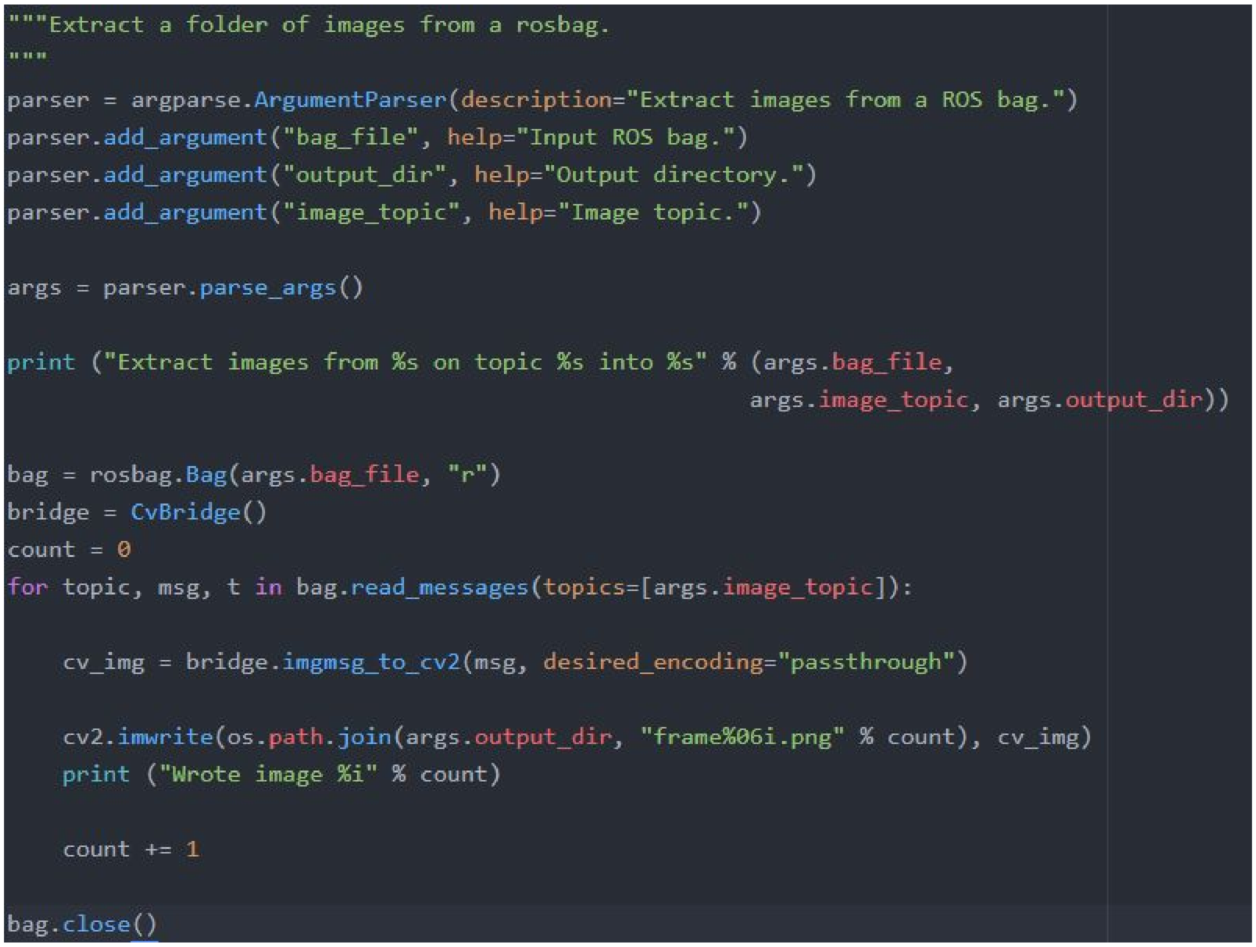
Outline:



Flowchart

Image Extraction:

The images contained in the bag file can be extracted using the below given code snippet:



```
"""Extract a folder of images from a rosbag.
"""
parser = argparse.ArgumentParser(description="Extract images from a ROS bag.")
parser.add_argument("bag_file", help="Input ROS bag.")
parser.add_argument("output_dir", help="Output directory.")
parser.add_argument("image_topic", help="Image topic.")

args = parser.parse_args()

print ("Extract images from %s on topic %s into %s" % (args.bag_file,
                                                      args.image_topic, args.output_dir))

bag = rosbag.Bag(args.bag_file, "r")
bridge = CvBridge()
count = 0
for topic, msg, t in bag.read_messages(topics=[args.image_topic]):

    cv_img = bridge.imgmsg_to_cv2(msg, desired_encoding="passthrough")

    cv2.imwrite(os.path.join(args.output_dir, "frame%06i.png" % count), cv_img)
    print ("Wrote image %i" % count)

    count += 1

bag.close()
```

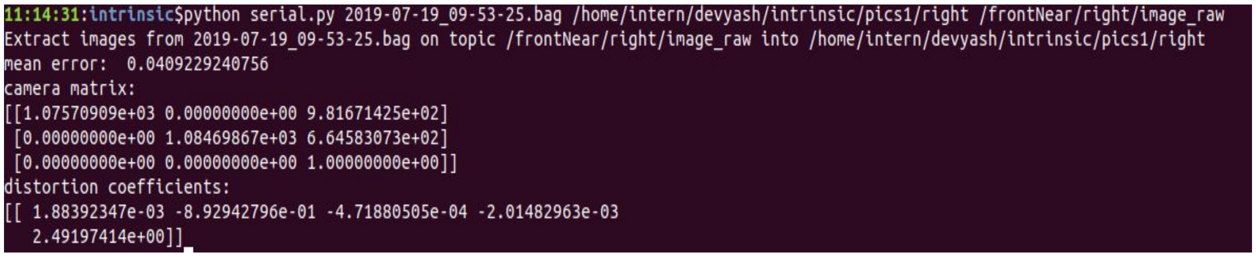
Code for image extraction

<u>RESULTS</u>

The algorithm takes 40 patterns from the given sample to calibrate the given camera. To cover all the possible orientations of the checkerboard, the selected sample images must be different in orientations. Three techniques have been used to select such samples-

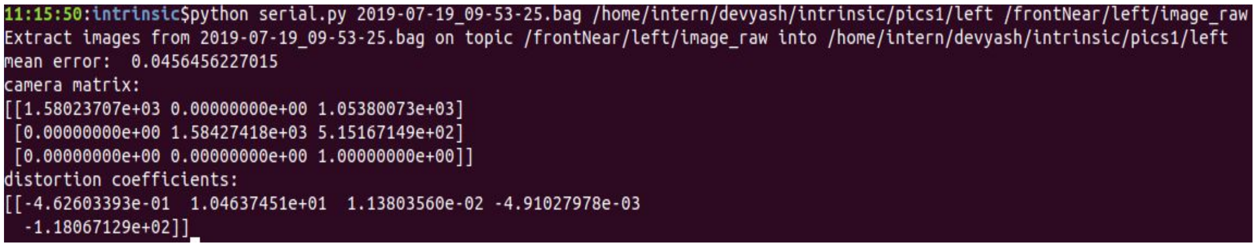
1. <u>Simply selecting first 40 images</u>: Very low reprojection error suggests a possibility of images with similar orientations of the board been selected for calibration. Therefore, a technique to select “good” images from the sample is required.

   For right images:



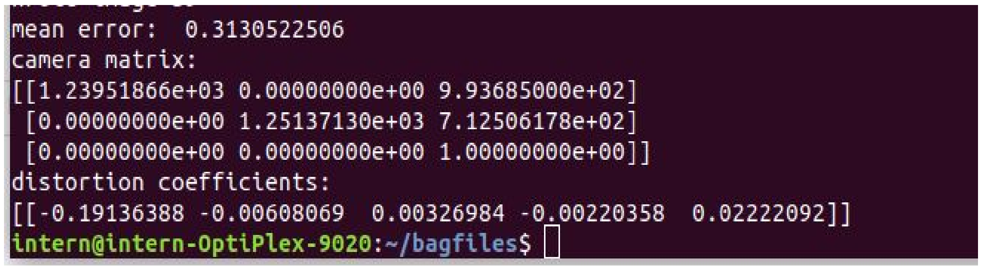
```
11:14:31:intrinsic$python serial.py 2019-07-19_09-53-25.bag /home/intern/devyash/intrinsic/pics1/right /frontNear/right/image_raw
Extract images from 2019-07-19_09-53-25.bag on topic /frontNear/right/image_raw into /home/intern/devyash/intrinsic/pics1/right
mean error:  0.0409229240756
camera matrix:
[[1.07570909e+03 0.00000000e+00 9.81671425e+02]
 [0.00000000e+00 1.08469867e+03 6.64583073e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
distortion coefficients:
[[ 1.88392347e-03 -8.92942796e-01 -4.71880505e-04 -2.01482963e-03
   2.49197414e+00]]
```

For left images:



```
11:15:50:intrinsic$python serial.py 2019-07-19_09-53-25.bag /home/intern/devyash/intrinsic/pics1/left /frontNear/left/image_raw
Extract images from 2019-07-19_09-53-25.bag on topic /frontNear/left/image_raw into /home/intern/devyash/intrinsic/pics1/left
mean error:  0.0456456227015
camera matrix:
[[1.58023707e+03 0.00000000e+00 1.05380073e+03]
 [0.00000000e+00 1.58427418e+03 5.15167149e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
distortion coefficients:
[[-4.62603393e-01  1.04637451e+01  1.13803560e-02 -4.91027978e-03
  -1.18067129e+02]]
```

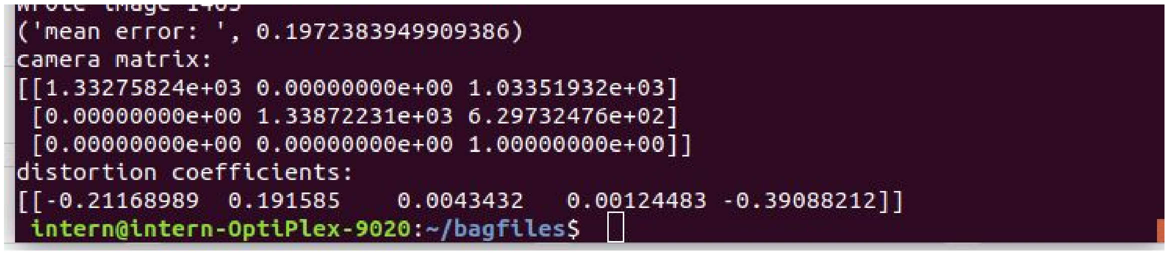
2). Skipping a few images after each image: Another approach we came up with is to skip about 20 to 25 images after selecting one so as to keep the variation amongst the selected images. The results obtained on including every 30th image are:



```
mean error:  0.3130522506
camera matrix:
[[1.23951866e+03 0.00000000e+00 9.93685000e+02]
 [0.00000000e+00 1.25137130e+03 7.12506178e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
distortion coefficients:
[[-0.19136388 -0.00608069  0.00326984 -0.00220358  0.02222092]]
intern@intern-OptiPlex-9020:~/bagfiles$
```

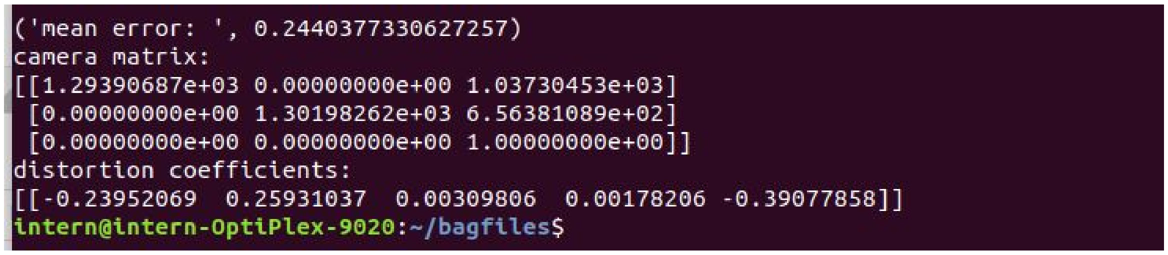
3). Calculating Structural Similarity index and filtering: Involves calculating the structural similarity index to compare every image with the previous image selected for calibration, based on the assumption that each image is more likely to be similar to the image previous to it as compared to others. A threshold value of SSIM is selected arbitrarily and the images with their value higher than the threshold are rejected. The results, obviously, are highly sensitive to the threshold value.

a. Results with threshold SSIM = 0.65



```
('mean error: ', 0.1972383949909386)
camera matrix:
[[1.33275824e+03 0.00000000e+00 1.03351932e+03]
 [0.00000000e+00 1.33872231e+03 6.29732476e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
distortion coefficients:
[[-0.21168989  0.191585    0.0043432   0.00124483 -0.39088212]]
 intern@intern-OptiPlex-9020:~/bagfiles$
```

b. Results with threshold SSIM = 0.62



```
('mean error: ', 0.2440377330627257)
camera matrix:
[[1.29390687e+03 0.00000000e+00 1.03730453e+03]
 [0.00000000e+00 1.30198262e+03 6.56381089e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
distortion coefficients:
[[-0.23952069  0.25931037  0.00309806  0.00178206 -0.39077858]]
intern@intern-OptiPlex-9020:~/bagfiles$
```

References

1. **Multiple View Geometry** in Computer Vision. Second Edition. Richard Hartley, Andrew Zisserman.
2. OpenCV Python Tutorials - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html

: