AirLab Summer School Series:

Visual Odometry Tutorial

Yafei Hu

July 07, 2020

THE ROBOTICS INSTITUTE

air lab

Part 1

Fundamentals of Computer Vision

1. Pinhole camera projection model

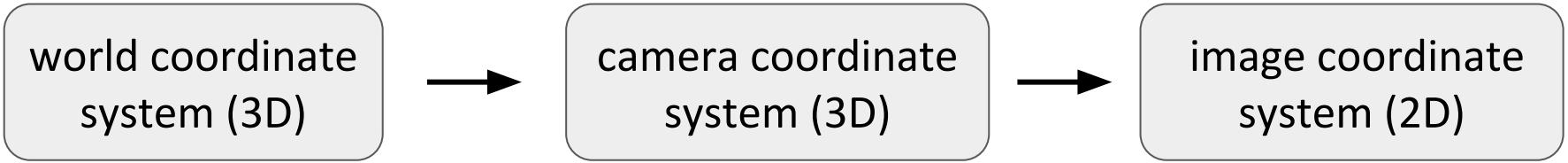
Camera projection is a transformation (mapping) between 3D world and 2D image

This mapping is described as:

$$x = PX$$

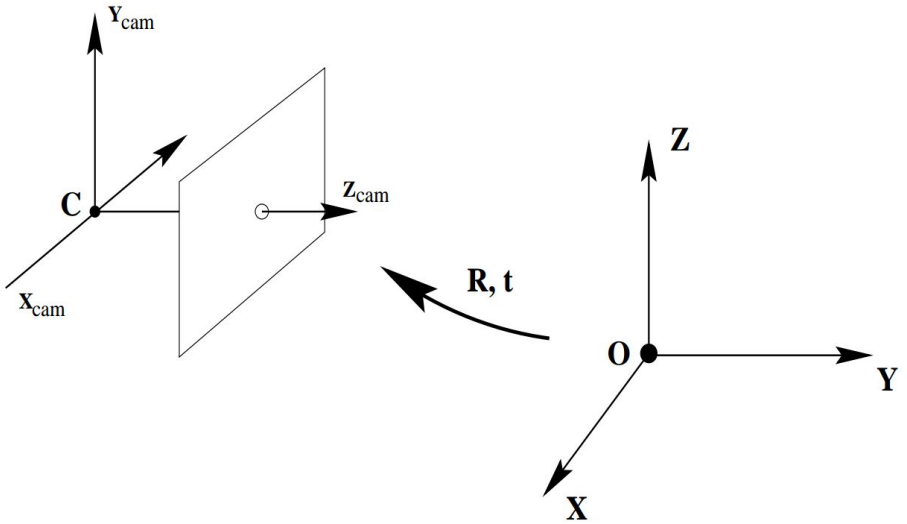
x: 2D Image point, P: Projection matrix, X: 3D world point

The projection consists of two parts:



1. Pinhole camera projection model

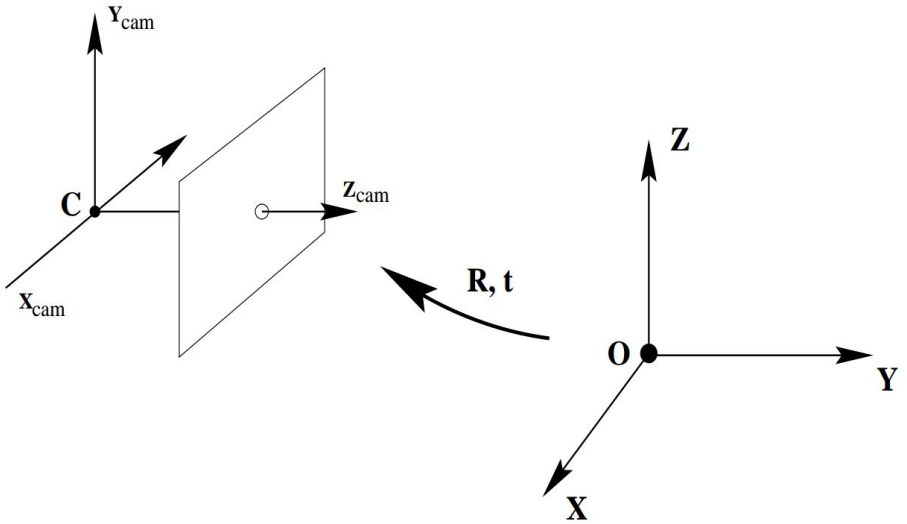
$[R,\ t]$ transform a 3D point in world coordinate system, denoted as X_W to camera coordinate system, denoted as X_C



in homogeneous coordinates: $$\tilde{X_C} = T\tilde{X_W} = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{X_W}$$

1. Pinhole camera projection model

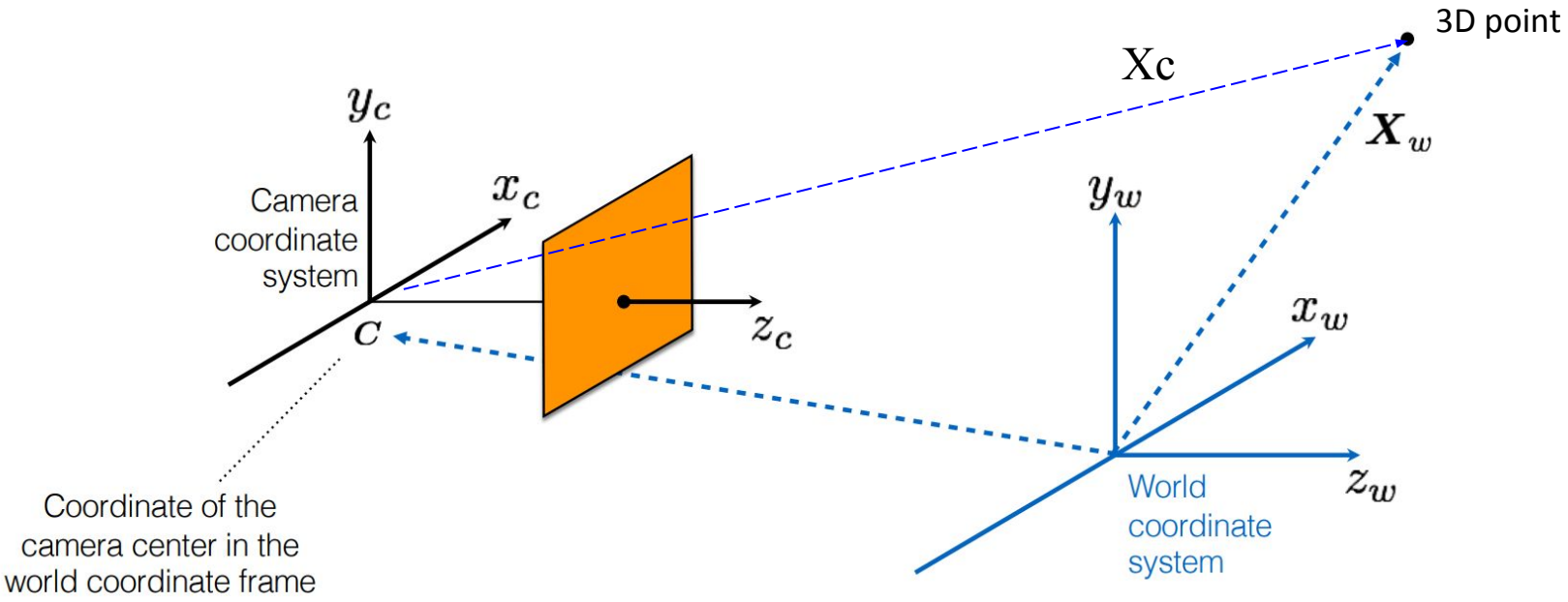
Does this $[R,t]$ means the rotation and translation of the camera?



in homogeneous coordinates: $\tilde{X_C} = T\tilde{X_W} = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{X_W}$

1. Pinhole camera projection model

e.g. No rotation, just pure translation:



$$X_C = X_W - C$$

C is also the camera translation in relative to the world coordinate system

1. Pinhole camera projection model

The real rotation and translation of camera is represented by the inverse transformation matrix of T

$$T^{-1} = \begin{bmatrix} R^T & -R^T t \\ \mathbf{0}^T & 1 \end{bmatrix}$$

R^T is rotation, $-R^T t$ is the translation of the camera, relative to the world coordinate system

1. Pinhole camera projection model

An object is projected to the image plane through pinhole camera:

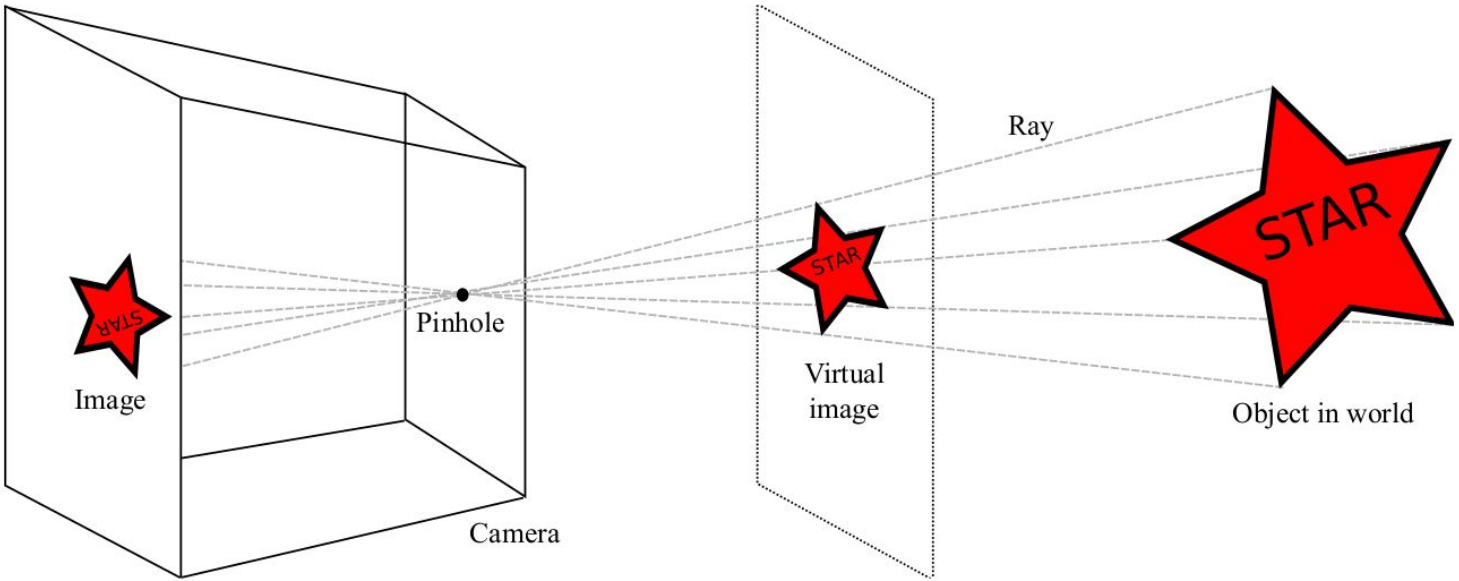
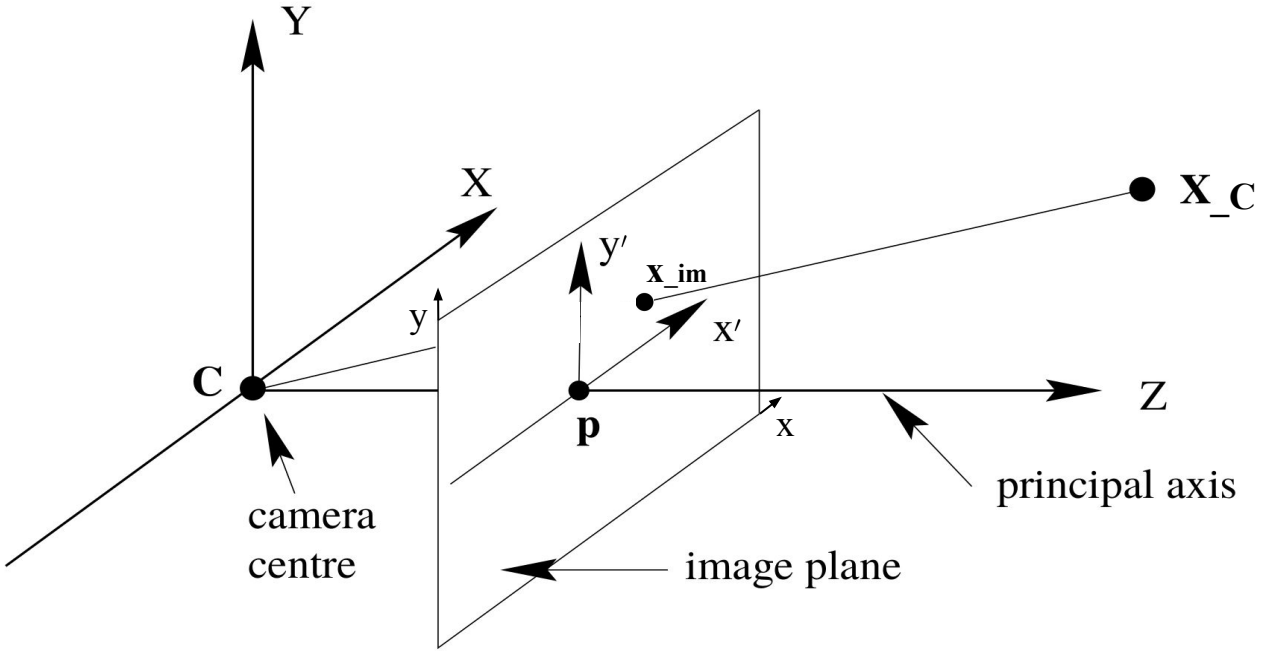


image plane

virtual image plane

1. Pinhole camera projection model

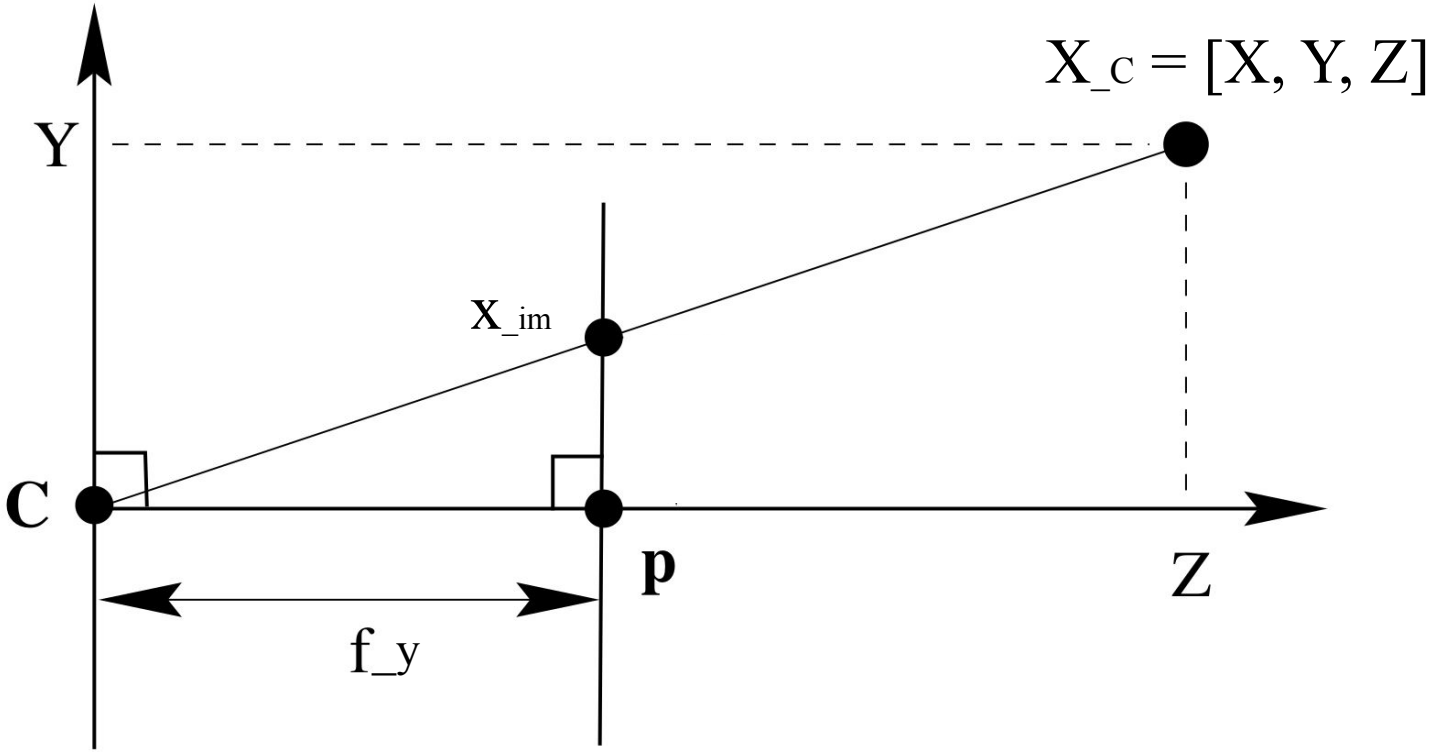
Then let's see how this 3D point in camera coordinate system, denoted as X_C, is projected to image coordinate system (**image plane**), denoted as x



C is the **camera center (or optical center)** and p(c_x, c_y) the **principal point**

1. Pinhole camera projection model

Take one plane of the camera coordinate system as example:



1. Pinhole camera projection model

What is the coordinate x, in image coordinate system?
(Hint: use properties of Similar Triangles)

$$\frac{Z}{f_y} = \frac{Y}{y} \qquad \frac{Z}{f_x} = \frac{X}{x}$$

$$y = \frac{f_y Y}{Z} \qquad x = \frac{f_x X}{Z}$$

Which is a mapping from 3D Euclidean space to 2D Euclidean space

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{f_x X}{Z} \\ \\ \frac{f_y Y}{Z} \end{bmatrix}$$

Is it linear?
What does this remind you of?

1. Pinhole camera projection model

What is the coordinate x, in image coordinate system?
(Hint: use properties of Similar Triangles)

$$\frac{Z}{f_y} = \frac{Y}{y} \qquad \frac{Z}{f_x} = \frac{X}{x}$$

$$y = \frac{f_y Y}{Z} \qquad x = \frac{f_x X}{Z}$$

Which is a mapping from 3D Euclidean space to 2D Euclidean space

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{f_x X}{Z} \\ \\ \frac{f_y Y}{Z} \end{bmatrix}$$

Is it linear? NO!

What does this remind you of? Homogeneous coordinate!

1. Pinhole camera projection model

Considering the principal point p,
whose coordinate in 2D image coordinate system is [c_x, c_y],

$$x_{im} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{f_x X}{Z} + c_x \\ \frac{f_y Y}{Z} + c_y \end{bmatrix}$$

$$\tilde{x_{im}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f_x X}{Z} + c_x \\ \frac{f_y Y}{Z} + c_y \\ 1 \end{bmatrix}$$

This gives us the coordinate of a projected 3D point, in 2D image coordinate system

How to write this mapping in matrix form?

1. Pinhole camera projection model

$$\tilde{x_{im}} = K X_C$$

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x X + c_x Z \\ f_y Y + c_y Z \\ Z \end{bmatrix} = Z \begin{bmatrix} f_x X / Z + c_x \\ f_y Y / Z + c_y \\ 1 \end{bmatrix}$$

K is called **camera intrinsic matrix**, **camera intrinsic parameters,** or **calibration matrix**

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

1. Pinhole camera projection model

Combine the 3D transformation together:

$$\tilde{X_C} = T\tilde{X_W} = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{X_W}$$

$$\tilde{x_{im}} = K\tilde{X_C} = K \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{X_W}$$

Sometimes we write in this form,

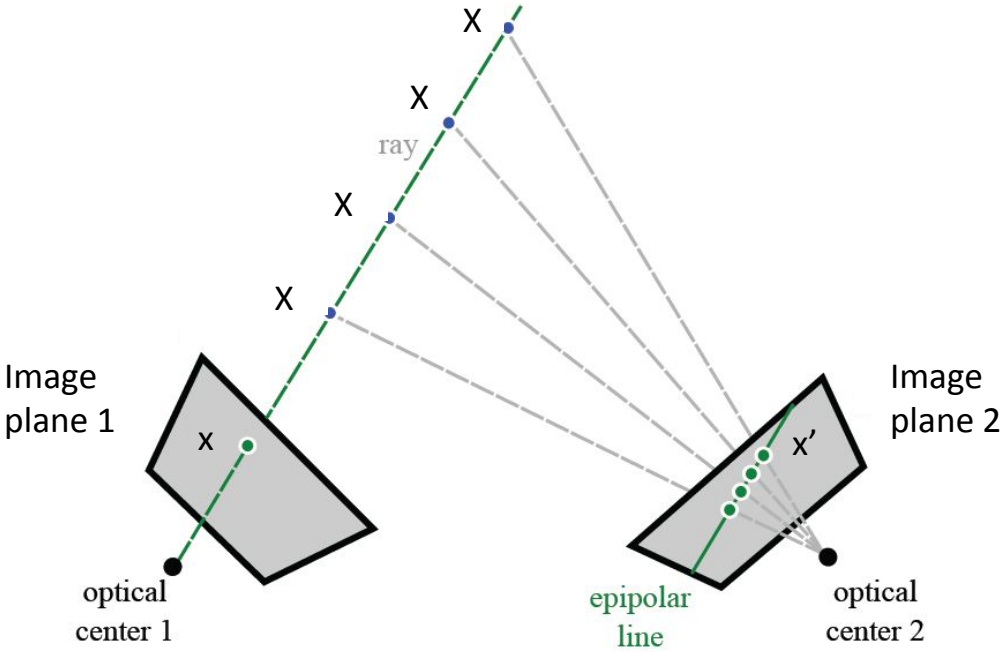
$$\tilde{x_{im}} = K\tilde{X_C} = K[R|t]\tilde{X_W}$$

[R|t] is called **camera extrinsic parameters**

2. Epipolar constraints

2.1 Epipolar line

- The 3D point X projected to x must lie in the ray that passes optical center 1 and x
- The projection of this ray on second camera forms a line, called **epipolar line**
- x and x’ forms **correspondence**

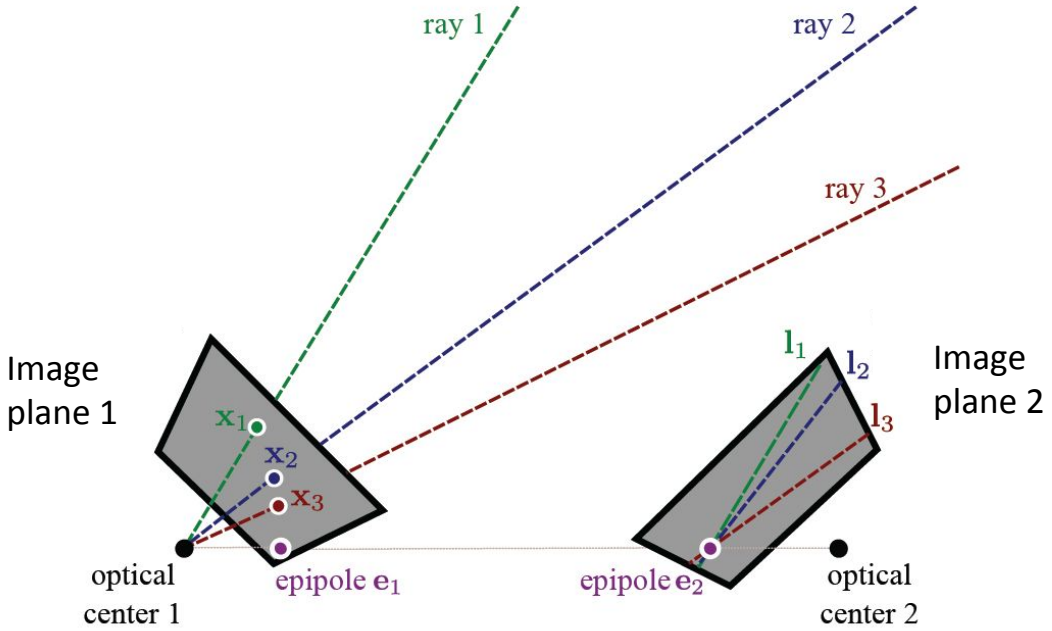


2. Epipolar constraints

2.2 Epipoles

Now consider a number of points in the first image:

Each point x_1, x_2 and x_3 is associated with a ray

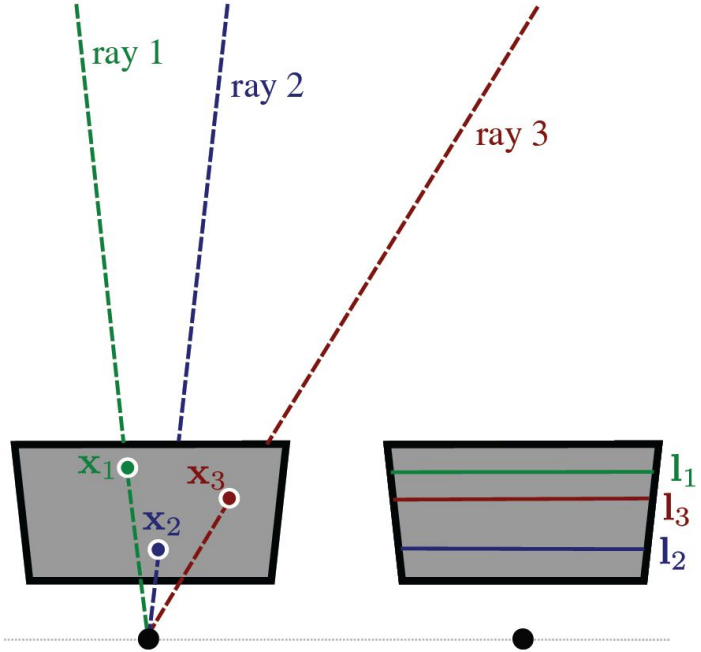


2. Epipolar constraints

Special cases of epipoles:

(1) Camera movement is a pure translation perpendicular to the optical axis (parallel to the image plane)

The epipolar lines are parallel and the epipole is at infinity

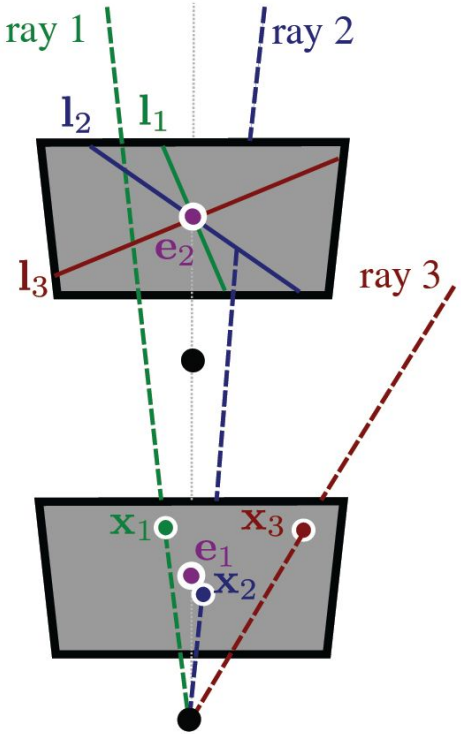


2. Epipolar constraints

Special cases of epipoles:

(2) Camera movement is a pure translation along the optical axis

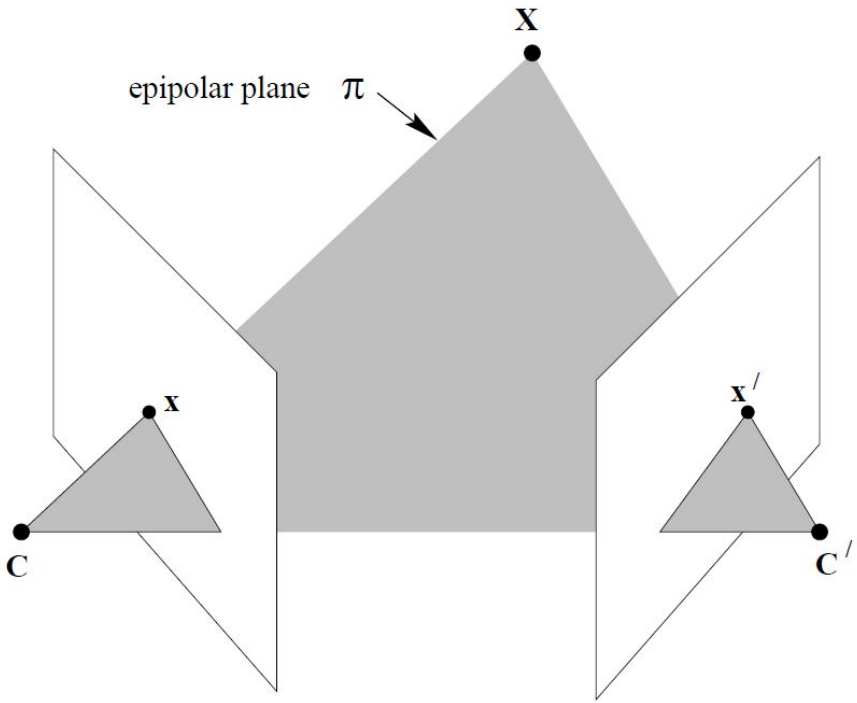
The epiholes have same coordinates in both images. Epipolar lines form a radial pattern.



2. Epipolar constraint

2.3 Epipolar plane

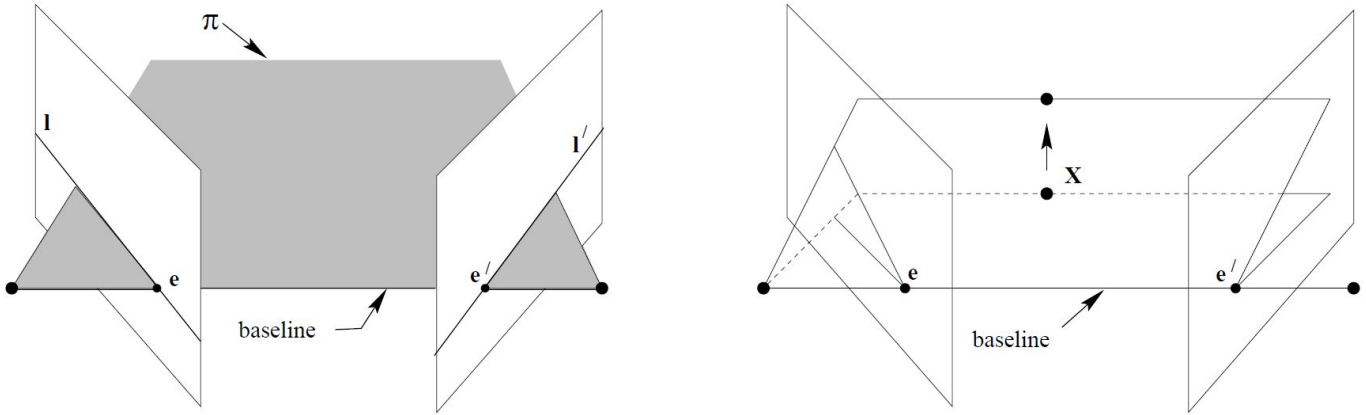
The optical centers C and C’, 3D point X, and its projection on images x and x’ lie in a common plane π, called **epipolar plane**.



2. Epipolar constraint

2.4 Baseline

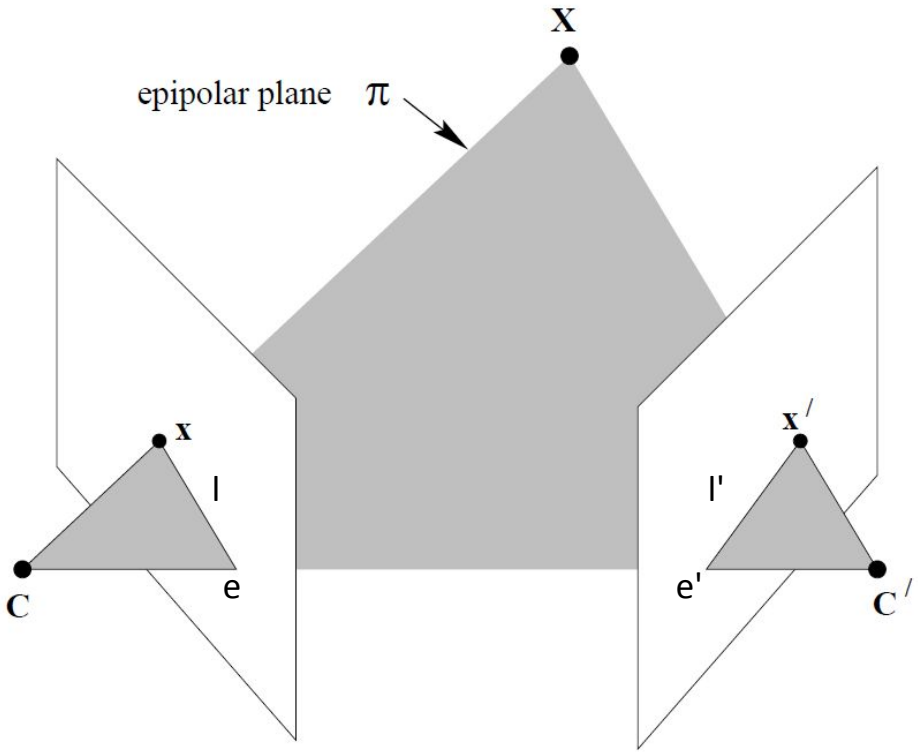
The camera **baseline** is the line formed by optical center C, C’ and epipoles e, e’



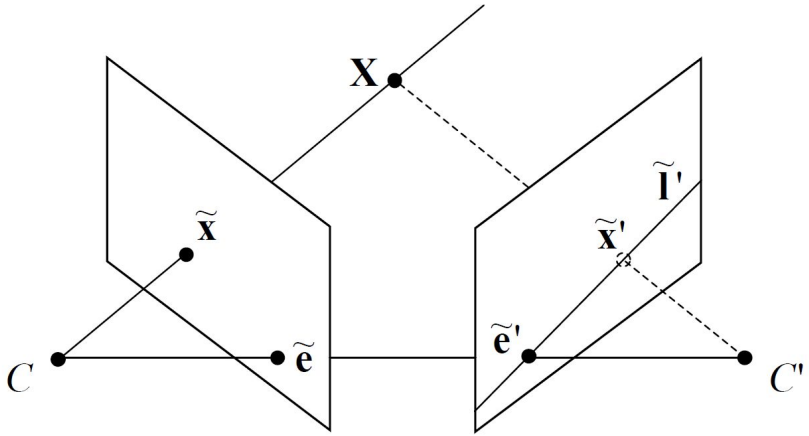
- Baseline intersects each image plane at the epipoles e and e’.
- Any plane π containing the baseline is an epipolar plane, and intersects the image planes in corresponding epipolar lines l and l’.
- As the position of the 3D point X varies, the epipolar planes “rotate” about the baseline. This family of planes is known as an epipolar pencil. All epipolar lines intersect at the epipole.

3. The Essential matrix

The **essential matrix**, denoted as E, is a **3 x 3** matrix that encodes epipolar constraint.



3. The Essential matrix



We define essential matrix E as

$$E = [t_\times]R = t \times R$$

Which also gives the epipolar constraint,

$$\tilde{x'}^T E \tilde{x} = 0$$

4. The Fundamental matrix

Then we give the definition of **fundamental matrix,** denoted as F

$$F = (K'^{-1})^T [t_\times] R K^{-1}$$

fundamental matrix also contains the epipolar constraints

$$\tilde{x'}^T F \tilde{x} = 0$$

Relationship between fundamental matrix F and essential matrix E

$$F = (K'^{-1})^T E K^{-1}$$

E* is a special case of *F

4. The Fundamental matrix

More about fundamental matrix

Epipolar line:

Given image point x in one view, we can get the epipolar line l' in another view,

$$l' = F\tilde{x}$$

According to epipolar constraint,

$$\tilde{x'}^T F\tilde{x} = 0$$

Substitute epipolar line we have,

$$\tilde{x'}^T l' = 0$$

What does it mean?

4. The Fundamental matrix

Function of a line in 3D space:

$$ax + by + c = 0$$

Sometimes represented as,

$$\boldsymbol{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

If a point x'= [x, y, 1] is on a line l', then we have,

$$\tilde{x'}^T l' = 0$$

This shows epipolar constraint in other perspective

4. The Fundamental matrix

What about epipole?
Epipole also lies in epipolar line,

$$e'^T l' = 0$$

With the definition of epipolar line,

$$e'^T F \tilde{x} = 0$$

So we have,

$$e'^T F = 0$$

Take the transpose of both side,

$$F^T e' = 0$$

4. The Fundamental matrix

Given a set of matched points,

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

the relationship between F and x:

$$\tilde{x'}^T F \tilde{x} = 0$$

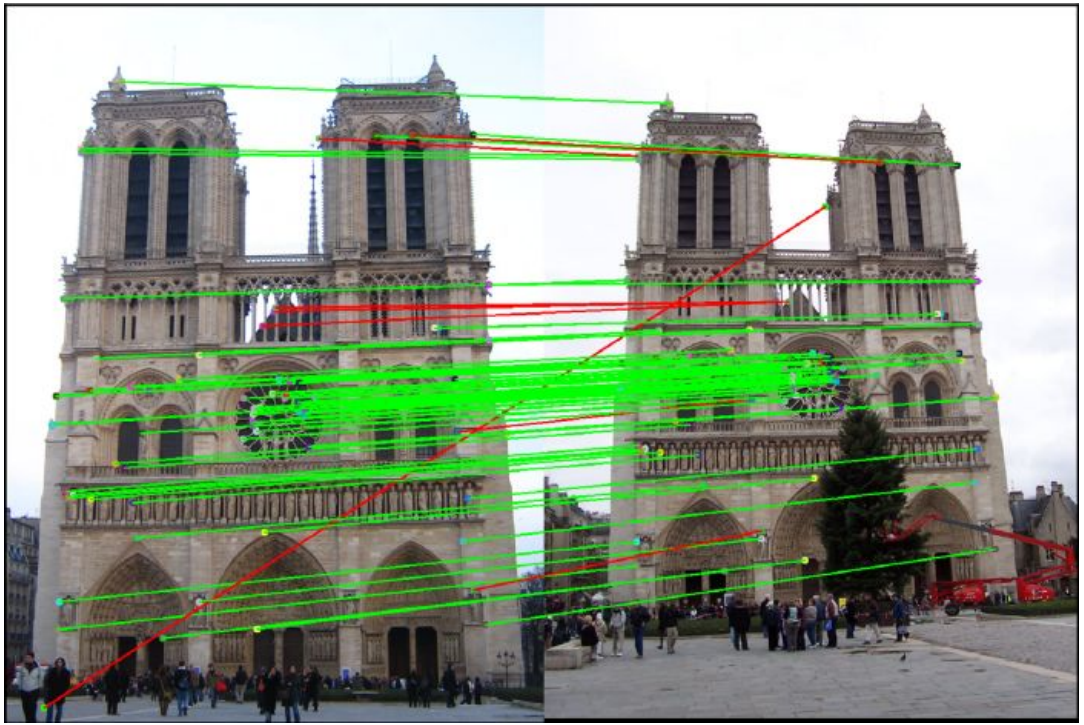
The fundamental Matrix can be computed by 8-point algorithm (8 correspondence)

More recently, 5 correspondence [1]

[1] David Niste´r, An Efficient Solution to the Five-Point Relative Pose Problem, in PAMI 2004

5. RANSAC Algorithm

Recall feature meaturing:

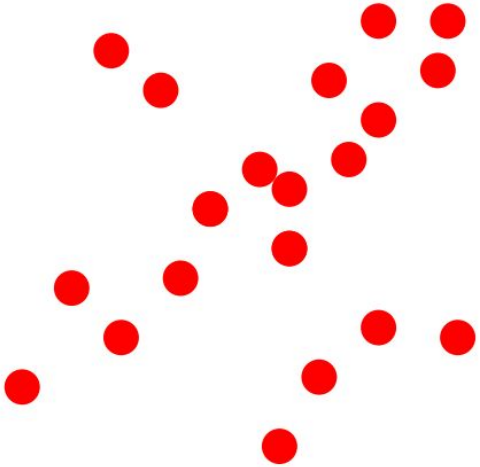


The incorrectly matched feature points are called **outliers**

We want less outliers!

5. RANSAC Algorithm

RANdom SAmple Consensus (RANSAC) algorithm in general:
Try to fit a line with some data (contains outliers)

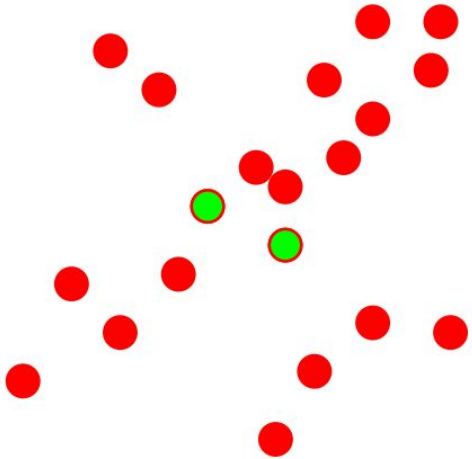


1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

5. RANSAC Algorithm

RANdom SAmple Consensus (RANSAC) algorithm in general:
Try to fit a line with some data (contains outliers)

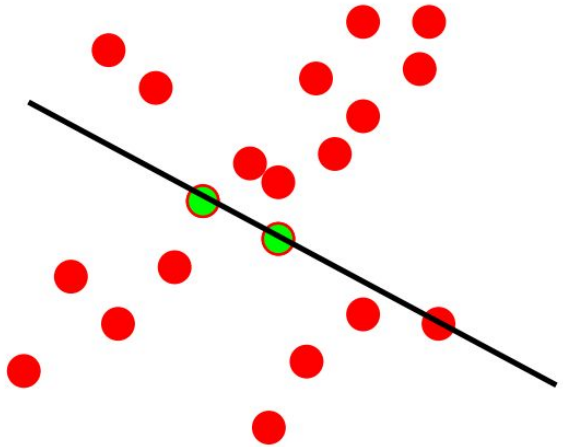


1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

5. RANSAC Algorithm

RANdom SAmple Consensus (RANSAC) algorithm in general:
Try to fit a line with some data (contains outliers)

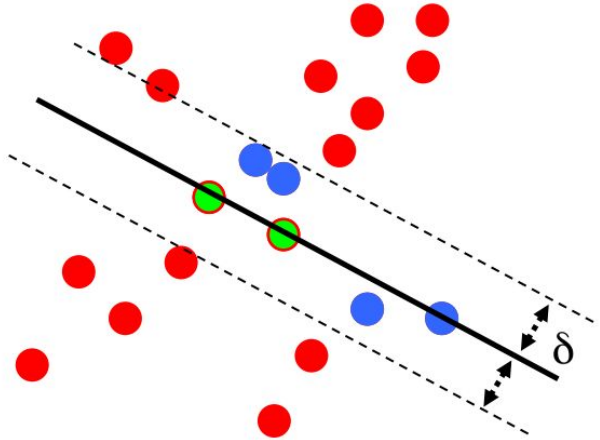


1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

5. RANSAC Algorithm

RANdom SAmple Consensus (RANSAC) algorithm in general:
Try to fit a line with some data (contains outliers)



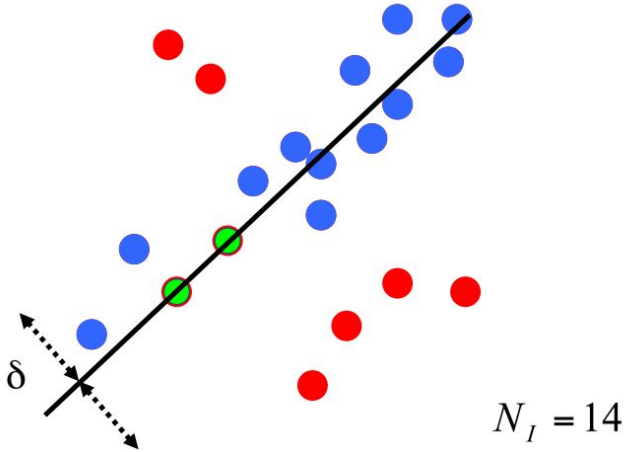
Number of **inliers** $N_I = 6$

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence

5. RANSAC Algorithm

RANdom SAmple Consensus (RANSAC) algorithm in general:
Try to fit a line with some data (contains outliers)



1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

5. RANSAC Algorithm

RANdom SAmple Consensus (RANSAC) algorithm for computing fundamental matrix F:

1. In Each iteration:

randomly choose 8 points correspondences from all points correspondences

2. Compute fundamental matrix F from this 8 points

3. Compute the number of inliers:

search all N points correspondence, keep those point correspondence which satisfy

$$\tilde{x'}^T F \tilde{x} \leq \text{Threshold}$$

4. Choose the F with the max number of inliers

6. Solve camera pose from Essential matrix

Take SVD for essential matrix E

$$\mathbf{E} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^T$$

The essential matrix must have two singular values which are equal and another which is zero

$$\mathbf{\Sigma} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Define matrix

$$\mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{W}^{-1} = \mathbf{W}^T = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Translation t and rotation R can be computed as

$$[\mathbf{t}]_\times = \mathbf{U}\,\mathbf{W}\,\mathbf{\Sigma}\,\mathbf{U}^T$$

$$\mathbf{R} = \mathbf{U}\,\mathbf{W}^{-1}\,\mathbf{V}^T$$

7. Feature Detector and Descriptor

Introduction

- Mainly about point features
- Point features can be used to find a sparse set of corresponding in different images

What kinds of feature points might one use to establish a set of correspondences between these images?

Two pairs of images to be matched.

7. Feature Detector and Descriptor

Introduction

- Mainly about point features
- Point features can be used to find a sparse set of corresponding in different images

Two pairs of images to be matched.

What kinds of feature points might one use to establish a set of correspondences between these images?

Corners and edges!

These points are often called interest points

7. Feature Detector and Descriptor

Types of detectors and descriptors

Interested point detector: Harris corner detector, Good Features to Track (Shi-Tomasi Corner Detector)

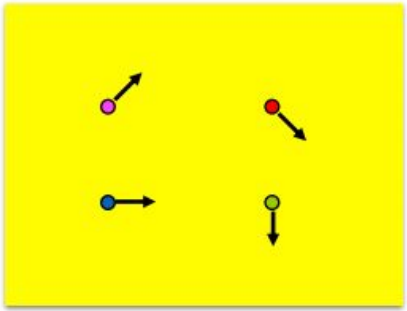
Gradient based detector and descriptor: SIFT, SURF

Binary detector and/or descriptor: FAST (detector), BRIEF (descritor),
ORB (Oriented FAST and Rotated BRIEF)

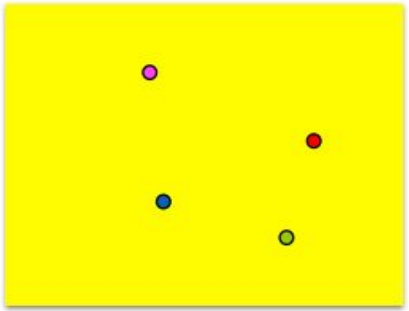
8. Optical Flow and Lucas-Kanade Algorithm

8.1 Optical flow

Estimate the motion of each pixel between two consecutive image frames, based on two assumptions: 1. brightness consistency 2.small motion



$I(x, y, t)$



$I(x, y, t')$

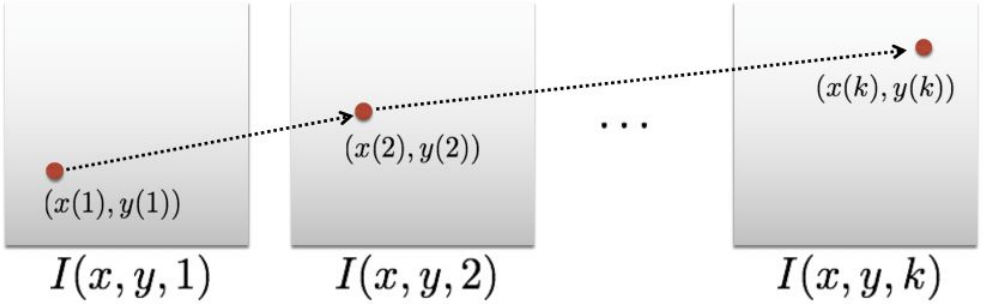
What we want to estimate is the 2D motion vector from time t to time t’

8. Optical Flow and Lucas-Kanade Algorithm

8.1 Optical flow

brightness constancy assumption:

The 2D projection of the same 3D point ‘moves’ across different frames



brightness consistency assumption tells us that the pixel intensity of these 2D projection points does not change

$$I(x(t), y(t), t) = C$$

Where C is a constant

8. Optical Flow and Lucas-Kanade Algorithm

8.2 Lucas-Kanade algorithm

Solve motion vectors:

According to brightness consistency assumption

$$\boldsymbol{I}(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t) = \boldsymbol{I}(x, y, t)$$

where *dx, dy* and *dt* denotes the changes in *x, y* directions and in time *t*

According to small motion assumption, we can do 1st-order Taylor expansion:

$$\boldsymbol{I}\left(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t\right) \approx \boldsymbol{I}\left(x, y, t\right) + \frac{\partial \boldsymbol{I}}{\partial x}\mathrm{d}x + \frac{\partial \boldsymbol{I}}{\partial y}\mathrm{d}y + \frac{\partial \boldsymbol{I}}{\partial t}\mathrm{d}t$$

So we have,

$$\frac{\partial \boldsymbol{I}}{\partial x}\mathrm{d}x + \frac{\partial \boldsymbol{I}}{\partial y}\mathrm{d}y + \frac{\partial \boldsymbol{I}}{\partial t}\mathrm{d}t = 0$$

8. Optical Flow and Lucas-Kanade Algorithm

8.2 Lucas-Kanade algorithm

Then we have

$$\frac{\partial \boldsymbol{I}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t}+\frac{\partial \boldsymbol{I}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t}=-\frac{\partial \boldsymbol{I}}{\partial t}$$

Let's explain what these variables mean:

$\frac{\partial \boldsymbol{I}}{\partial x}$ $\frac{\partial \boldsymbol{I}}{\partial y}$ are the gradients of pixel intensity in x and y directions, respectively

$\frac{\mathrm{d}x}{\mathrm{d}t}$ $\frac{\mathrm{d}y}{\mathrm{d}t}$ are velocities of the motion of the pixel in x and y directions, respectively

$\frac{\partial \boldsymbol{I}}{\partial t}$ is the gradient of the pixel intensity over time

8. Optical Flow and Lucas-Kanade Algorithm

8.2 Lucas-Kanade algorithm

Then we do a different version of the representations for these variables:

$$I_x u + I_y v = -I_t$$

Where I_x, I_y are gradients of pixel intensity, u and v velocities in x and y directions, I_t the gradient of the pixel intensity over time.

In vector form we have,

$$\begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\boldsymbol{I}_t$$

8. Optical Flow and Lucas-Kanade Algorithm

8.2 Lucas-Kanade algorithm

We don't use just one pixel, usually we use a patch and assume all the pixels in this patch have the same motion.

e.g. a $w \times w = n$ patch, for some pixel k in this patch we have

$$\begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y \end{bmatrix}_k \begin{bmatrix} u \\ v \end{bmatrix} = -\boldsymbol{I}_{tk}, \quad k = 1, \ldots, w^2$$

So for n pixels we have,

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix}$$

8. Optical Flow and Lucas-Kanade Algorithm

8.2 Lucas-Kanade algorithm

We define

$$A = \begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \qquad b = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix}$$

So we have

$$\boldsymbol{A} \begin{bmatrix} u \\ v \end{bmatrix} = -\boldsymbol{b}$$

The solution for u, v vector?

8. Optical Flow and Lucas-Kanade Algorithm

8.2 Lucas-Kanade algorithm

Two unknowns need to equations.

But what if we have more than two equations?

$$\begin{bmatrix} u \\ v \end{bmatrix}^{*} = -\left(\boldsymbol{A}^{T}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{T}\boldsymbol{b}$$

This is the least square solution for this **overdetermined** linear equation

and $\left(\boldsymbol{A}^{T}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{T}$ is called **Pseudoinverse** or **Moore-Penrose inverse** of matrix $\boldsymbol{A}$

This algorithm is called **Lucas-Kanade algorithm** [1]

Lucas-Kanade algorithm can be used to track key-points in visual SLAM.

[1] Simon Baker and Iain Matthews, Lucas-Kanade 20 Years On: A Unifying Framework, in International Journal of Computer Vision 56(3), 221–255, 2004

9. Direct Methods for Camera Pose and Depth Estimation

Drawbacks of feature-based SLAM:

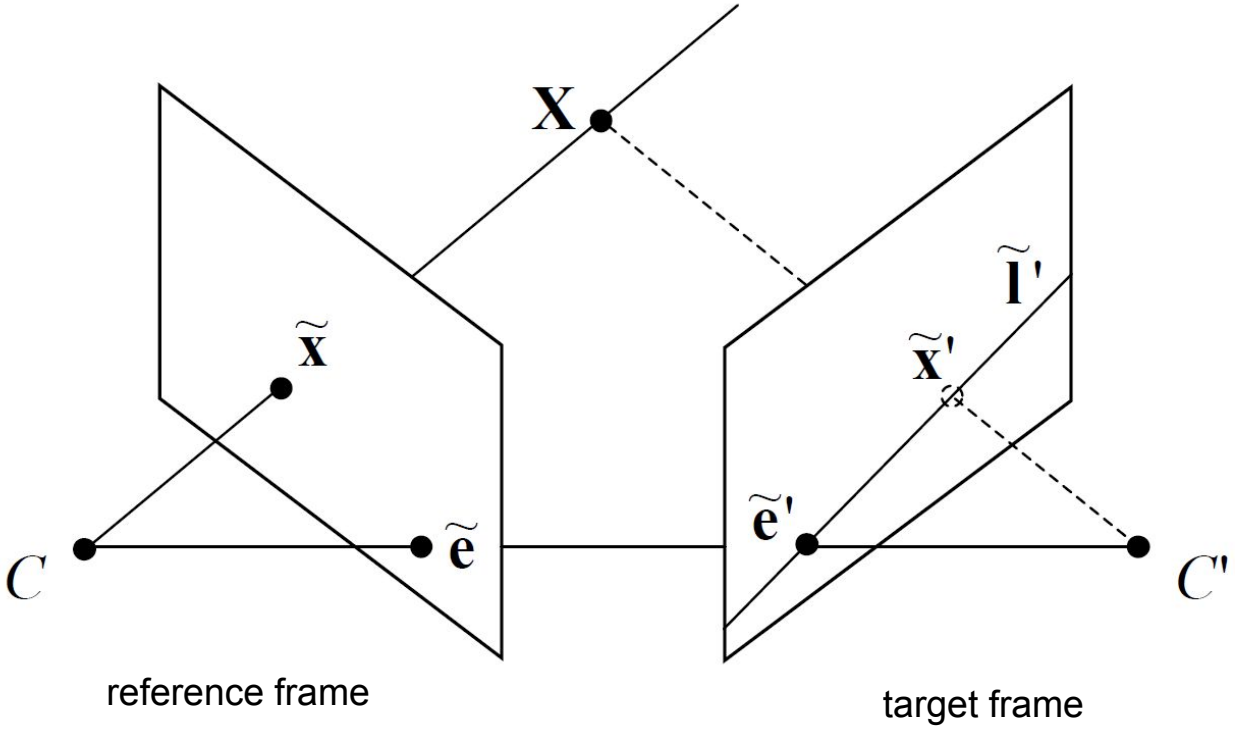
- Computing feature descriptors are time-consuming, especially gradient based features, i.e. SIFT
- Discard many useful information in the images
- Impossible for dense reconstruction

Possible solutions:

- Use raw image pixel for matching after computing feature detectors (semi-direct method)
- Direct matching raw pixels without even feature detectors (direct method)

9. Direct Methods for Camera Pose and Depth Estimation

Back to the two-view geometry figure:



9. Direct Methods for Camera Pose and Depth Estimation

For a point in 3D world coordinate system

$$X_W = [X, Y, Z]^T$$

Its projection on reference image frame is,

$$\tilde{x}_{im\_ref} = \frac{1}{\widehat{Z}} K X_W$$

The 3D point can also be expressed as

$$X_W = \widehat{Z} K^{-1} \tilde{x}_{im\_ref}$$

Then project the 3D point to another frame with estimated R, t and Z

$$\tilde{x}_{im\_warp} = K[\widehat{R}|\widehat{t}]\widehat{Z}K^{-1}\tilde{x}_{im\_ref}$$

This will form a image if we have multiple 3D points and its projection. This image is formed by 'moving' the original pixel in reference image to the new 2D position in image coordinate.

This process is called **warping**.

9. Direct Methods for Camera Pose and Depth Estimation

According to brightness constancy assumption, if we have perfect estimation of R, t and Z, the warped image and target image should be the same.

However, it's not possible, so we have the following loss called **photometric loss:**

$$L = \frac{1}{HW} \sum_{i=1}^{HW} |I_{target}(i) - I_{warp}(i)|$$

Where $I_{ref}(i)$ and $I_{warp}(i)$ are the pixel intensities of the two images.

By minimizing the **photometric loss ,** we can optimize the estimated R, t and Z

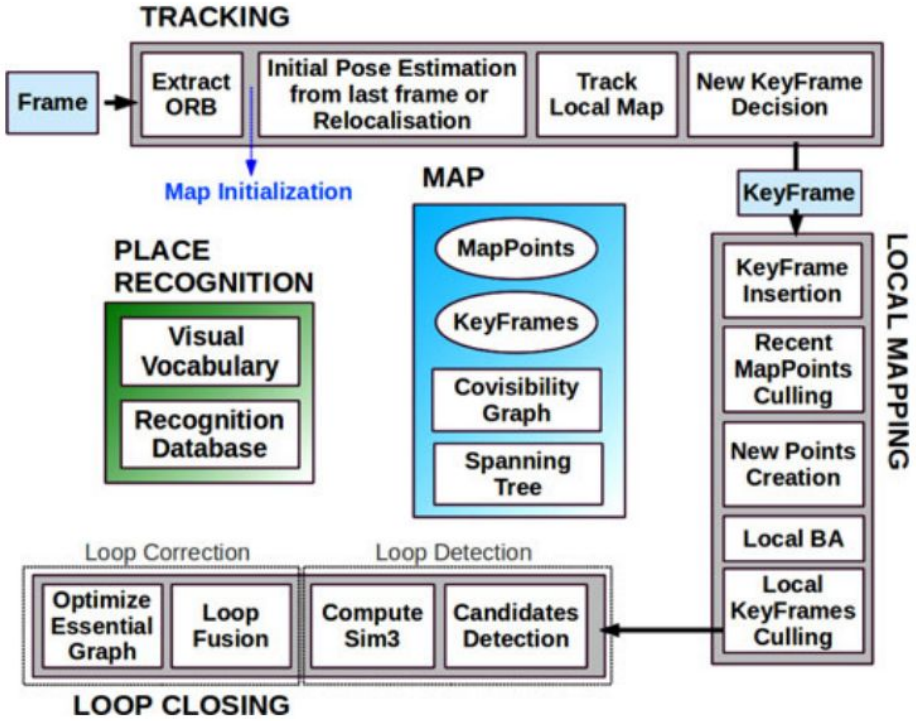
How to minimize this loss?
This optimization problem is a typical non-linear least optimization problem
Solutions are Gauss-Newton and Levenberg–Marquardt algorithms

Part 2

Traditional Feature based and Direct VO/VSLAM

1. Feature based: ORB-SLAM

Overview of ORB-SLAM



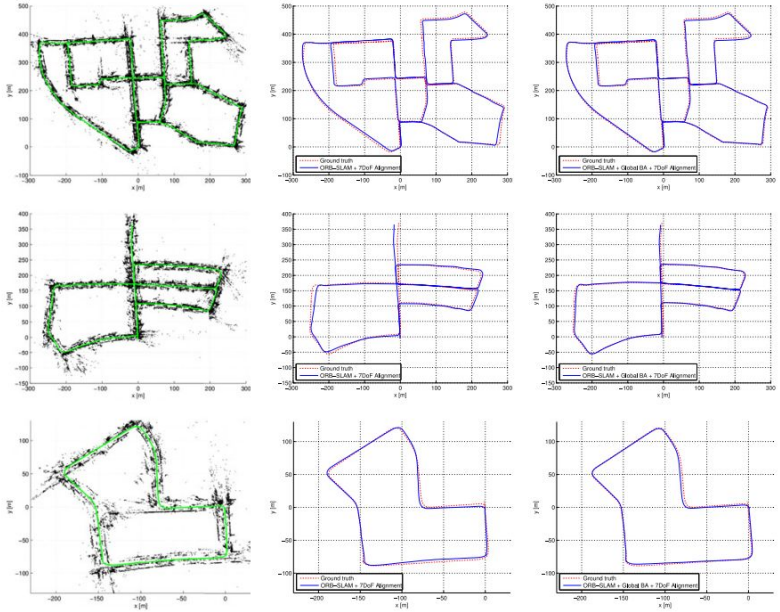
Bundle Adjustment

Raul Mur-Artal, J. M. M. Montiel ´and Juan D. Tardos, ORB-SLAM: A Versatile and Accurate Monocular SLAM System, TRO 2015

1. Feature based: ORB-SLAM

Results

Feature matching for different scale and for dynamic objects

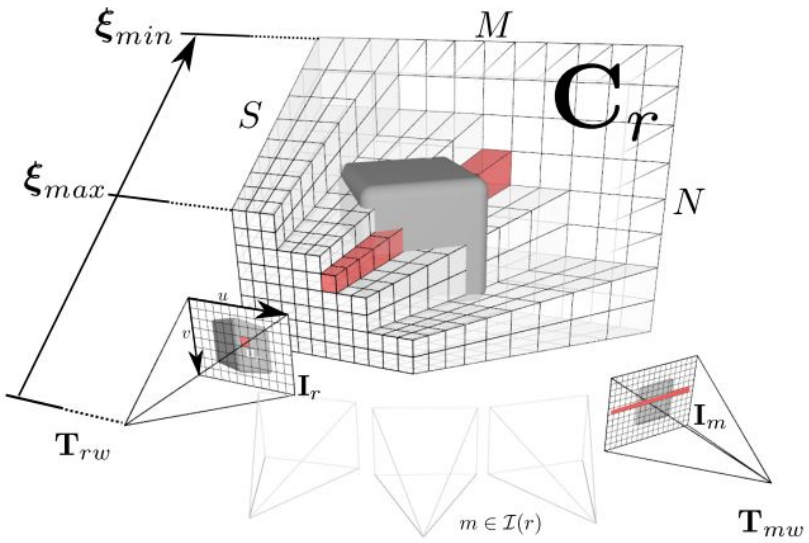


Results on KITTI odometry dataset

Raul Mur-Artal, J. M. M. Montiel ´and Juan D. Tardos, ORB-SLAM: A Versatile and Accurate Monocular SLAM System, TRO 2015

2. Dense Direct Method: DTAM

Depth estimation:



projective photometric cost volume:

$$\mathbf{C}_r(\mathbf{u}, d) = \frac{1}{|\mathcal{I}(r)|} \sum_{m \in \mathcal{I}(r)} \|\rho_r \left(\mathbf{I}_m, \mathbf{u}, d\right)\|_1 ,$$

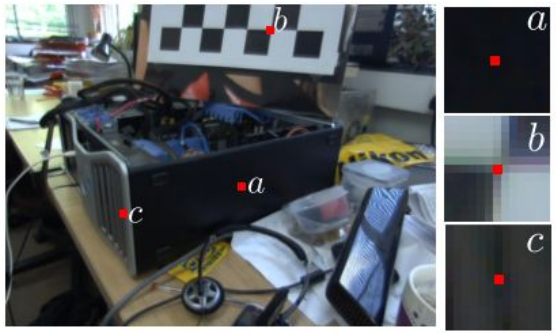
Pixel-level photometric error:

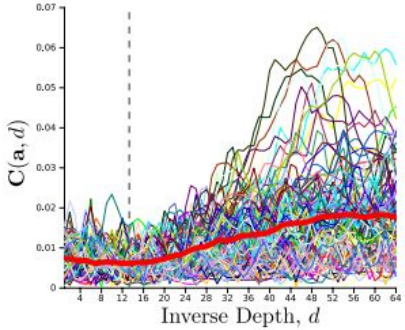
$$\rho_r \left(\mathbf{I}_m, \mathbf{u}, d\right) = \mathbf{I}_r \left(\mathbf{u}\right) - \mathbf{I}_m \left(\pi \left(\mathrm{K}\mathrm{T}_{mr}\pi^{-1} \left(\mathbf{u}, d\right)\right)\right)$$

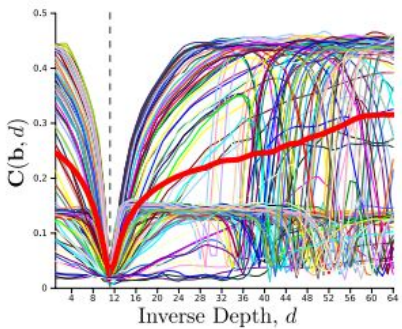
Richard A. Newcombe, Steven J. Lovegrove and Andrew J. Davison, DTAM: Dense Tracking and Mapping in Real-Time

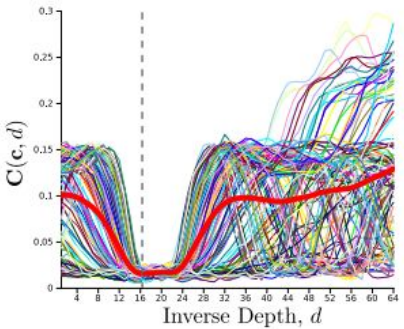
2. Dense Direct Method: DTAM

Plots for the single pixel photometric functions ρ(u) and the resulting total data cost C(u)









So even direct method need texture!

Richard A. Newcombe, Steven J. Lovegrove and Andrew J. Davison, DTAM: Dense Tracking and Mapping in Real-Time

2. Dense Direct Method: DTAM

Incremental cost volume construction:

The current inverse depth map extracted as the current minimum cost for each pixel row as 2, 10 and 30 overlapping images are used in the data term (from left)

Richard A. Newcombe, Steven J. Lovegrove and Andrew J. Davison, DTAM: Dense Tracking and Mapping in Real-Time

2. Direct Method: DTAM

Camera pose tracking by minimizing the following term:

$$F(\boldsymbol{\psi}) = \frac{1}{2}\sum_{\mathbf{u}\in\Omega}\left(f_{\mathbf{u}}\left(\boldsymbol{\psi}\right)\right)^2 = \frac{1}{2}\|f(\boldsymbol{\psi})\|_2^2,$$

$$f_{\mathbf{u}}(\boldsymbol{\psi}) = \mathbf{I}_l\left(\pi\left(\mathrm{KT}_{lv}(\boldsymbol{\psi})\pi^{-1}\left(\mathbf{u},\xi_v\left(\mathbf{u}\right)\right)\right)\right) - \mathbf{I}_v\left(\mathbf{u}\right)$$

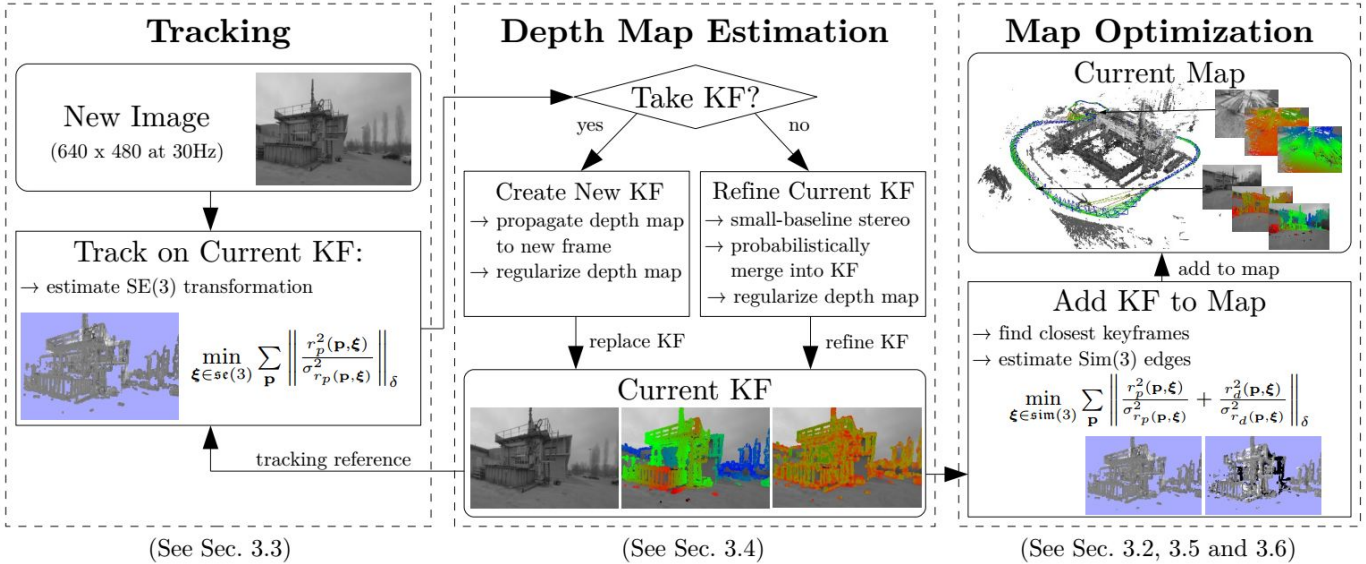
$$\mathrm{T}_{lv}(\boldsymbol{\psi}) = \exp\left(\sum_{i=1}^{6}\boldsymbol{\psi}_i \underset{\mathbb{SE}(3)}{\mathrm{gen}_i}\right).$$

The whole DTAM system estimate depth and pose in an interleaved manner

Richard A. Newcombe, Steven J. Lovegrove and Andrew J. Davison, DTAM: Dense Tracking and Mapping in Real-Time

3. Semi-Dense Direct Method: LSD-SLAM

Framework



Jakob Engel and Thomas Schops and Daniel Cremers, LSD-SLAM: Large-Scale Direct Monocular SLAM, ECCV 2016

3. Semi-Dense Direct Method: LSD-SLAM

Weighted Photometric loss:

$$E(\boldsymbol{\xi}) = \sum_i \underbrace{\left(I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \boldsymbol{\xi}))\right)^2}_{=:r_i^2(\boldsymbol{\xi})},$$

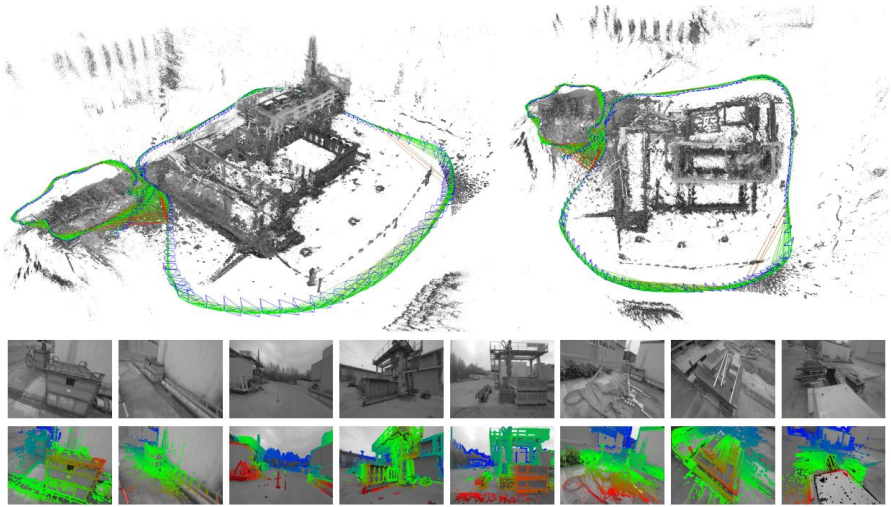
$$E(\boldsymbol{\xi}) = \sum_i w_i(\boldsymbol{\xi}) r_i^2(\boldsymbol{\xi}),$$

Jakob Engel and Thomas Schops and Daniel Cremers, LSD-SLAM: Large-Scale Direct Monocular SLAM, ECCV 2016

1. Introduction to direct SLAM/VO

1.2 Semi-Dense Direct Method: LSD-SLAM

Qualitative results: mapping and semi-dense map estimation



Semi-dense means only using pixels with high gradients (edges and smooth intensity variations regions)

Jakob Engel and Thomas Schops and Daniel Cremers, LSD-SLAM: Large-Scale Direct Monocular SLAM, ECCV 2016

Part 3

Learning based VO

1. Supervised method

1.1 PoseNet

Approach: Camera pose regression with CNN
Camera pose represented as translation x and rotation q in quaternion form

$$\mathbf{p} = [\mathbf{x}, \mathbf{q}]$$

With ground truth supervision, we get the training loss:

$$loss(I) = \|\hat{\mathbf{x}} - \mathbf{x}\|_2 + \beta \left\| \hat{\mathbf{q}} - \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\|_2$$

The network architecture is modified from GoogLeNet

Alex Kendall, Matthew Grimes, Roberto Cipolla, PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization, in ICCV 2015

1. Supervised method

1.1 PoseNet

Experimental results:

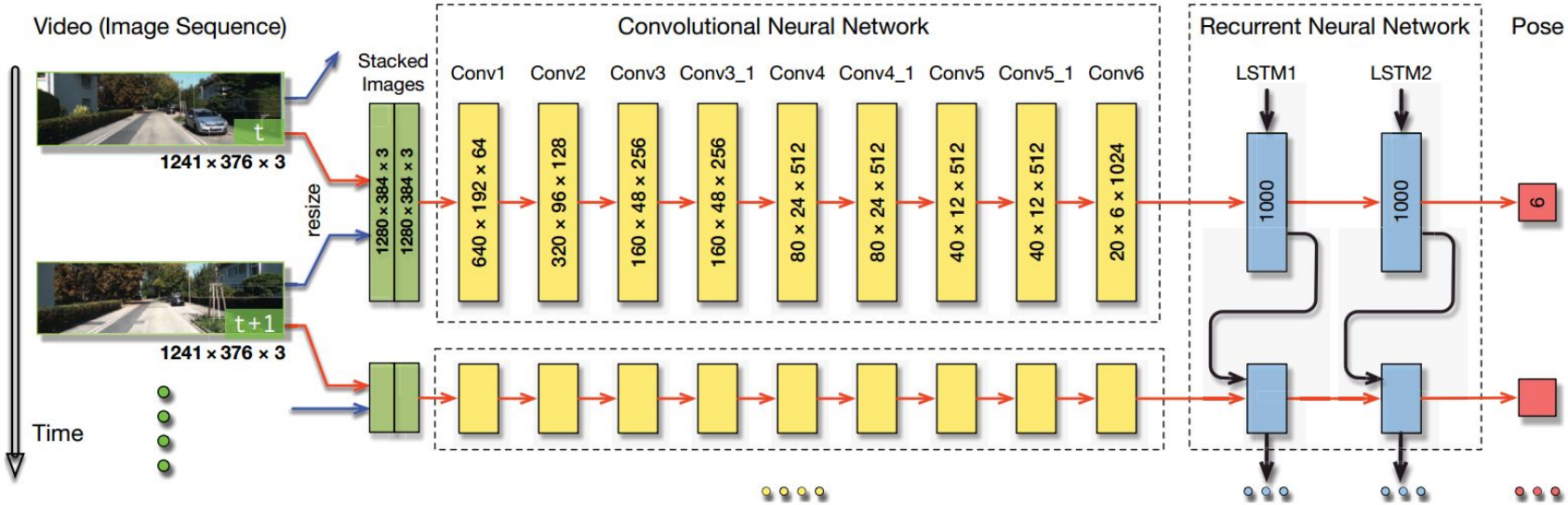
King's College *Street* *Old Hospital* *Shop Façade* *St Mary's Church*

Figure 4: **Map of dataset** showing training frames (green), testing frames (blue) and their predicted camera pose (red). The testing sequences are distinct trajectories from the training sequences and each scene covers a very large spatial extent.

Alex Kendall, Matthew Grimes, Roberto Cipolla, PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization, in ICCV 2015

1. Supervised method

1.2 DeepVO

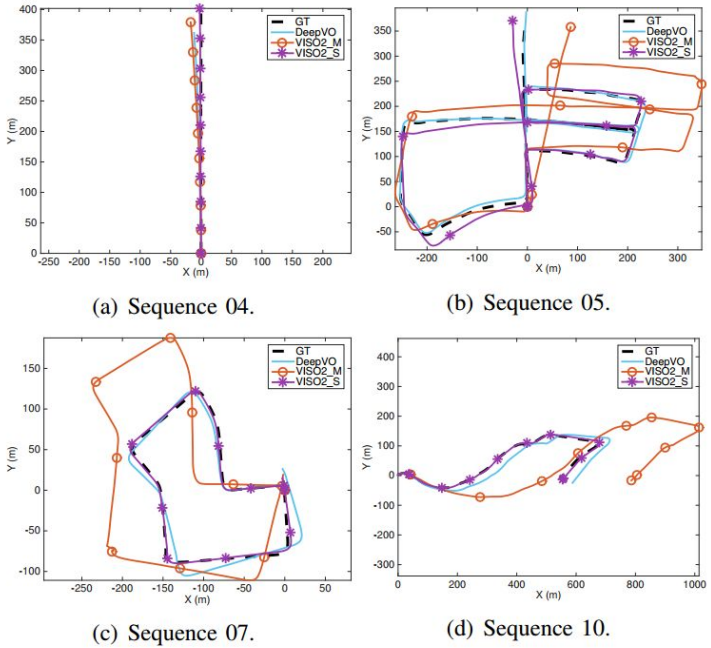


Network architecture

Sen Wang, Ronald Clark, Hongkai Wen, Niki Trigoni, DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks, in ICRA 2017

1. Supervised method

1.2. DeepVO
Experimental Results



(a) Sequence 04. (b) Sequence 05.

(c) Sequence 07. (d) Sequence 10.

Camera pose estimation results in KITTI VO datasets

Sen Wang, Ronald Clark, Hongkai Wen, Niki Trigoni, DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks, in ICRA 2017

2. Self-supervised method

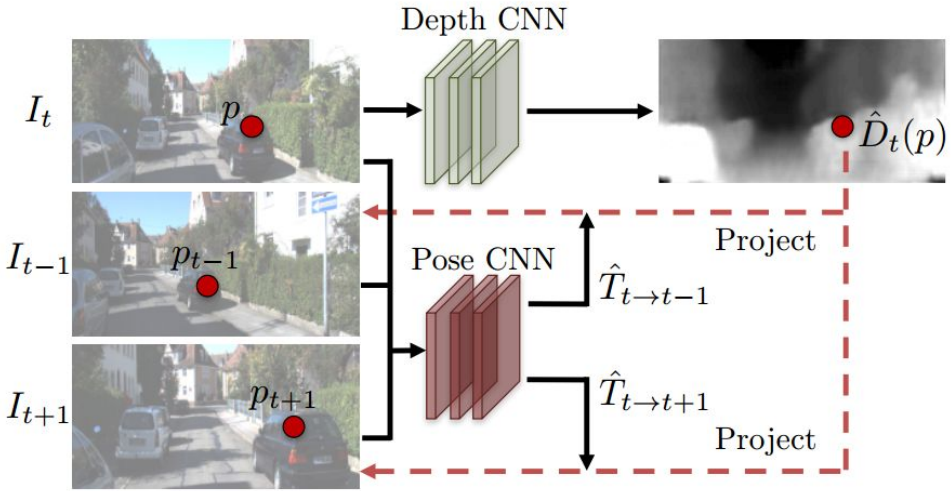
SfM-Learner

Tinghui Zhou, Matthew Brown, Noah Snavely, David G. Lowe, Unsupervised Learning of Depth and Ego-Motion from Video, CVPR 2017

2. Self-supervised method

SfM-Learner

Warping and photometric loss:



$$p_s \sim K\hat{T}_{t\to s}\hat{D}_t(p_t)K^{-1}p_t$$

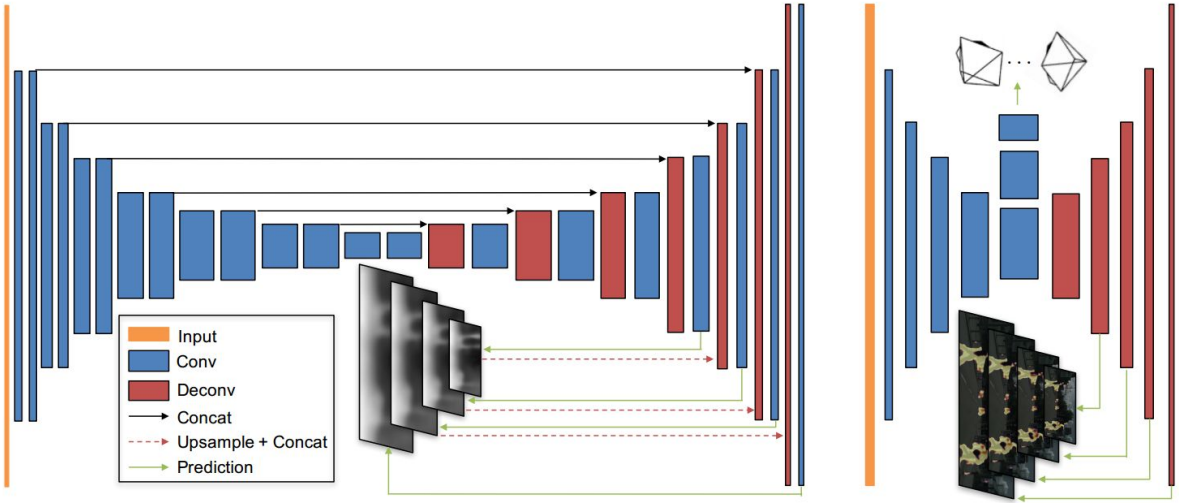
$$\mathcal{L}_{vs} = \sum_s \sum_p |I_t(p) - \hat{I}_s(p)|$$

Tinghui Zhou, Matthew Brown, Noah Snavely, David G. Lowe, Unsupervised Learning of Depth and Ego-Motion from Video, CVPR 2017

2. Self-supervised method

SfM-Learner

Network structures:



(a) Single-view depth network

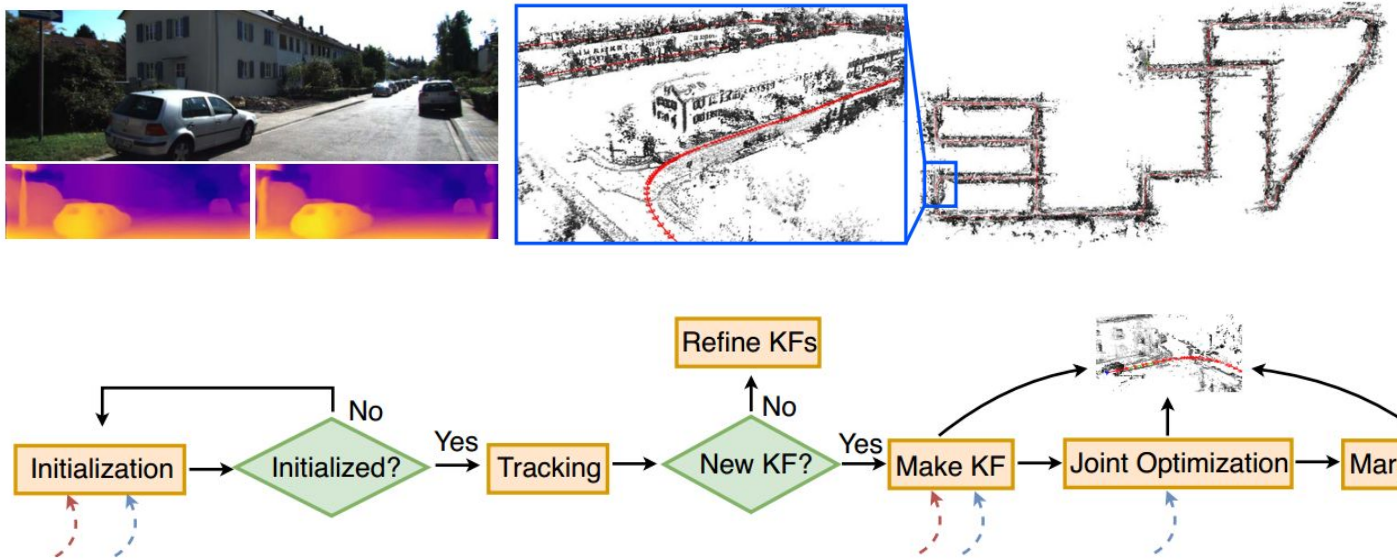
(b) Pose/explainability network

Tinghui Zhou, Matthew Brown, Noah Snavely, David G. Lowe, Unsupervised Learning of Depth and Ego-Motion from Video, CVPR 2017

3. Hybrid method

DVSO

Overview

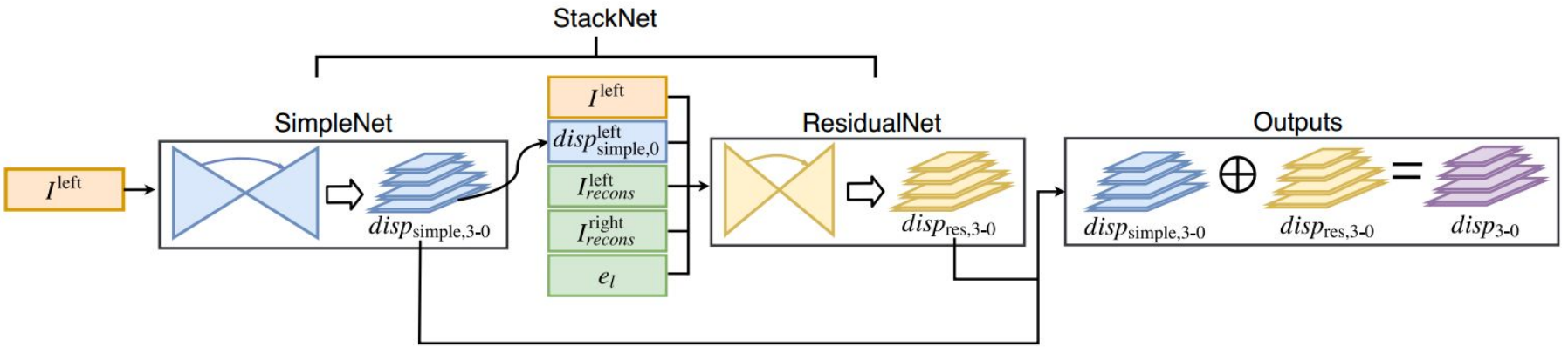


Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers, Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry, ECCV 2018

3. Hybrid method

DVSO

Depth estimation

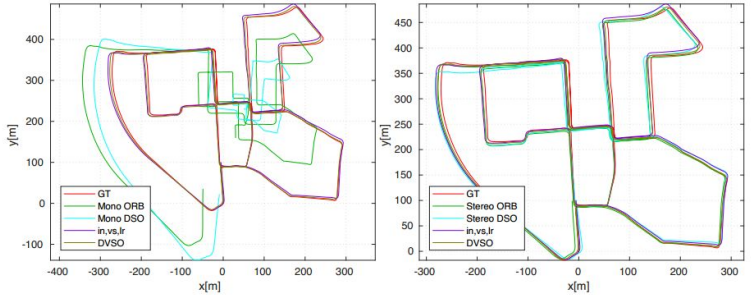


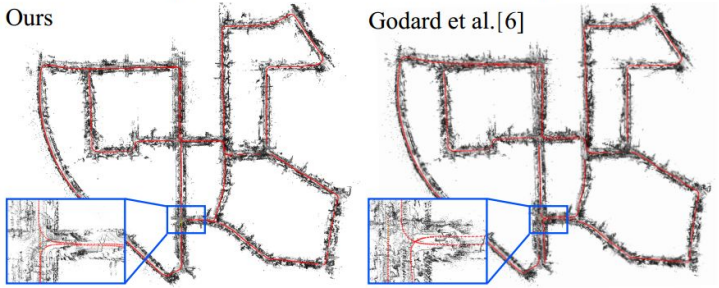
Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers, Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry, ECCV 2018

3. Hybrid method

DVSO

Pose estimation on KITTI:





Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers, Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry, ECCV 2018

Recommended Open Source Visual Odometry/SLAM Implementations

Tutorial Purpose: PySLAM: https://github.com/luigifreda/pyslam

Feature based: ORB-SLAM 2: https://github.com/raulmur/ORB_SLAM2

Direct Method: DSO: https://github.com/JakobEngel/dso

Learning based: SfM-Learner: https://github.com/ClementPinard/SfmLearner-Pytorch