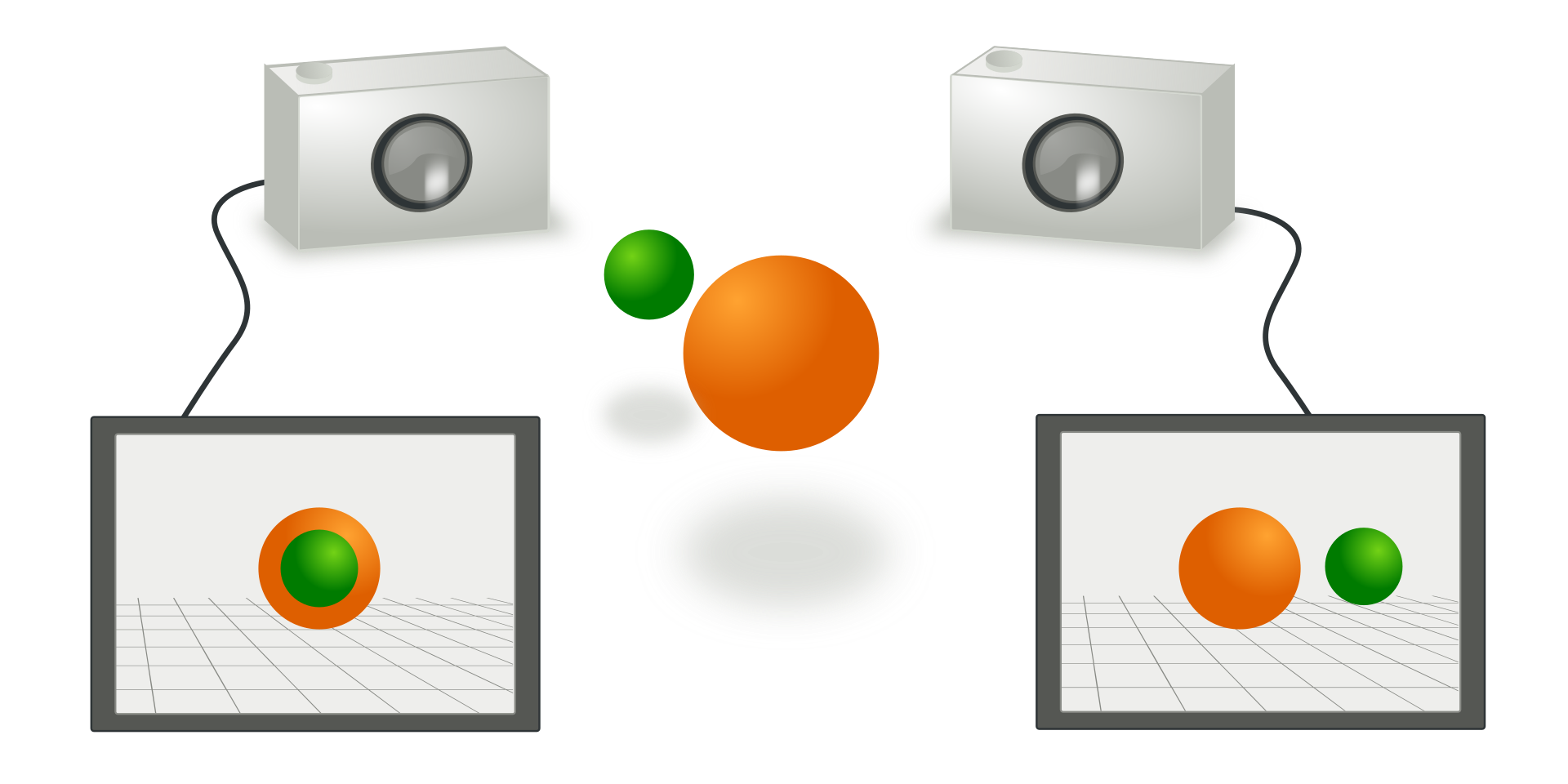


# 2. Mathematical background

## 1. Epipolar Geometry

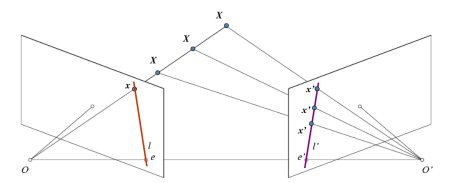
Epipolar geometry is the geometry of stereo vision. When two cameras view a 3D scene from two distinct positions, there are a number of geometric relations between the 3D points and their projections onto the 2D images that lead to constraints between the image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model.



### 1.1. Definition

The figure below depicts two pinhole cameras looking at point X. In real cameras, the image plane is actually behind the focal center. However, we simplified the problem by placing a *virtual* image plane in front of the camera, since real plane is symmetric about the focal center of the lens.

Each camera captures a 2D image of the 3D world. This conversion from 3D to 2D is referred to as a perspective projection and is described by the pinhole camera model.



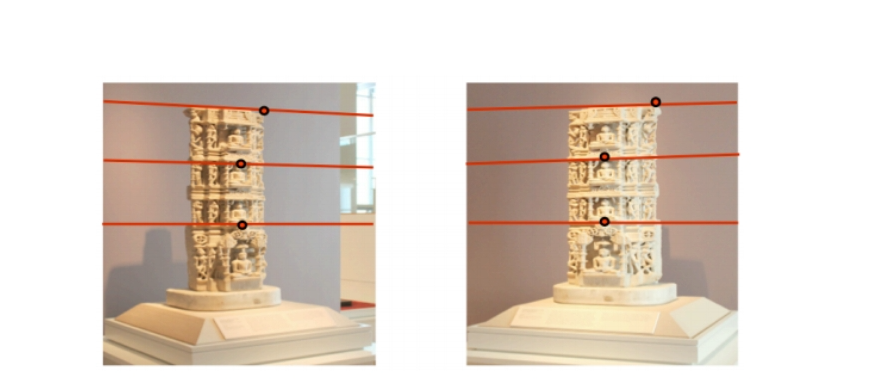
### 1.2. Epipole or epipolar point

Each center projects onto a distinct point into the other camera’s image plane. These two image points, denoted by and , are called epipoles or epipolar points.

Epipolar point is the intersection of line connecting two camera’s center and with their image plane.

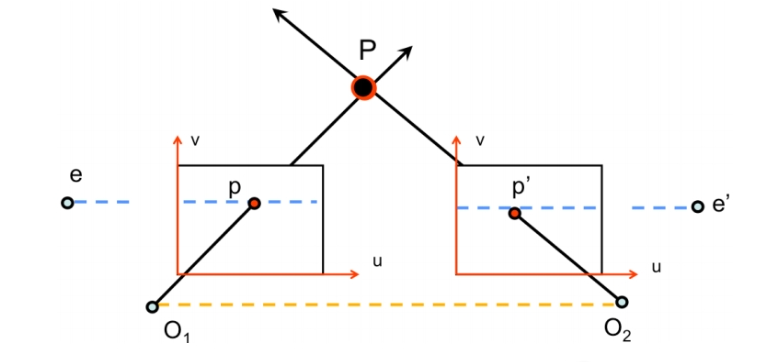
### 1.3. Epipolar line

Line projected on right camera creating a line called the epipolar line. Symmetrically, the line is seen by the right camera as a point and is seen as epipolar line by the left camera.



The red lines are epipolar lines

When two image planes are parallel then the epipoles and are located at infinity. Then the epipolar lines are parallel to axis of image.



### 1.4. Epipolar plane

form a plane, called epipolar plane. The epipolar plane and all epipolar lines intersect the epipoles regardless of where is located.

### 1.5. Epipolar constraint

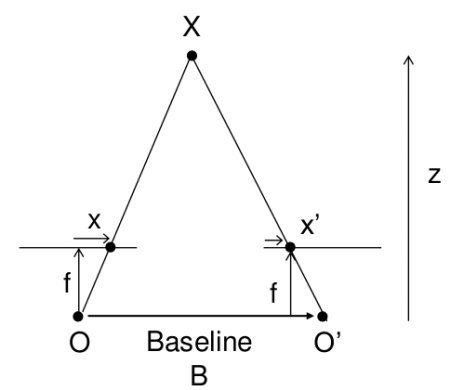
If the relative position of the two cameras is known, this leads to two important observations:

Assume the projection point , the epipolar line and the point projects into the right image is known. A point which must lie on this particular epipolar line.

This provides an epipolar constraint: the projection of on the right camera plane xR must be contained in the epipolar line. All points e.g.  on the line will verify that constraint.

Epipolar constraints can also be described by the essential matrix or the fundamental matrix between the two cameras.

### 1.6. Disparity and Depth map



The above diagram contains equivalent triangles. Writing their equivalent equations will yield us following result:

So the depth would be:

Where:

* and are the distance between image points and their corresponding camera center.
* is the baseline, distance between two camera center.
* is the focal of both camera (they should have the same).

So in short, the above equation says that the depth of a point in a scene is inversely proportional to the difference in distance of corresponding image points and their camera centers.

## 2. Essential matrix

### 2.1. Coordinate representation

This derivation follows the paper by Longuet-Higgins.

For simplicity, we assume all the cameras are **normalized** and project the 3D world onto their respective image planes. i.e .

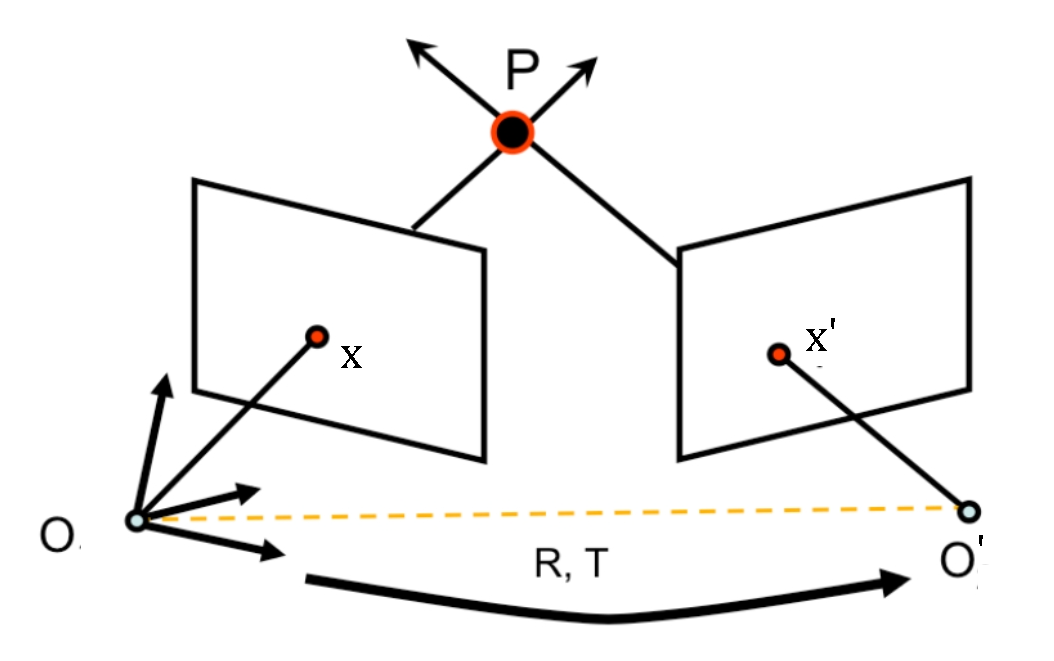
Let the 3D coordinates of a point **P** be and relative to each cmera’s coordinate system.

The mapping from the coordinates of a 3D point P to the 2D image coordinates of the point’s projection onto the image plane, according to the pinhole camera model, is given by:

More compactly as:

Where are the image 2D coordinate, are real life 3D cooordinates.

### 2.2. Set up camera framework



Let further assume world reference coordinate is associated with the first camera with the second camera offset by a rotaion matrix and 3 dimensional translation matrix . This implies:

So the camera matrix will be:

Or (because of cameras are normalized):

### 2.3. Essential matrix derivation

Since the vector and lie in the same epipolar line, then their cross product would produce a vector normal to the epipolar plane:

**Reminder from linear algebra:** The cross product between any two vectors a and b as a matrix-vector multiplication:

Combining this expression with Equation 6, we can convert the cross product term into matrix multiplication, giving:

Then, the **Essential Matrix** is creating a com- pact expression for the epipolar constraint:

The Essential matrix is a 3 × 3 matrix that contains 5 degrees of freedom. It has rank 2 and is singular.

### 2.4. Essential matrix mapping

Different to a homography which maps a point to a point, an essential matrix maps a **point** to a **line**. Furthermore, let’s consider an epipolar line , with the form of , or in vector form:

Then it is easy to see that, from Equation 8:

### 2.5. Essential matrix kernel

Since every lines on image plane pass epipolar. So , combine with Equation 9, we have . Furthermore, thus is normal with .

In short, essential kernel defines the epipole:

(points in normalized camera coordinates)

## 3. Fundamental matrix

### 3.1. Camera matrix

How do you generalize to uncalibrated cameras? Recall the Equation 3:

First, we must definde is a point from 3d world. We get two projected point on cameras:

Say we have canonical cameras transform space by a general homography matrix , then we have projections of to the corresponding camera images.

### 3.2. Fundamental matrix derivation

Recall that in the canonical case from Equation 7:

By substituting in the values of and , we get:

Let the matrix as the **Fundamental Matrix** which acts the same to the Essential matrix from previous but also encondes information about the camera matrices and and the relative translation T and rotation R between the cameras.

Therefore, it is also useful in computing the epipolar lines associated with p and p′, even when the camera matrices K, K′ and the transformation R, T are unknown.

### 3.3. Propertises of fundamental matrix

Similar to the Essential matrix, we can compute the epipolar lines and from just the fundamental matrix and the corresponding points.

Fundamental matrix contains 7 degrees of freedom, while Essential matrix’s 5 degrees of freedom.

If we know the fundamental matrix, then simply knowing a point in an image gives us an easy constraint (the epipolar line) of the corresponding point in the other image. Therefore, without knowing the actual position of in 3D space, or any of the extrinsic or intrinsic characteristics of the cameras, we can establish a relationship between any and .

## 4. The Eight-Point algorithm

### 4.1. Formulating a homogeneous linear equation

With each correspondent and

The constraint can be rewritten as:

That is represents the flatten **Fundamental matrix** vector and this vector must be othorgonal to vector .

Each pair of corresponding image points produces a vector . Given a set of 3D points corresponding to a set of vector and all of them must satisfy:

Collect vector as the row of matrix and:

Where is a matrix with .

### 4.2. Solving the equation

In pracitce, there are noise so solution vector f is defined only up to an unknown scale. So it is better to use more than eight correspondences and create a larger . Furthermore, is often rank-deficient, so we approximate by **Linear least squares**:

The subject is to avoid the trivial solution f.

The solution to this optimize problem can be found by Singular Value Decomposition (SVD). is the right singular vector corresponding to the smallest singular value of . A reshape of this into matrix give result called as .

### 4.3. Enforcing the internal constraint

An important property of the fundamental matrix is that it is singular, in fact of rank 2. Furthermore, the left and right null spaces ( and ) of are generated by the vectors representing the two epipoles in the images i.e . However, often, dealing with noisy image gives the result from Equation 14 usually does not have rank 2.

We find a best rank-2 matrix approximation of F by the mean of:

The constrain is to make is singular.

This problem is solved again by SVD, where $F = UV^T $ then the best rank-2 approximation is found by:

### 4.4. Normalized algorithm

#### 4.4.1. Problems

The problem of the standard algorithm is that is often ill-conditioned for SVD. For SVD to work properly, shuld have one singular value equal or near to zero, with the rest are non zero.

However, correspondences coordinate will often have extremely large values in the first and second compared to the third due to large pixel range of mordern digital camera.

Furthermore, if the image points used to construct lie in a relatively small region of the image , then and are relatively similar, resulting in has one ery large singular value, with the rest relatively small.

#### 4.4.2. Solution

To solve this, map each coordinate system of two images independently into a new system satisfying two conditions:

* The origin of the new system should be at the centroid (center of gravity) of the image points. This is accomplished by translating original origin to new one.
* After the translation, the coordinates have to be uniformed so that the mean of distance from each points to the origin equals . This can e done by the scaling factor for each respective image

Afterwards, a distinct coordinate transformation for each of the two images. We obtain a new homogeneous image:

This normalization is only dependent on the image points which are used in a single image and is, in general, distinct from normalized image coordinates produced by a normalized camera.

Note that we overload the notations because of the obivious relations.

The epipolar constraint based on the fundamental matrix can now be rewritten as:

Where .

This means that it is possible to use the normalized homogeneous image coordinates, and , to estimate the transformed fundamental matrix using the basic eight-point algorithm described above.

The solution is now more well-defined from the homogeneous equation than is relative to . Once has been determined, we can de-normalized to give by:

## 5. Image Rectification

### 5.1. Parallel image planes

Recall that when two image planes are parallel then the epipoles and are located at infinity and the epipolar lines are parallel to the axis of image. We can assume that the two cameras has the same intrinsic and there is no rotation between them . Furthermore, we assume there is only a translation along the axis . Then the essential matrix would be:

Once is known, we can find the directions of the epipolar line associated with point in the second image plane:

We can see that epipolar line horizontal, parallel to the axis. As is the direction of , which is computed in the same manner.



Furthermore, if we use the epipolar constraint , then we arrive the fact that , which is very handy in indentifying correspondences. Therefore **rectification** makes any two given images become parallel, becomes useful then discerning the relationship between corresponding points in images.

### 5.2. Rectification Setup:

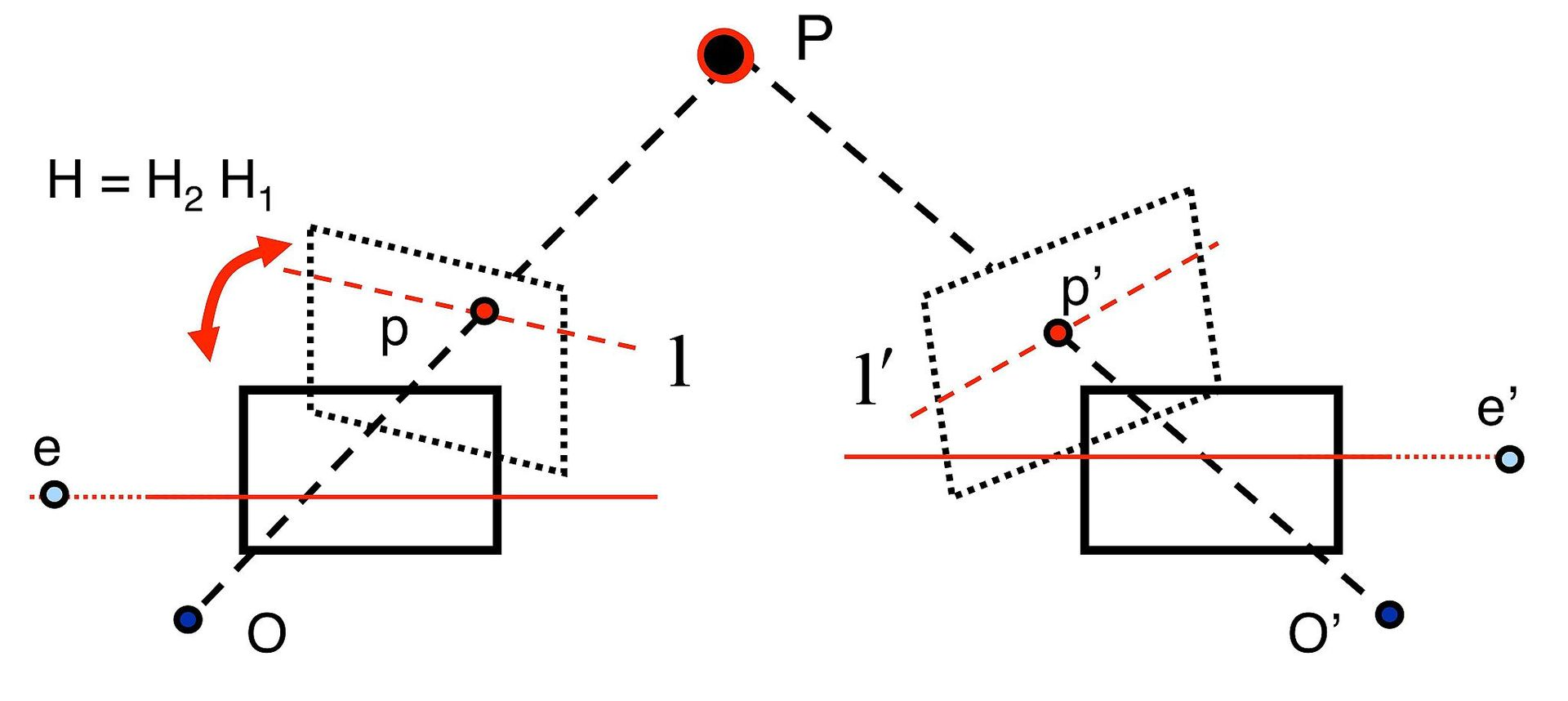
We compute two homographies and that apply to the image planes to make the resulting planes parallel.

Rectifying a pair of images does not require knowledge of camera intrinsic and , or the transformation matrix . Instead, we can use the Fundamental matrix estimated by the Normalized Eight Point algorithm. Upon getting the Fundamental matrix, we can compute the epipolar lines and for each correspondence and and .

From the set of epipolar lines, we can then estimate the epipoles and of each image. This is because epipole lies in the intersection of all the epipolar lines. In pratice, due to noises, all the epipolar lines will not intersect in a single point. Therefore, epipole esstimated by minimizing the least squared error of fitting a point to all the epipolar lines. Recall that a line is defined by :

We can use SVD to find . is the smallest eigenvector of .

Generally, the solution epipoles and are not at infinity along the horizontal axis. If they were, by definition, the images have already been parallel. Thus we gain some insight to make the images parallel: find a homography to map epipole to infinity along the axis.



Specifically, this means that we want to find a pair of homographies H1, H2 that we can apply to the images to map the epipoles to infinity. We map the second epipole to a point at infinity on the horizontal axis. One condition that leads to good results in practice is to insist that the homography acts like a transformation that applies a translation and rotation on points near the center of the image.

### 5.3. Sending epipoles to infinity

First, we translate the second image so its center is at in homogeneous coordinates using the translation matrix :

After translation, we apply a rotation to place the epipole on the the horizontal axis at some point . If the translated epipole is located at homogeneous coordinates , then the rotation applied is:

Where if and otherwise.

After applying this rotation, bringing any point at to a point at infinity on the horizontal axis requires applying the transformation:

After applying this transformation, we finally have an epipole at infinity, so we can translate back to the regular image space. Thus, the homography that we apply on the second image to rectify it is:

Now that a valid $ H\_2 $ is found, we need to find a matching homography $ H\_1$ for the first image. We do so by finding a transformation that minimizes the sum of square distances between the corresponding points of the images:

## 6. Camera Pose

### 6.1. Estimate Camera Pose from Essential Matrix

The camera pose consists of 6 degrees-of-freedom (DOF): Rotation (Roll, Pitch, Yaw) and Translation of the camera with respect to the world. Since the E matrix is identified, the four camera pose configurations where is the camera center and is the rotation matrix, can be computed. Thus, the camera pose can be written as: where is the intrinsic parameter matrix.

Four pose configurations can be computed from the E matrix. Let and .

The four configurations are:

* and
* and
* and
* and

### 6.2. Check for Chirality Condition using Triangulation

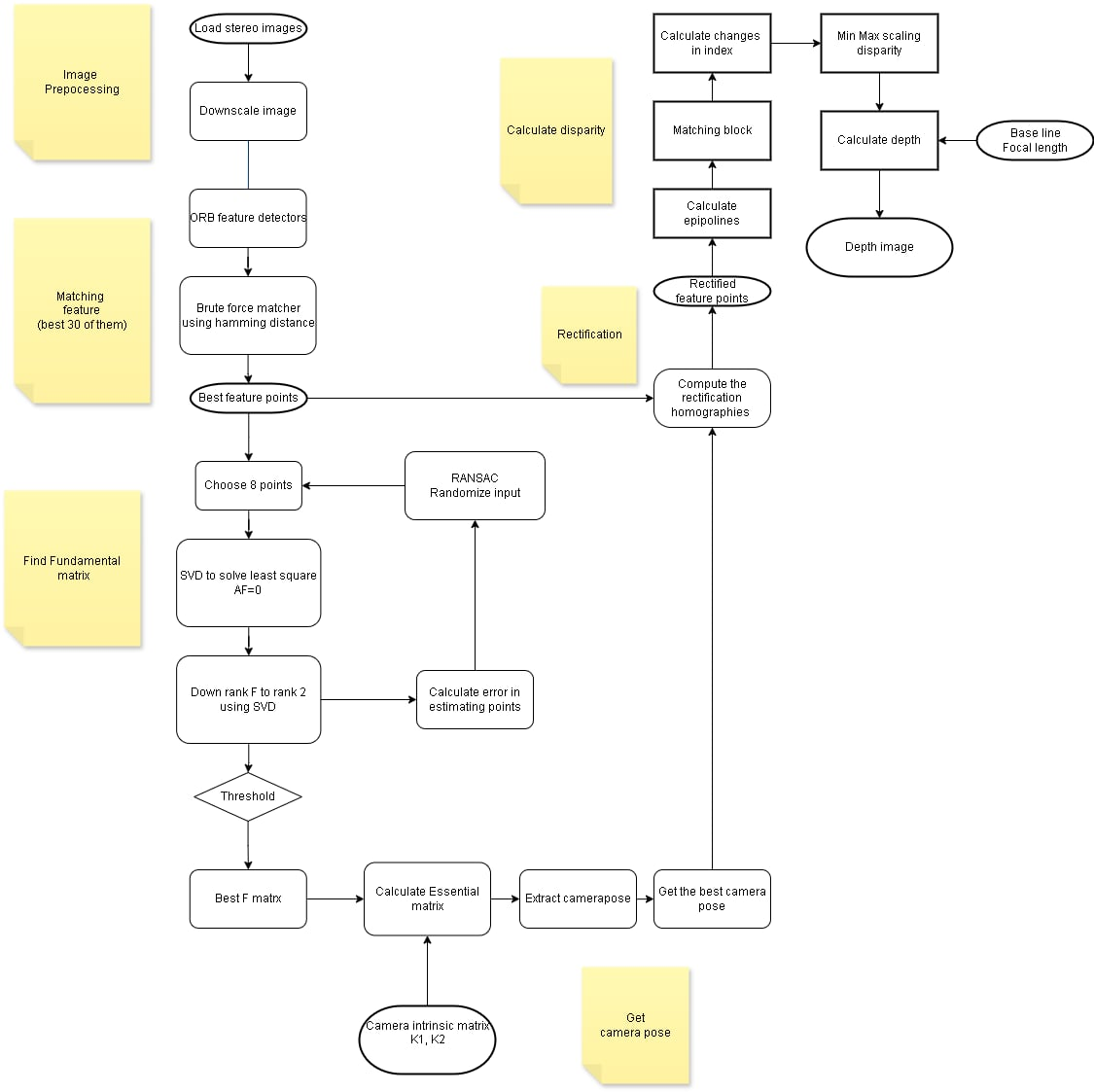
In the previous section, we computed four different possible camera poses for a pair of images using the essential matrix. To find the correct unique camera pose, we need to remove the ambiguity. This can be accomplished by checking **the chirality condition**, i.e., **the reconstructed points must be in front of the cameras**.

To check the chirality condition, triangulate the 3D points (given two camera poses) using linear least squares to check the sign of depth in the camera coordinate system with respect to the camera center.

A 3D point is in front of a camera if it satisfies , where is the third row of the rotation matrix (z-axis of the camera). Not all triangulated points satisfy this condition due to the presence of correspondence noise. The best camera configuration is the one that produces the maximum number of points satisfying the chirality condition.

# 3. Solution

Most of the algorithm used in our codes are described above.



## 3.1. calibration.py

In this file, we compute the feature points, then estimate the fundamental and essential matrix from them. Furthermore, we find the correct camera pose (Rotation and Translation) from the matrix.

### 3.1.1. def draw\_keypoints\_and\_match(img1: MatLike, img2: MatLike, nfeatures: int = 500) -> Tuple[NDArray, NDArray, MatLike]: …

Finding keypoints using ORB feature detection and descriptors in the image and find best matches using brute force based matcher.

### 3.1.2. def compute\_Fundamental\_matrix(kp1\_list: NDArray, kp2\_list: NDArray) -> NDArray: …

Calculate the matrix from a set of 8 points using SVD. Furthermore, the rank of matrix is reduced from 3 to 2 to make the epilines converge.

More details at section [8 points algorithm](#X93bd63d86f28363d0ee68b450dcdc27315aaa4e)

### 3.1.3. def RANSAC\_F\_mat(kp1\_list: NDArray, kp2\_list: NDArray, max\_inliers=20, threshold=0.05, max\_iter=1000) -> NDArray: …

Shortlist the best F matrix using RANSAC based on the number of inliers.

RANSAC randomize 8 feature points to compute and calculate errors in estimating the points. Then we classify inliers with predefined threshold. Best is one with the most inliers.

### 3.1.4. def compute\_Essential\_matrix(F\_mat: NDArray, K1: NDArray, K2: NDArray) -> NDArray: …

Calculation of the Essential matrix with , and camera intrinsic.

More details at section [Essential-matrix](#Xe3922fb50a7936d15750767c1efd2e6d9833240)

### 3.1.5. def get\_camerapose(E\_mat: NDArray) -> list[list[NDArray]]: …

Extracts all the camera pose solutions from the E matrix.

More details at section [Camera Pose](#X3ec3ed43ca4021f69d685d65d0eb8f12efc3f98)

### 3.1.6 def disambiguate\_camerapose(camera\_pose: list[list[NDArray]], list\_kp1: NDArray): …

Find the correct camera pose based on the chirelity condition from all 4 solutions.

More details at section [Camera Pose](#X3ec3ed43ca4021f69d685d65d0eb8f12efc3f98)

### 3.1.7. def drawlines(img1src: MatLike, img2src: MatLike, lines, pts1src, pts2src, random\_seed=0) -> tuple[MatLike, MatLike]: …

Visualize the epilines on both the images

## 3.2. correspondence.py

### 3.2.1. def sum\_of\_squared\_diff(pixel\_vals\_1, pixel\_vals\_2):

We use SSD formular:

### 3.2.2. def block\_comparison(y, x, block\_left, right\_array, block\_size, x\_search\_block\_size, y\_search\_block\_size):

Block comparison function to find minimum SSD match. Two image block are correspodence if their SSD is the minimum.

### 3.2.3. def ssd\_correspondence(img1, img2):

Correspondence applied on the whole image to compute the disparity map and finally disparity map is scaled by min max scaling.

## 3.3. depth.py

### 3.3.1. def disparity\_to\_depth(baseline, f, img):

More details at section [Depth Map](#Xad97cb4e443f84ef15e090872681d03b27578a8)

## 3.4. rectification.py

### 3.4.1. def rectification(img1, img2, pts1, pts2, F):

Rectify the images to make camera pose’s parallel and thus make epiplines as horizontal. Since camera distortion parameters are not given we will use cv2.stereoRectifyUncalibrated(), instead of stereoRectify().

# 4. Evaluation

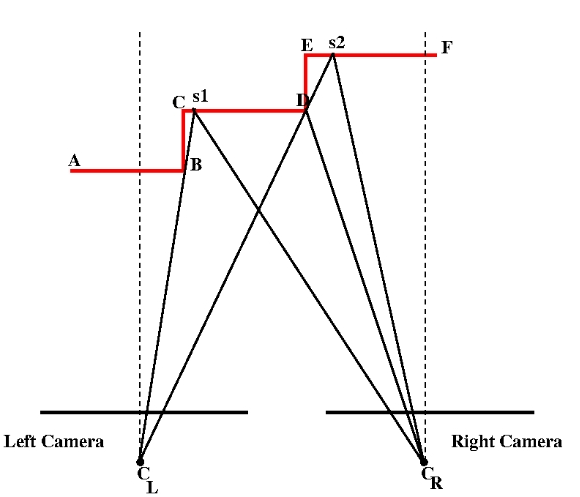
## 4.1. The results achieved

## 4.2. Compare with the results of the group on the same topic

## 4.3. Previous studies

### 4.3.1. Calculating correspondences (SGM)

We used Sum of Squared Differences (SSD) to calculate the coressponding block. Howvever, this function inherently assume there is no sudden depth variations and occlusions. Furthermore, this assume continuous or functionally variable disparities within the correlation window. Thus often produce very noisy results, jumping and gap pixels.



One portion of the scene may obscure other portions in either or both of the two cameras. Ex: .

The Semi-Global Matching (SGM) implements Mutual Information cost function. The pixelwise cost and the smoothness constraints are expressed by defining the energy E(D) that depends on the disparity image D:

Where:

* represents the disparity map.
* is the data term that measures the fidelity of the disparity map at pixel .
* denotes the neighborhood of pixel .
* is an indicator function.
* First term is the similarity cost, the sum of all pixel matching costs for the disparities of .
* penalty for small disparity changes (1 pixel).
* penalty is constant and larger for all larger disparity changes.

Using a lower penalty for small changes permits an adaptation to slanted or curved surfaces. The constant penalty for all larger changes (i.e. independent of their size) preserves discontinuities. While SSD is too simple to capture these characters.

Its module is available at *cv2.StereoBM\_create*

### 4.3.2. Disaprity post Filtering (WSL)

Since Weighted Least Squares (WSL) - based Fast Global Image smoothing mathemetical background as well as its algorithm is outside of our scope. We will not include it here.

Its objective is to enhance the quality of the input low-resolution depth map by increasing its spatial resolution. Low-resolution depth map and its associated high-resolution color image are used as inputs.

Its module is available at *cv2.ximgproc.createDisparityWLSFilter*

### 4.3.3. Deep Learning Based Approach

Deep learning

# References

https://web.stanford.edu/class/cs231a/course\_notes/03-epipolar-geometry.pdf https://cmsc426.github.io/sfm/#essential https://en.wikipedia.org/wiki/Epipolar\_geometry https://en.wikipedia.org/wiki/Essential\_matrix https://en.wikipedia.org/wiki/Fundamental\_matrix\_(computer\_vision)

For 4. Evaluation

https://core.ac.uk/download/pdf/11134866.pdf https://engineering.purdue.edu/kak/Tutorials/SemiGlobalMatching.pdf

WSL Filter: https://publish.illinois.edu/visual-modeling-and-analytics/files/2014/10/FGS-TIP.pdf