VisualSLAM – A 4 hours' Introduction

Danping Zou
Assistant Professor
dpzou@sjtu.edu.cn
http://drone.sjtu.edu.cn/dpzou
Lab of Navigation and Location-based Service
Shanghai Jiao Tong University

2017, 25th July, ROS Summer School @ECNU

What's SLAM

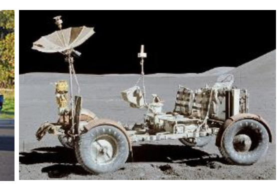
- **SLAM** :**S**imultaneous **L**ocalization **A**nd **M**apping.

Autonomous robot:

Where to go ?

How to move ?

Constructing a map of an unknown environment while simultaneously keeping track of the robot's location.

SLAM with different kinds of sensors

2D laser rangefinder

3D liDar

RGB-D camera

- High precision
- Bulky

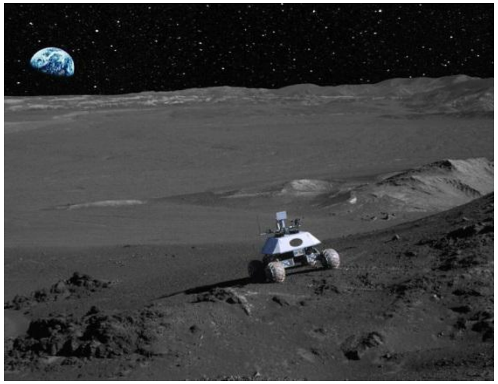
Since 2005, there has been intense research into VSLAM (**Visual SLAM**) using primarily visual (camera) sensors.

- Passive sensing
- Light & compact
- Energy saving
- Ubiquity

Visual SLAM

- Input sensor are video cameras
- build 3D point clouds
- Estimate the self-pose of the camera
- both are processed in **real time**

1. A complementary to other sensors:
   - GPS
   - IMU
   - Laser range finder
2. Works in GPS-denied environments
   - Indoor
   - Cave
   - Mars, Moon

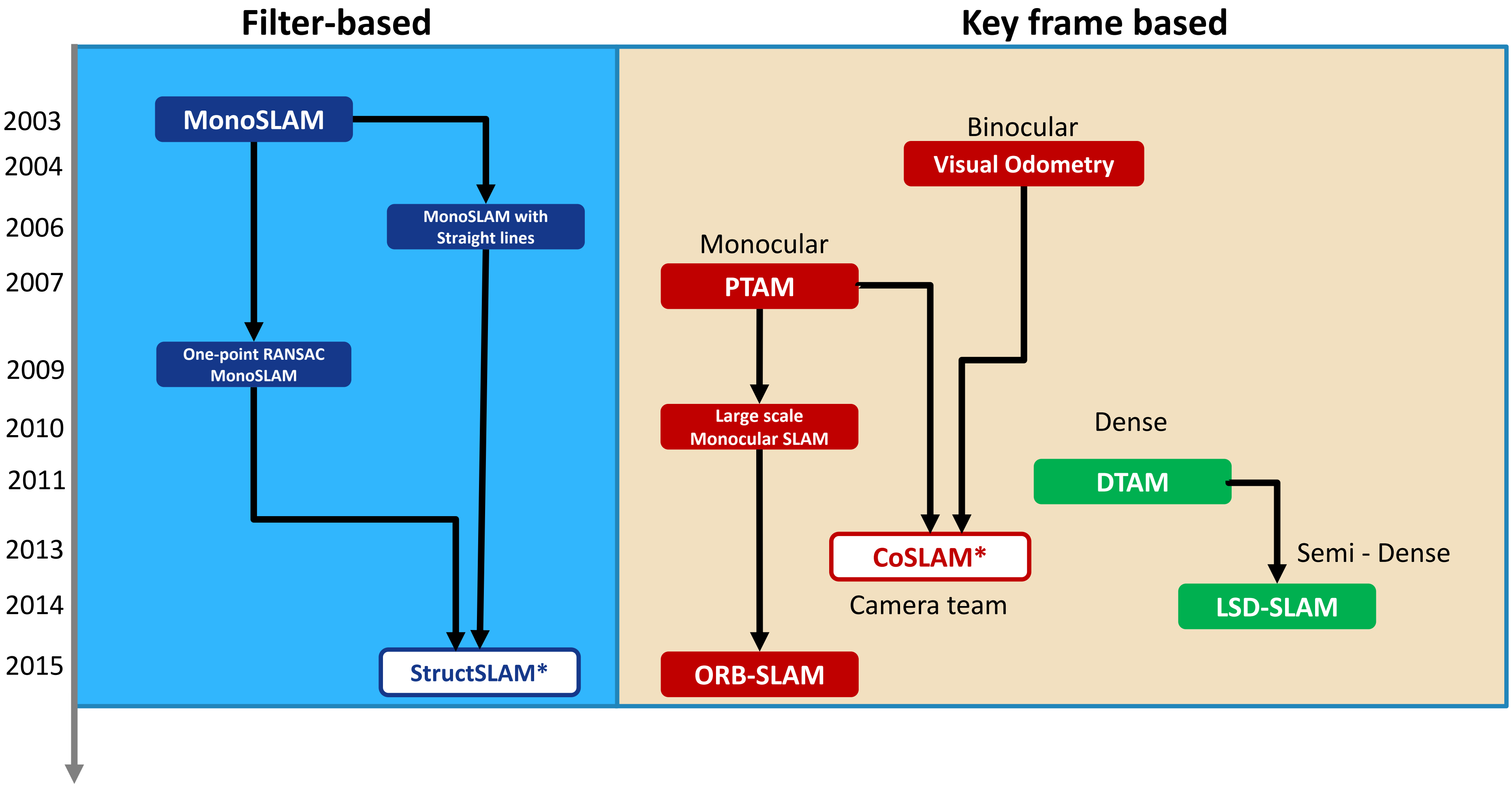


Other applications

- Inside-out motion tracking in AR/VR

- Get the position and attitude of the camera
- Put the virtual object into the scene

Development of Visual SLAM in recent ten years



Open source software for VisualSLAM

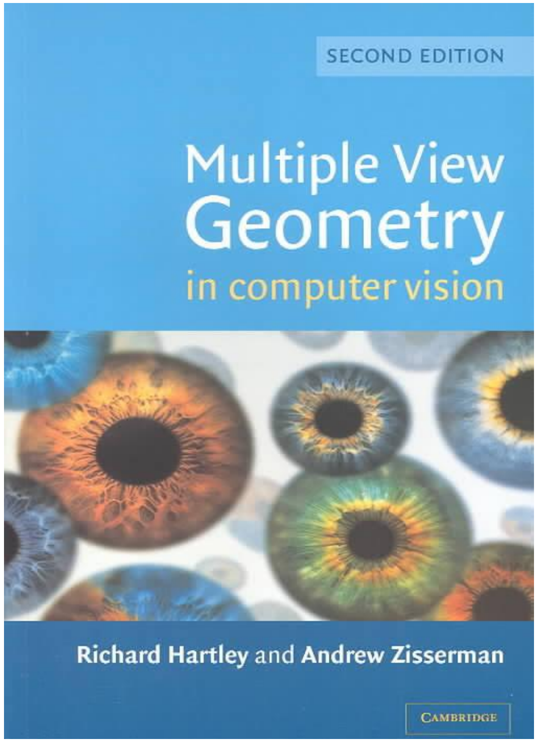
- PTAM(http://www.robots.ox.ac.uk/~gk/PTAM/)
- ORBSLAM(http://webdiis.unizar.es/~raulmur/orbslam/)
- MonoSLAM(http://webdiis.unizar.es/~jcivera/code/1p-ransac-ekf-monoslam.html)
- VisualSFM (http://ccwu.me/vsfm/)
- CoSLAM (https://github.com/danping/CoSLAM)

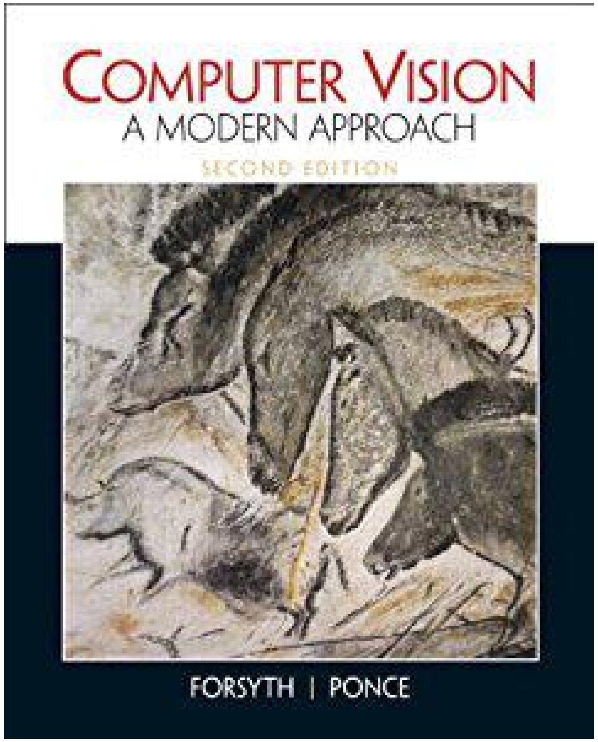
Outline

- Basic Theory
  - Projective geometry
  - Pinhole camera model
  - Camera calibration
  - Two camera geometry

- Design a typical Visual SLAM system
- Two Visual SLAM systems:
  - Extended Kalman Filter approach:
    - StructSLAM
  - Visual SLAM for a group of robots
    - CoLSAM

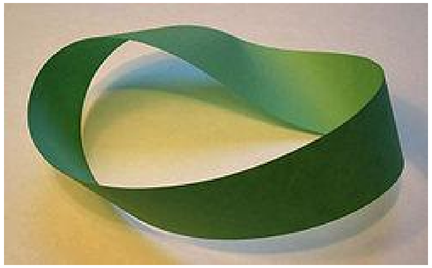
Basic theory





Projective Geometry

- Homogenous coordinates
  - Represent an n-dimensional vector by a $n+1$ dimensional coordinate



$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ w \end{pmatrix} = \lambda \mathbf{x} \sim \begin{pmatrix} x_1/w \\ x_2/w \\ \dots \\ x_n/w \end{pmatrix}$$

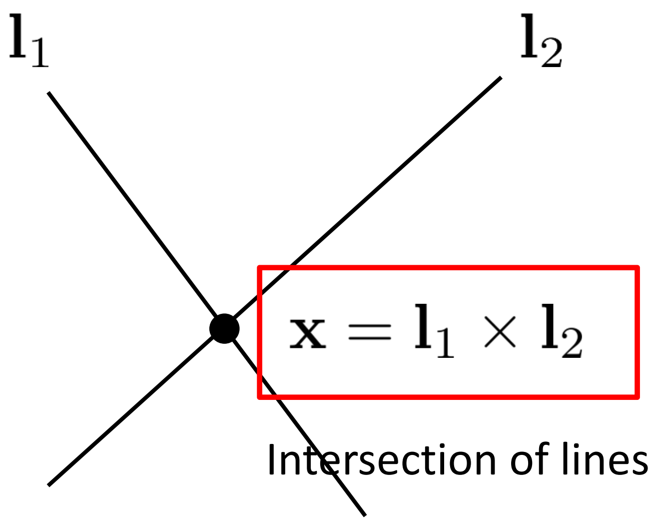
Homogenous coordinate Cartesian coordinate

  - Can represent infinite points or lines

August Ferdinand Möbius
1790-1868

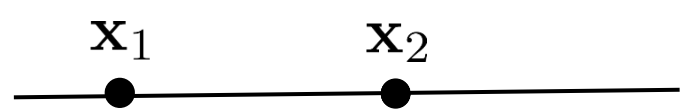
Homogenous coordinates

- 2D points and lines
  - A point is represented by $\mathbf{x} = (x, y, 1)^T$
  - A line is represented by $\mathbf{l} = (a, b, c)^T$
  - A line equation is $\mathbf{l}^T\mathbf{x} = 0$



$\boxed{\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2}$

Intersection of lines



$\boxed{\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2}$

Line across two points

Here, ‘ $\times$ ‘ represents cross production

Homogenous coordinates

- 2D points and lines
  - An infinite point is represented by $\mathbf{x} = (x, y, 0)^T$
  - All infinite points are on the infinite line

$$\mathbf{l} = (0, 0, 1)^T$$

- 3D points and planes
  - A point is represented by $\mathbf{x} = (x, y, z, w)^T$
  - A plane is represented by $\Pi = (\pi_1, \pi_2, \pi_3, \pi_4)^T$

$$\begin{pmatrix} \Pi_1^T \\ \Pi_2^T \\ \Pi_3^T \end{pmatrix} \mathbf{x} = 0$$

Intersection of three planes

$$\begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{pmatrix} \Pi = 0$$

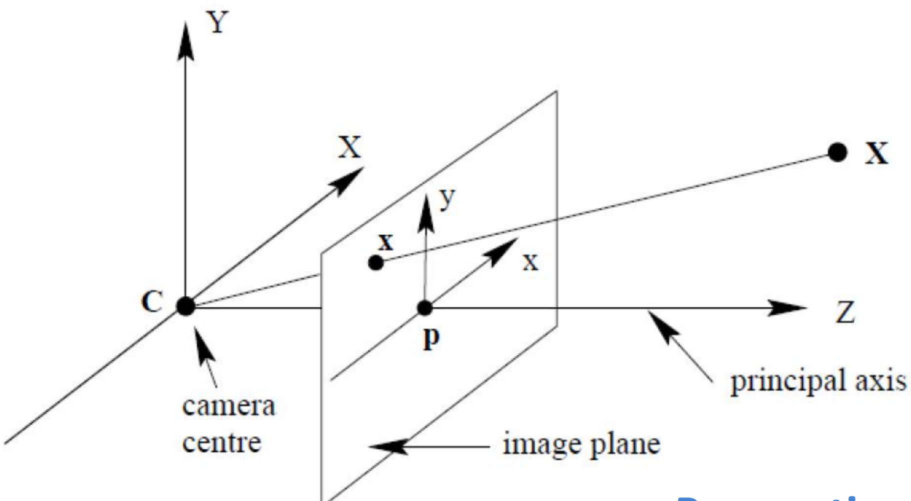
Plane across three points

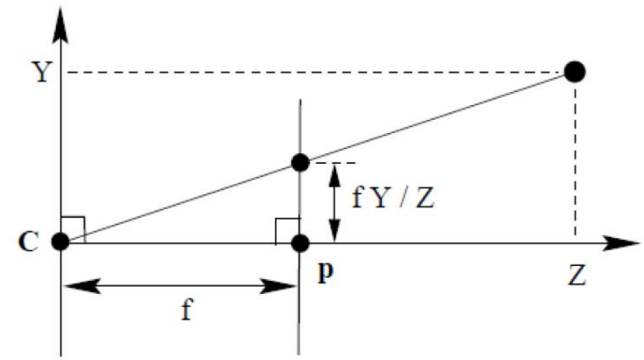
Single Camera Model

- **Pinhole camera model**
- **Camera intrinsic parameters**
- **Distortion model**

Pinhole camera model

- A pinhole camera model is illustrated as the follows





Perspective projection

$$\mathbf{X} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \Rightarrow \mathbf{x} = \begin{pmatrix} fX/Z \\ fY/Z \\ f \end{pmatrix} = \lambda \begin{pmatrix} u \\ v \\ f \end{pmatrix}$$

Pinhole camera model

- We have

$$\begin{pmatrix} u \\ v \\ f \end{pmatrix} \propto \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

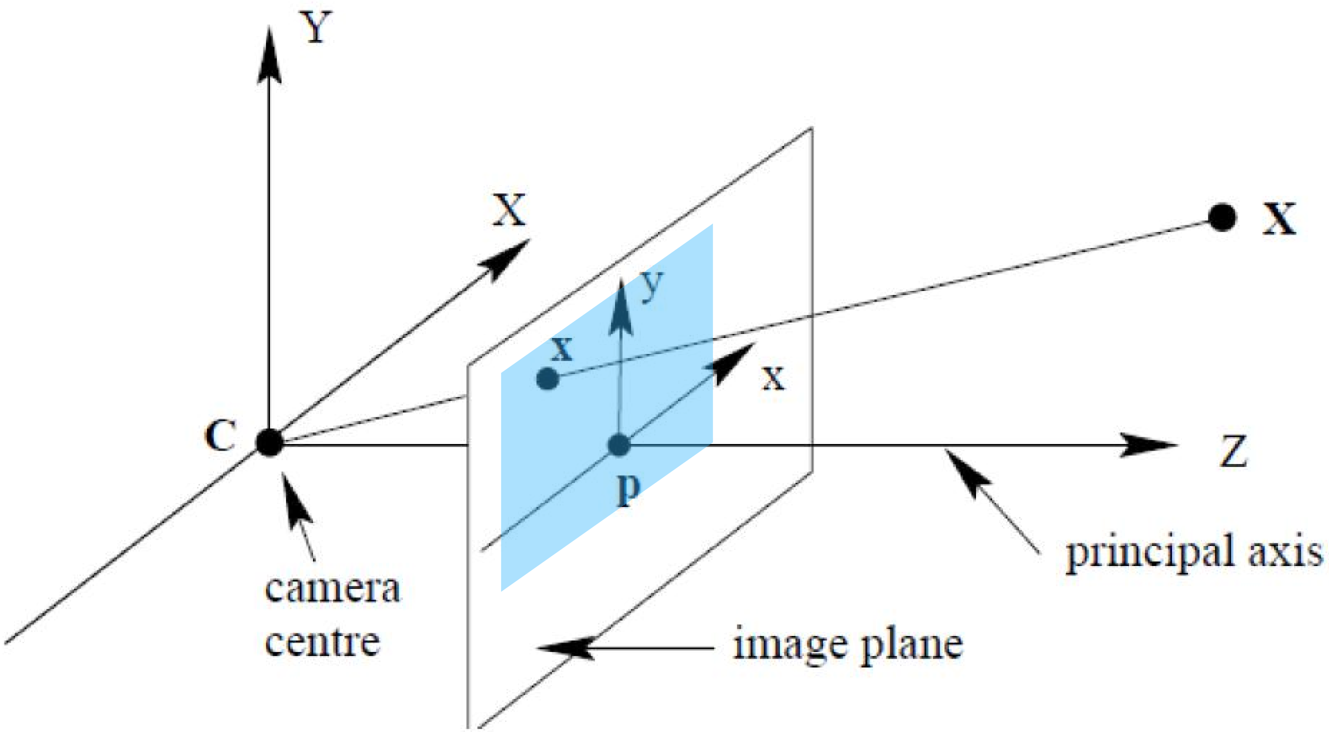
image point   3x4 projection matrix   scene point

$$\downarrow$$

$$[\mathbf{R}\ \mathbf{t}]$$

Pinhole camera model

- Normalized image point will be then sensed by CCD or CMOS :
  – Next step: Image plane -> Sensor frame

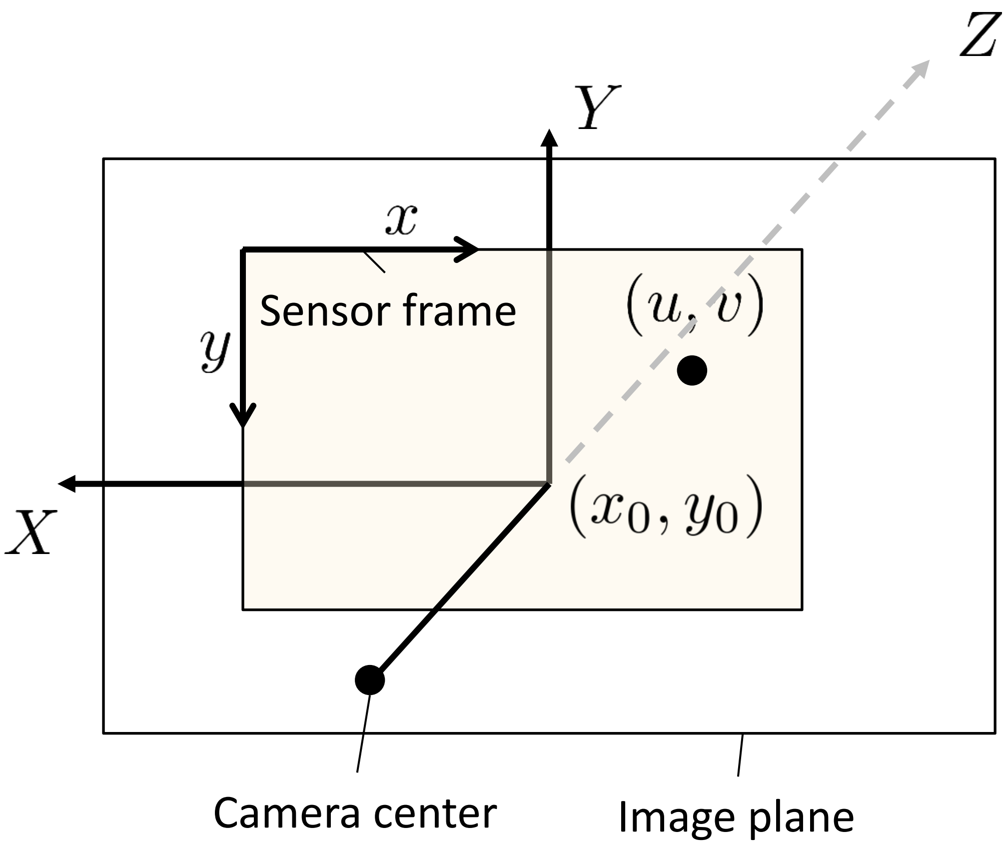


Pinhole camera model

- Intrinsic camera parameters:
  - Image plane -> Sensor frame

$$x = k_x u + x0$$
$$y = k_y u + y0$$

where k_x and k_y are scaling factors, of which the units are ***pixels/length***



Nikon D610 camera:

- Maximum image resolution:
  6016 × 4016
- CMOS size:
  35.9 x 24 mm

We have :

$$k_x = 0.168 pixel/\mu m$$
$$k_y = 0.167 pixel/\mu m$$

Pinhole camera model

- Camera intrinsic matrix (or camera calibration matrix)

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} fk_x & 0 & x_0 \\ 0 & fk_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \frac{1}{f} \begin{pmatrix} u \\ v \\ f \end{pmatrix}$$

$$= \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}$$

$$= \mathbf{K} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}$$

Camera intrinsics

- $\mathbf{K}$ is a 3×3 upper triangle matrix, called camera intrinsic matrix (or camera calibration matrix)

$$\mathbf{K} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- There are four parameters:
  - **Principle point** (x_0, y_0) , which is point where the optical axis intersects the image plane
  - **Scaling factors** α_x, α_y in image x and y directions
    - In most cases the scaling factors are nearly the same, so sometimes only **three** parameters are used – two for principle point and one for scaling.

How to get focus length ??

$$\alpha_x = f k_x \quad \Rightarrow \quad f = \alpha_x / k_x$$

Image distortion

- Due to the imperfect imaging system, images are usually distorted.

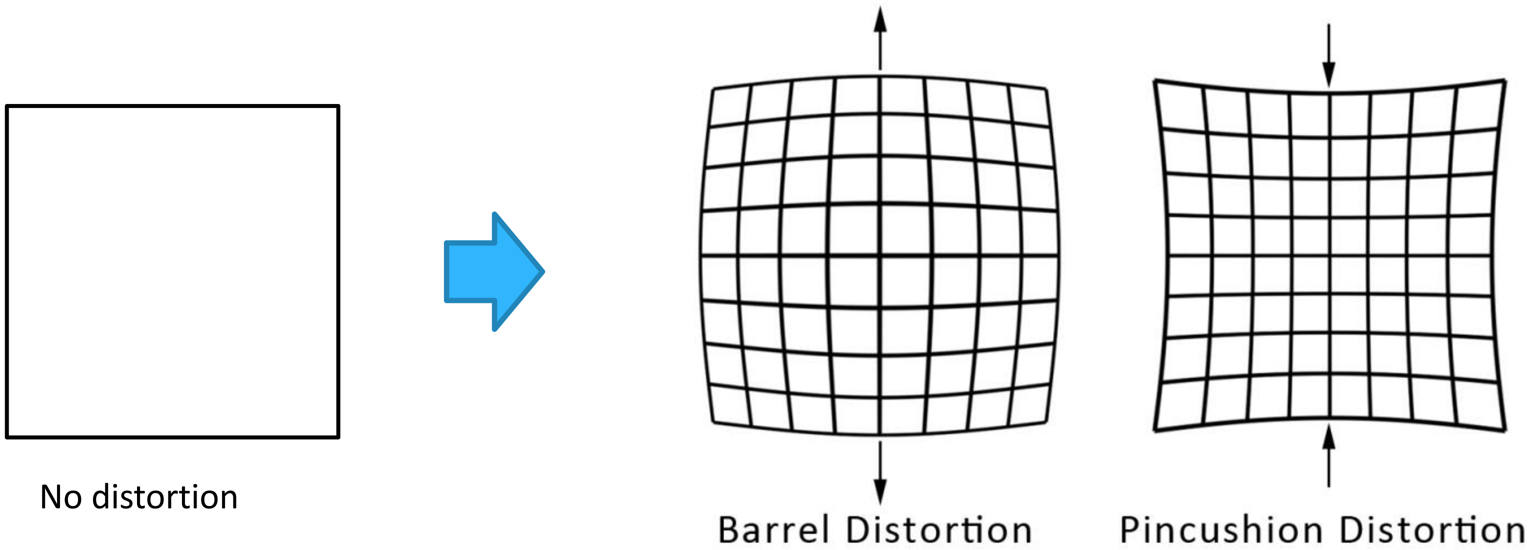


Image distortion

- Model I

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \underbrace{(1 + D_1 r^2 + D_2 r^4 + D_5 r^6)\begin{pmatrix} u \\ v \end{pmatrix}}_{\text{Radial distortion}} + \underbrace{\begin{pmatrix} 2D_3 uv + D_4(r^2 + 2u^2) \\ D_3(r^2 + 2v^2) + 2D_4 uv \end{pmatrix}}_{\text{Tangential distortion}}$$

Distortion coefficients: $D = [D_1, D_2, D_3, D_4, D_5]$

OpenCV, Matlab, Caltech camera calibration toolbox

Heikkila, Janne, and Olli Silven. "A four-step camera calibration procedure with implicit image correction." *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997.

Image distortion

- Model II – PTAM distortion model

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \frac{1}{r_u \omega} \arctan\left(2 r_u \tan\left(\frac{\omega}{2}\right)\right) \begin{bmatrix} f_x \frac{x}{z} \\ f_y \frac{y}{z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}$$

$$r_u \;:=\; \sqrt{\left(\frac{x}{z}\right)^2 + \left(\frac{y}{z}\right)^2}$$

A close form of inverse (distortion removal)

$$\begin{bmatrix} \tilde{u}_d \\ \tilde{v}_d \end{bmatrix} = \begin{bmatrix} (u_d - c_x) f_x^{-1} \\ (v_d - c_y) f_y^{-1} \end{bmatrix} \Rightarrow \begin{bmatrix} \tilde{u}_u \\ \tilde{v}_u \end{bmatrix} = \frac{\tan(r_d \omega)}{2 r_d \tan \frac{\omega}{2}} \begin{bmatrix} \tilde{u}_d \\ \tilde{v}_d \end{bmatrix}$$

$$r_d \;:=\; \sqrt{\tilde{u}_d^2 + \tilde{v}_d^2}$$

Image distortion

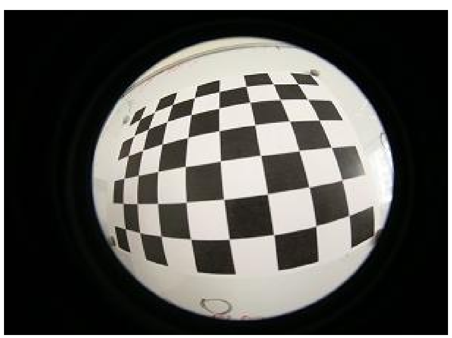
- Model III

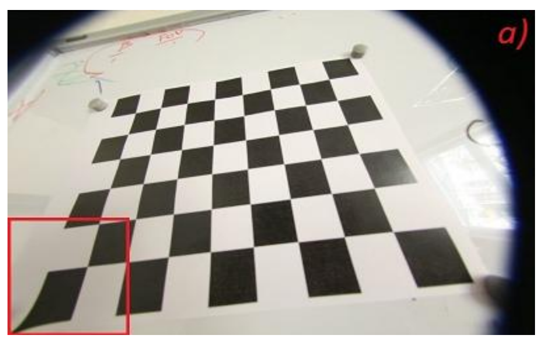
$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8)$$

$$r = \sqrt{u^2 + v^2} \qquad \theta = atan(r)$$

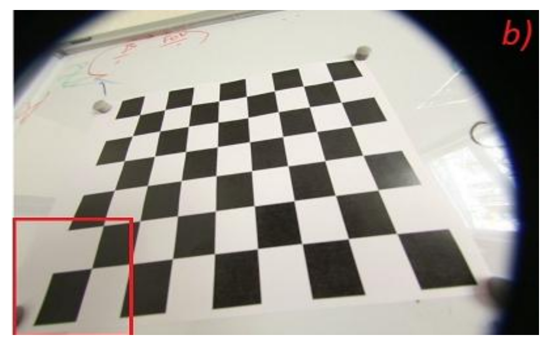
$$u' = (\theta_d / r)u$$

$$v' = (\theta_d / r)v$$





Model I

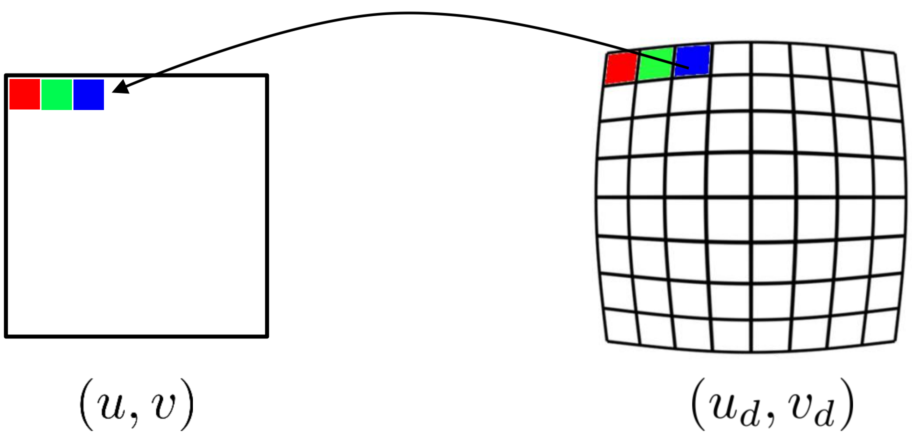


Model III

Kannala, Juho, and Sami S. Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses." *IEEE transactions on pattern analysis and machine intelligence* 28.8 (2006): 1335-1340.

Remove the distortion

- Rectify the distorted image:
  - For each pixel in the destination image (without distortion), find its corresponding pixel in the distorted image.
  - Fill the color of the corresponding pixel in the distorted image into the current pixel.
  - Repeat above steps until all pixels are filled.



Remove the distortion

- Compute the original coordinate from the distorted coordinate :

$$(u, v) \leftarrow (u_d, v_d)$$

  – Directly solve (u, v) from (u_d, v_d) is very difficult!

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \underbrace{(1 + D_1 r^2 + D_2 r^4 + D_5 r^6)}_{\tau} \begin{pmatrix} u \\ v \end{pmatrix} + \underbrace{\begin{pmatrix} 2D_3 uv + D_4(r^2 + 2u^2) \\ D_3(r^2 + 2v^2) + 2D_4 uv \end{pmatrix}}_{d\mathbf{x}}$$

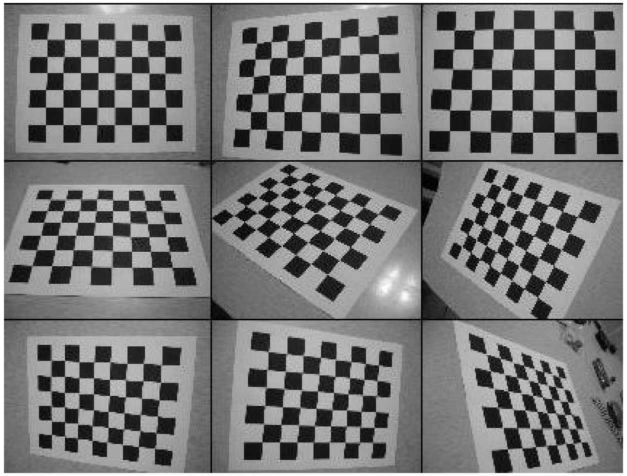
  – We can solve it iteratively :
    - Initially we let $(u_1, v_1) \leftarrow (u_d, v_d)$
    - Repeat until convergence :
      – Compute $\tau, d\mathbf{x}$ using (u_{i-1}, v_{i-1}).
      – We get (u_i, v_i) by

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \left(\begin{pmatrix} u_d \\ v_d \end{pmatrix} - d\mathbf{x} \right) / \tau$$

Usually, convergence can be achieved in 3~5 iterations

Camera calibration

- Calibration using checkerboard pattern
  - Use several snapshots of a checkerboard pattern to compute the intrinsic parameters.



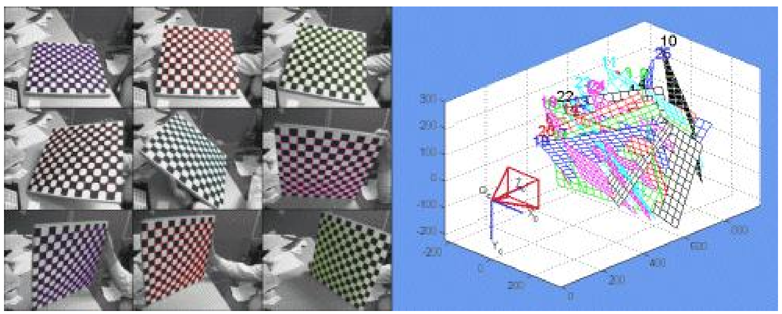
Zhang, Zhengyou. "A flexible new technique for camera calibration." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.11 (2000): 1330-1334.

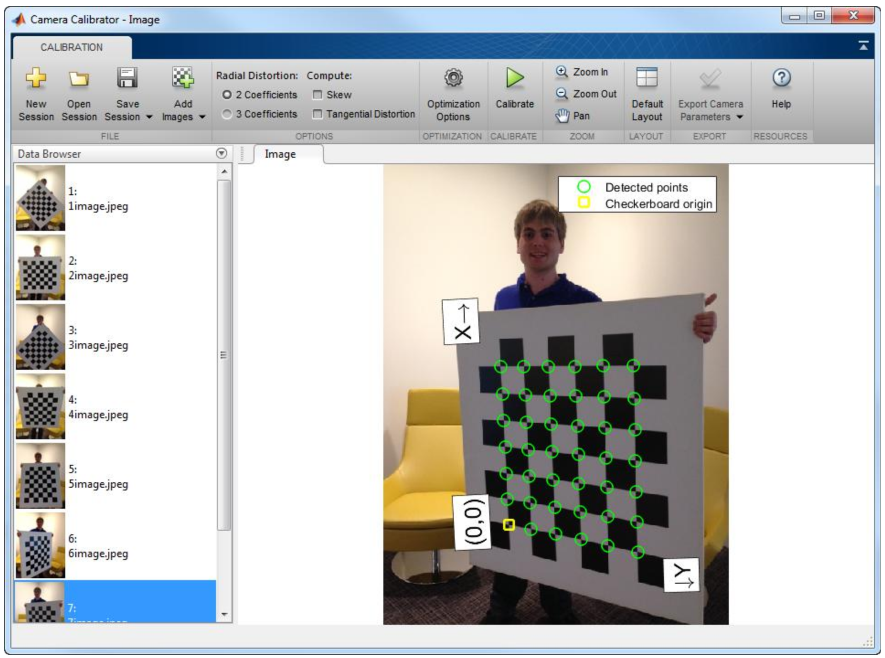
$$\mathbf{K} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{D} = [D_1, D_2, D_3, D_4, D_5]$$

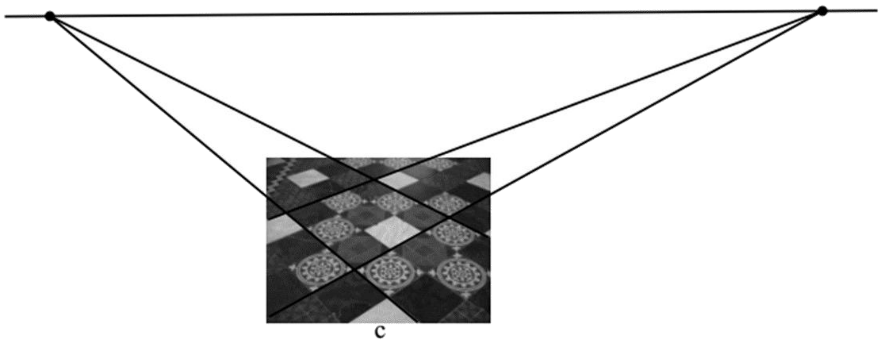
Calibration toolbox

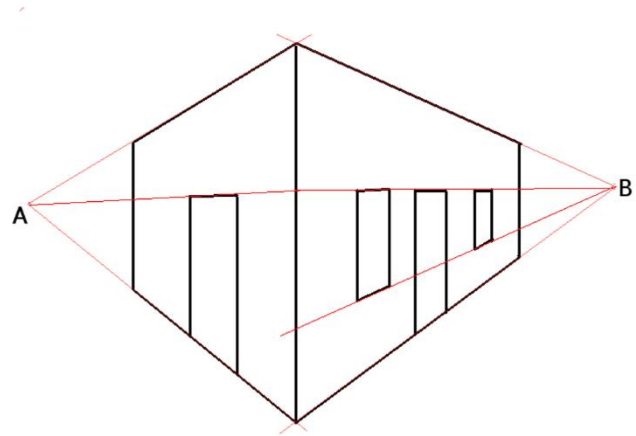
- Caltech camera calibration tool box
- Matlab computer vision system tool box





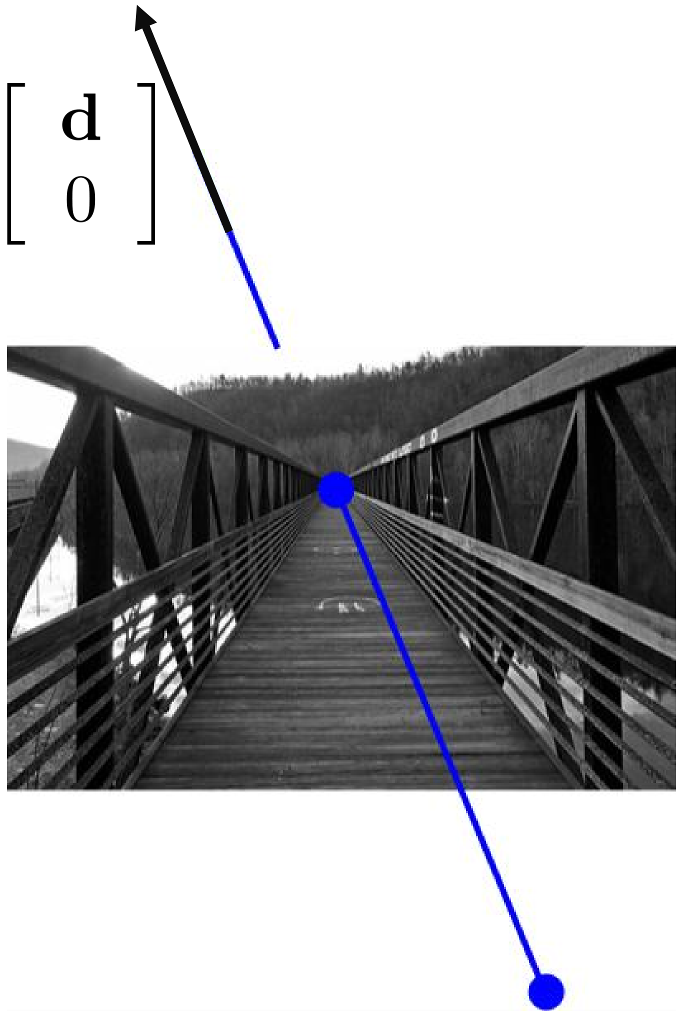
Vanishing points/lines





Vanishing points

$$\begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix}$$



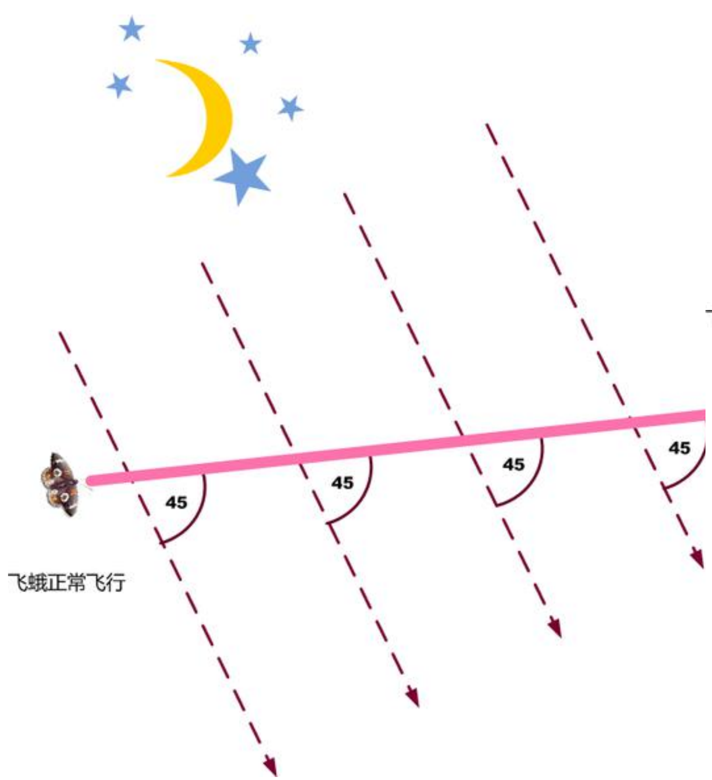
$$\mathbf{v} \sim [\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix}$$

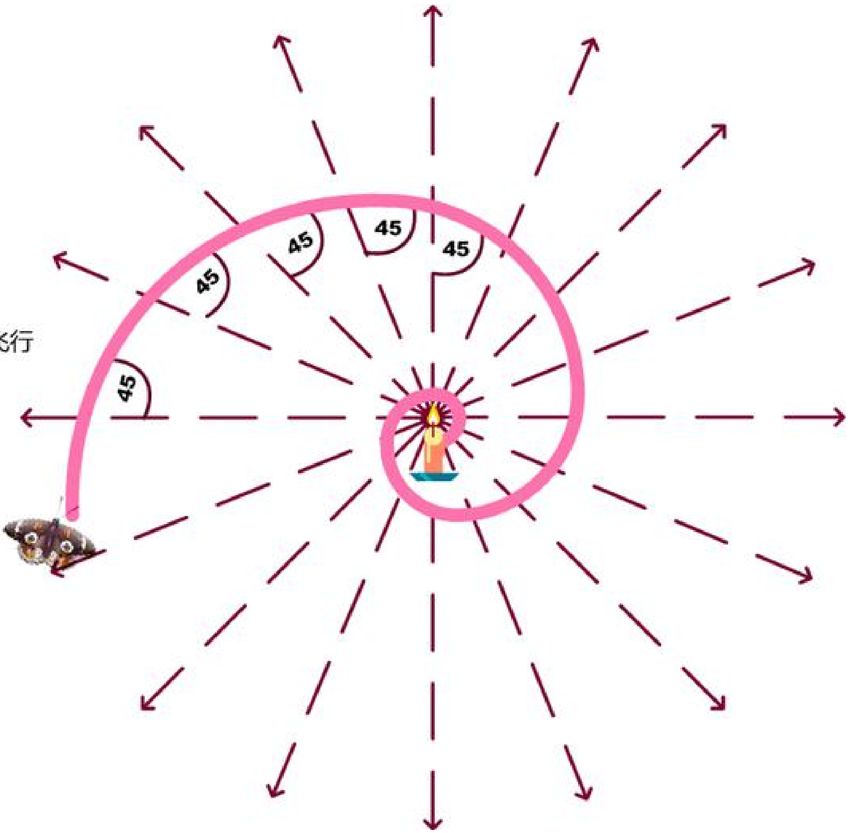
$$\boxed{\mathbf{v} \sim \mathbf{R}\mathbf{d}}$$

Vanishing points/lines

Vanishing points/lines

Vanishing points/lines





Vanishing points/lines

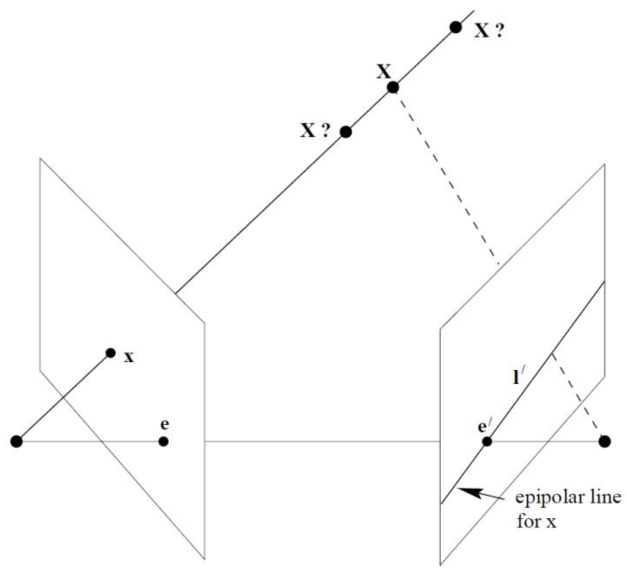
Two view geometry

Two view geometry

- Two view geometry tries to answer the following questions.
  - Given a image point in one view, where is its corresponding point in the other view?
    - **Epipolar constraint**
  - What is the relative pose between two views given a set of correspondences?
    - **Fundamental/Essential matrix estimation**
  - What is the 3D geometry of the scene?
    - **Triangulation**

Two view geometry

- Epiploar constraint:
  - Given a image point in the first view, how can we search its correspondence in the next view?



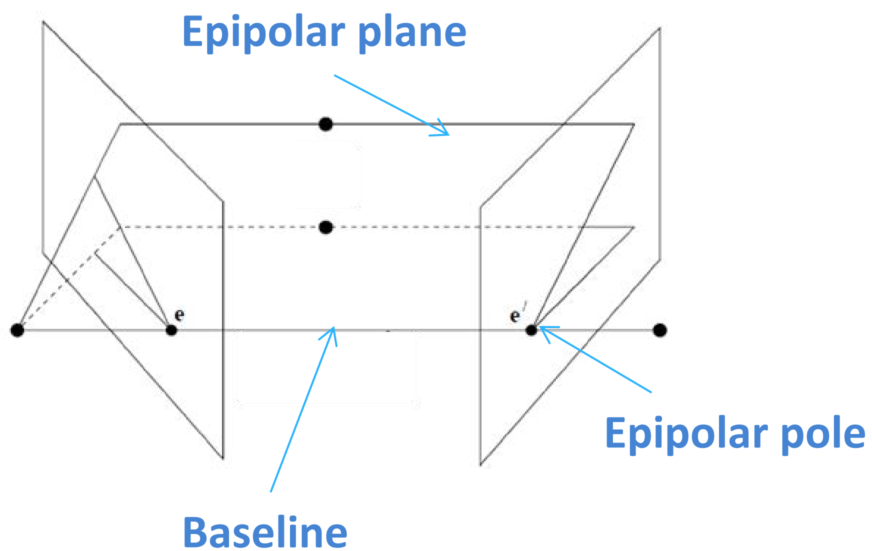
- Its correspondence lies in a line, which is named as '***epipolar line***' of $\mathbf{x}$.
- The geometry determining the epiplolar line is '***epipolar geometry***'
- The constraint that the correspondence of $\mathbf{x}$ should lie in the epipolar line is the ***epipolar constraint.***

Epiploar constraint

- The epipolar constraint is described mathematically as

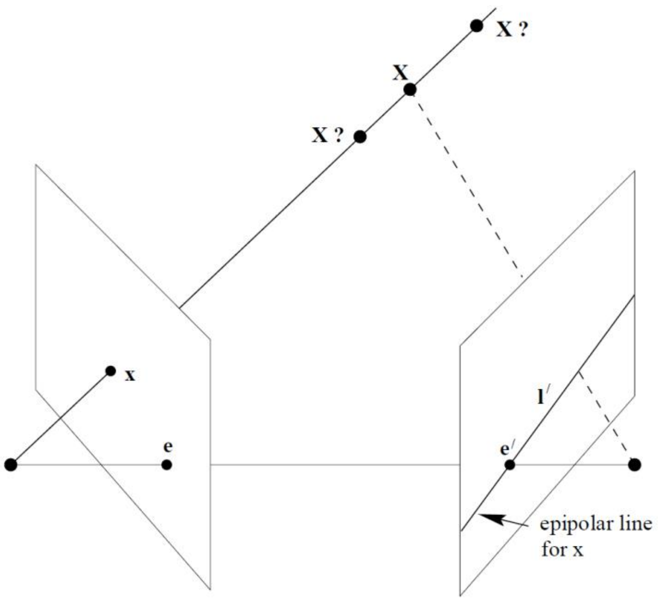
$$\mathbf{x'}^T F \mathbf{x} = 0$$

  - Here F is the **fundamental matrix**
  - $\mathbf{l} = F\mathbf{x}$ is the **epipolar line** of $\mathbf{x}$



Epiploar constraint

- Epipolar constraint - Derivation



$$\mathbf{x} = [I\ \mathbf{0}]\mathbf{X} \qquad \mathbf{x}' = [R\ t]\mathbf{X}$$

$$\mathbf{l}' = \mathbf{e}' \times \mathbf{x}'$$

$$\mathbf{l}' = t \times \mathbf{x}' \qquad (\mathbf{e}' = [R\ t](0,0,0,1)^T)$$

$$\begin{aligned}\mathbf{l}' &= t \times (R\mathbf{x} + t)\lambda \qquad (\mathbf{X} = \lambda(\mathbf{x}, 1)^T) \\ &= t \times R\mathbf{x} \\ &= [t]_\times R\mathbf{x}\end{aligned}$$

$$\boxed{\mathbf{x}'^T E \mathbf{x} = 0} \qquad (\mathbf{x}'^T \mathbf{l} = 0)$$

$E = [t]_\times R$ is the **essential matrix**

Epiploar constraint

- Essential matrix and fundamental matrix

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

$$\Downarrow$$

$$\mathbf{m}'^T K'^{-T} E K^{-1} \mathbf{m} = 0 \quad (\mathbf{m}' = K'\mathbf{x}', \mathbf{m} = K\mathbf{x})$$

$$\Downarrow$$

$$\mathbf{m}'^T F \mathbf{m} = 0$$

$$\Downarrow$$

$$\boxed{F = K'^{-T} E K^{-1}}$$

Fundamental matrix

Fundamental matrix estimation

- Given a set of point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, solve

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

  - Fundamental matrix is a rank-2 matrix and has seven degree of freedom
  - At least 7 points are required to solve the fundamental matrix, where each point provides one equation.
  - **Eight point algorithm**
  - **Seven point algorithm**

Fundamental matrix estimation

- **Eight point algorithm**
  - For each correspondence $\mathbf{x} \leftrightarrow \mathbf{x}'$, we have the equation

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

which can be written as

$$(x'x, x'y, x', y'x, y'y, y', x, y, 1) \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

where $\mathbf{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^\top$ holds the entities of $\boldsymbol{F}$

Two view geometry

- The entries of fundamental matrix can be solved by stacking all equations together.

$$\mathbf{Af} = 0$$

- Since $\boldsymbol{F}$ is determined up to scale only, at least eight points are required to solve $\boldsymbol{F}$.
- The solution can also be obtained by SVD decomposition.

<u>Least-squares solution</u>

(i) Form equations $\mathtt{A}\mathbf{f} = \mathbf{0}$.

(ii) Take SVD : $\mathtt{A} = \mathtt{UDV}^\top$.

(iii) Solution is last column of $\mathtt{V}$ (corresp : smallest singular value)

(iv) Minimizes $||\mathtt{A}\mathbf{f}||$ subject to $||\mathbf{f}|| = 1$.

Fundamental matrix estimation

- Singularity correction
  - The solution by 8 point algorithm does not satisfy the singularity condition.
    - F is rank-2 matrix or mathematically, $det(F) = 0$
  - SVD approximation
    - Decompose F by SVD $F = U\Sigma V^T$
    - Here $\Sigma = diag(r, s, t)$.
    - The SVD approximation of F is

$$F' = U diag(r, s, 0) V^T$$

F' is the 'closest' singular matrix to F in Frobenius norm!

Fundamental matrix estimation

- **Seven point algorithm**
  - If we impose the singularity condition on the unknowns, we get another equation.
  - So we can solve the fundamental matrix by 7 points
- Steps:
  - 1. Get the null space solution of $\mathbf{A}\mathbf{f} = 0$

$$\mathbf{f} = \lambda \mathbf{f}_0 + \mu \mathbf{f}_1$$

  - 2. Rewrite it into the matrix form

$$\mathrm{F} = \lambda \mathrm{F}_0 + \mu \mathrm{F}_1$$

  - 3. Condition $\det(F)$ = 0 gives a cubic equation of λ, μ
  - 4. Solve λ, μ and get F

Essential matrix estimation

- Compute essential matrix
  - Once the camera has been calibrated.
  - Only five points are required to solve essential matrix since there is only five degree of freedom in essential matrix

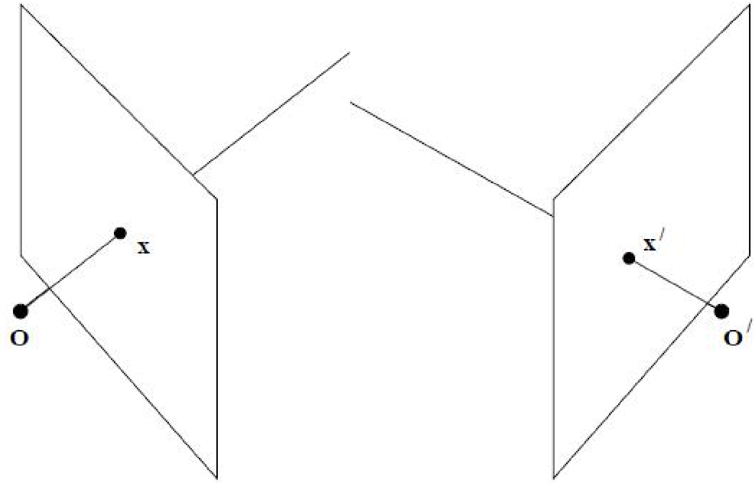
$$E = [t]_{\times} R$$

  - Nister’s five point algorithm

Nistér, David. "An efficient solution to the five-point relative pose problem."*Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26.6 (2004): 756-770.

Triangulation

- Knowing P and P'
- Knowing $\boldsymbol{x}$ and $\boldsymbol{x}'$
- Compute $\boldsymbol{X}$



$$\mathbf{x} = P\mathbf{X}$$

$$\mathbf{x}' = P'\mathbf{X}$$

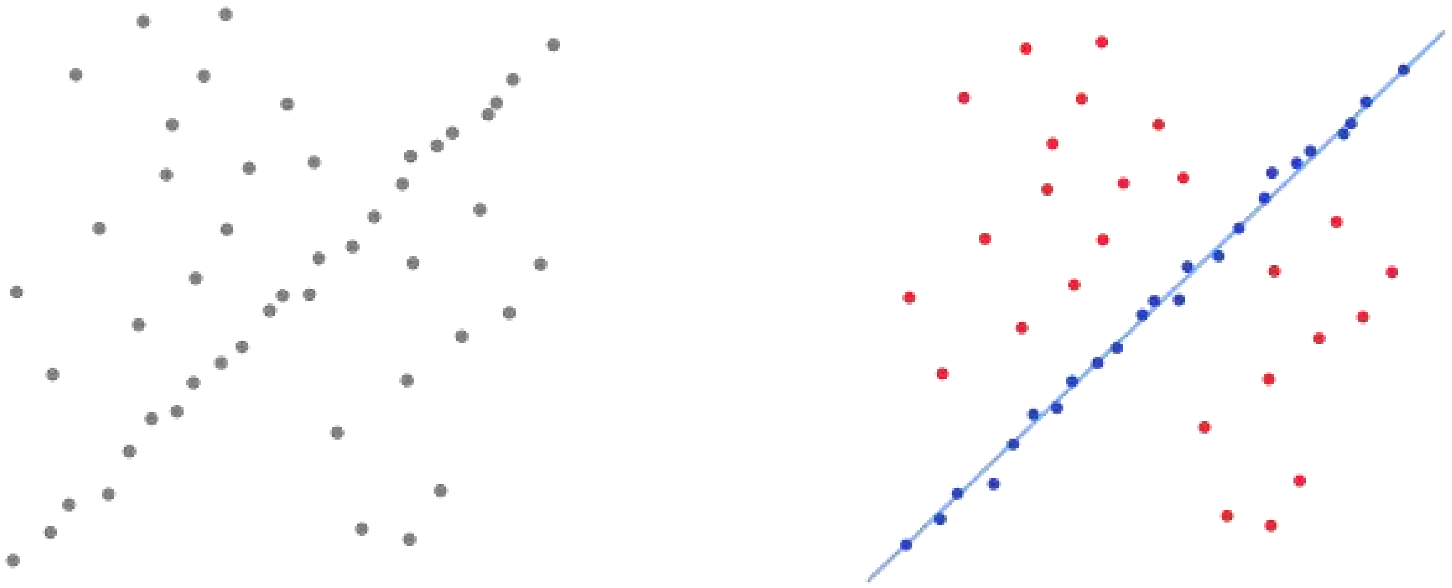
Refinement

- Minimizing the re-projection errors $\|x - f(P, X)\|^2 + \|x' - f(P', X)\|^2$

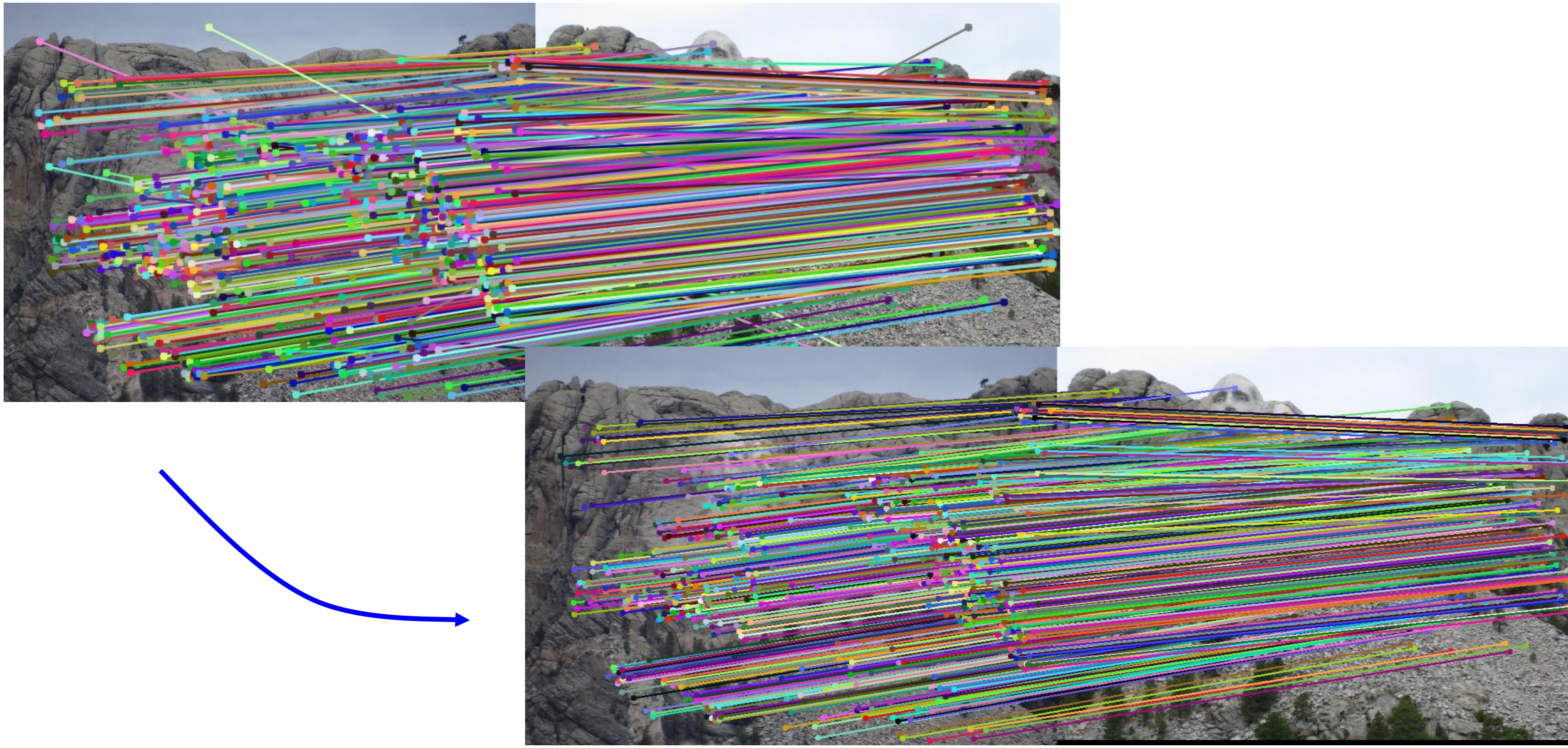
Here $f(P, X) = \begin{pmatrix} \frac{p_1 x + p_2 y + p_3 z + p_4}{p_9 x + p_{10} y + p_{11} z + p_{12}} \\ \frac{p_5 x + p_6 y + p_7 z + p_8}{p_9 x + p_{10} y + p_{11} z + p_{12}} \end{pmatrix}$. It is a nonlinear least square problem and can be solved by Levenberg-Marquardt algorithm efficiently.

RANSAC algorithm

- **RAN**dom **S**ample **A**nd **C**onsensus
  - Robust estimation under the presence of a significant number of outliers



RANSAC algorithm



RANSAC algorithm

- Randomly select a small subset of correspondences and solve the Fundamental/Essential matrix
- Evaluate the error residuals for the rest of the correspondences. The **Consensus set** is the set of correspondences within the error threshold
- Repeat above steps and finally select solution that yields the largest consensus set.

A quick way to learn all about this

- Write a simple program to reconstruct 3D points from two snapshots. For example, use your phone.
- The pipeline
  - 1. calibrate the camera intrinsic parameters
  - 2. take two pictures by your phone
  - 3. Match feature points (SIFT, SURF)
  - 4. Use RANSAC algorithm to estimate the fundamental matrix and remove the outlier
  - 5. use Nister's code to estimate the essential matrix from the inlier corresponding points
  - 6. extract the R and t from the essential matrix
  - 7. triangulate the 3D points


dws4.me

Outline

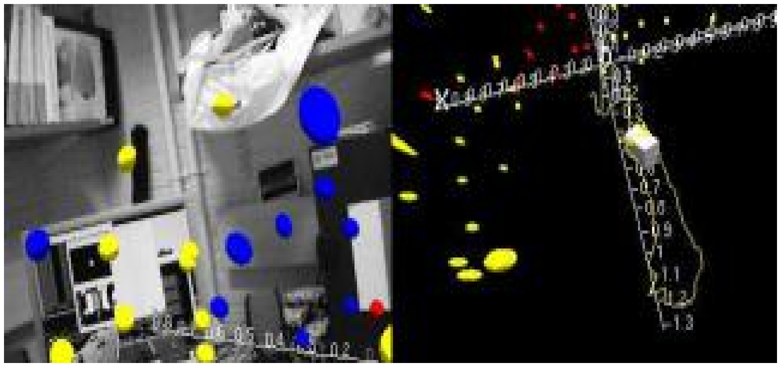
- Basic Theory
  - Projective geometry
  - Pinhole camera model
  - Camera calibration
  - Two camera geometry

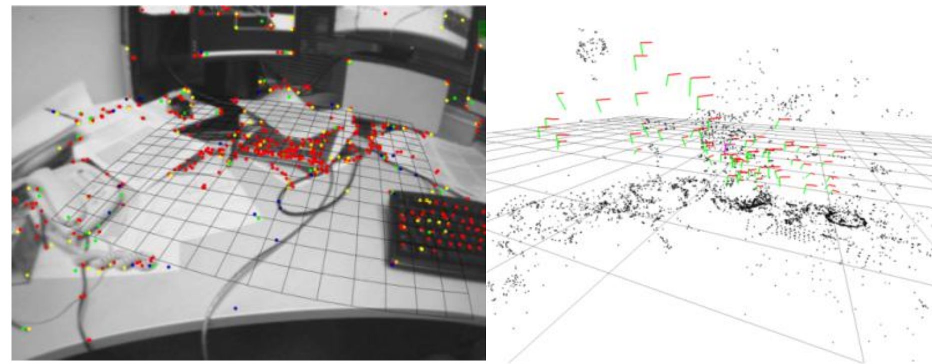
- Design a typical Visual SLAM system
- Two Visual SLAM systems:
  - Extended Kalman Filter approach:
    - StructSLAM
  - Visual SLAM for a group of robots:
    - CoLSAM

Monocular Visual SLAM

Filter-based approach

Key frame-baed approach (SFM)





2003, MonoSLAM

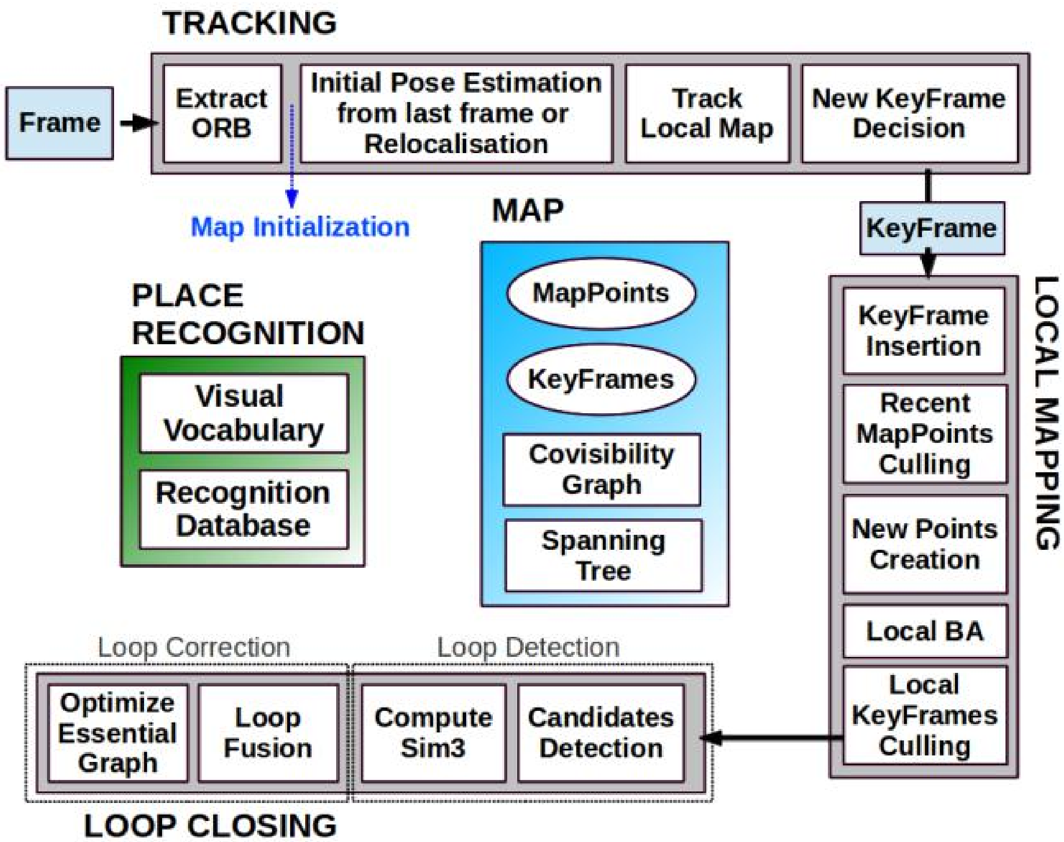
2007, PTAM

[1] Davison, Andrew J., et al. "**MonoSLAM**: Real-time single camera SLAM." Pattern Analysis and Machine Intelligence, IEEE Transactions on 29.6 (2007): 1052-1067.

[2] Klein, Georg, and David Murray. "**Parallel tracking and mapping for small AR workspaces**." Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. IEEE, 2007

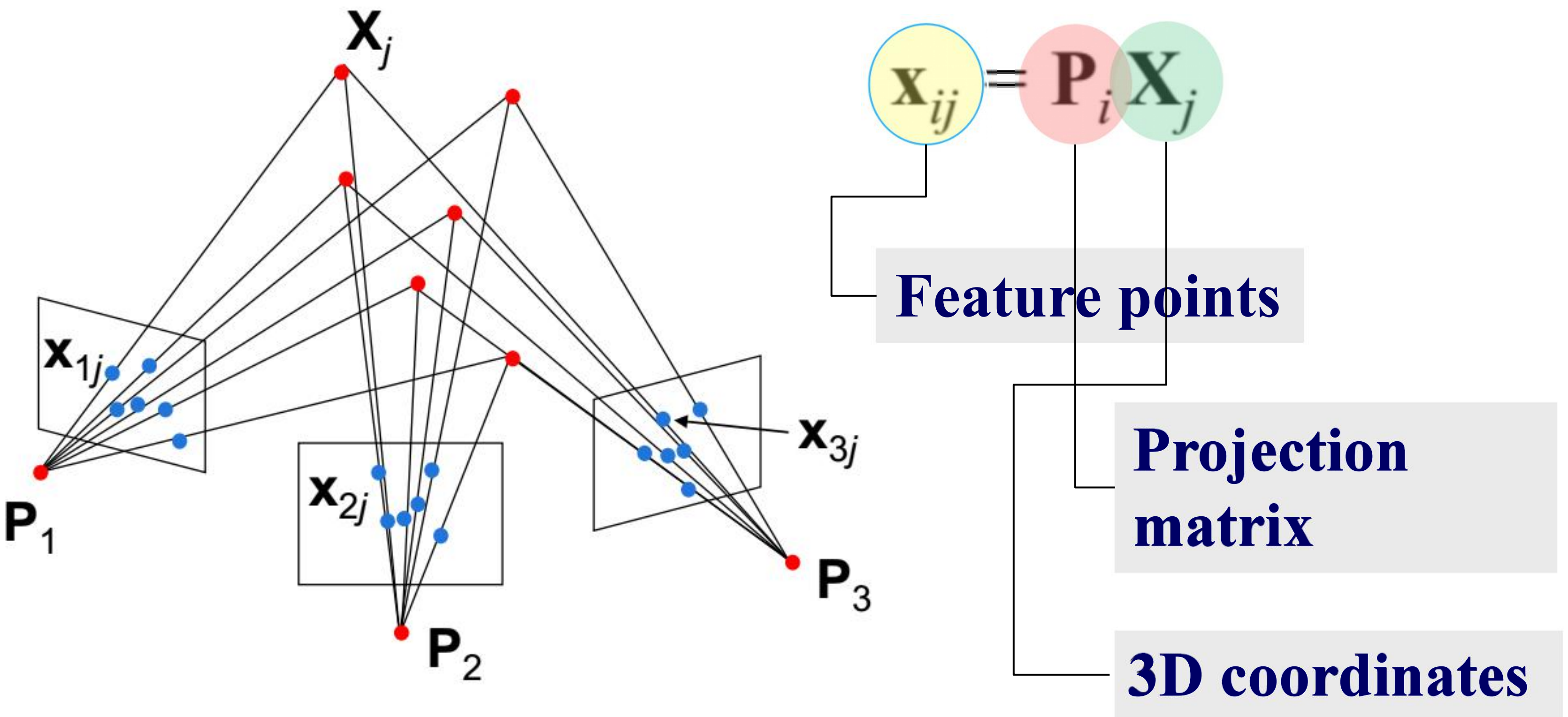
ORB-SLAM

An extension from PTAM
- Robust initialization
- Loop closing
- Visibility graph



Structure-from-motion

- 3D Model from Image Sequences



Structure-from-motion

Factorization/Batch | **Incremental SFM**

1990 — **2000**

Tomasi C, Kanade T

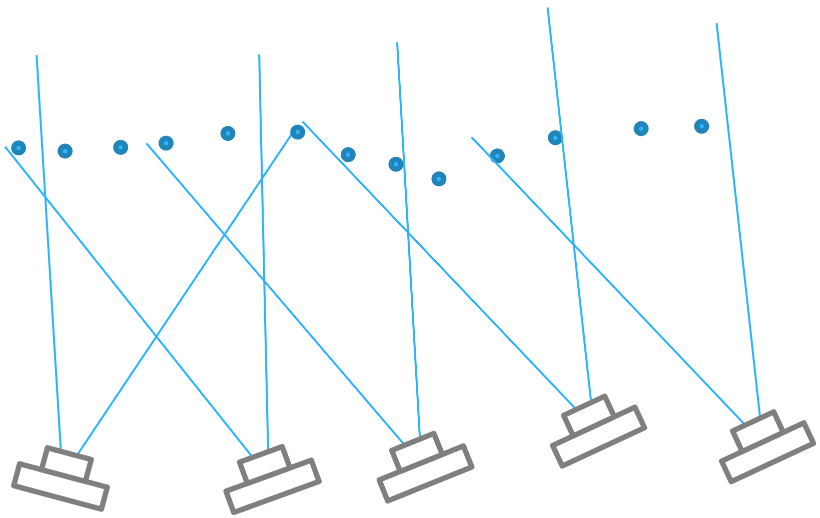
Shape and motion from image streams under orthography: a factorization method,IJCV,1992

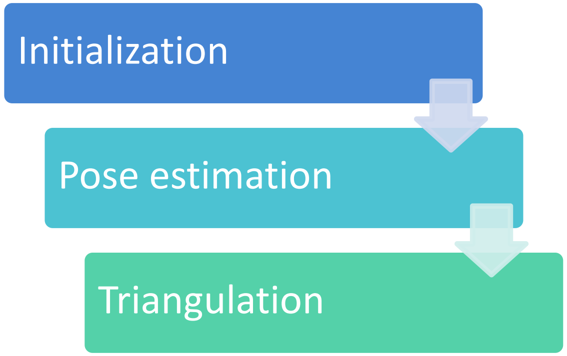
- **Photo Tourism (Bundler)**
- **Rome in a Day**

Optimization approach – Bundle Adjustment

A quick overview of a SFM system

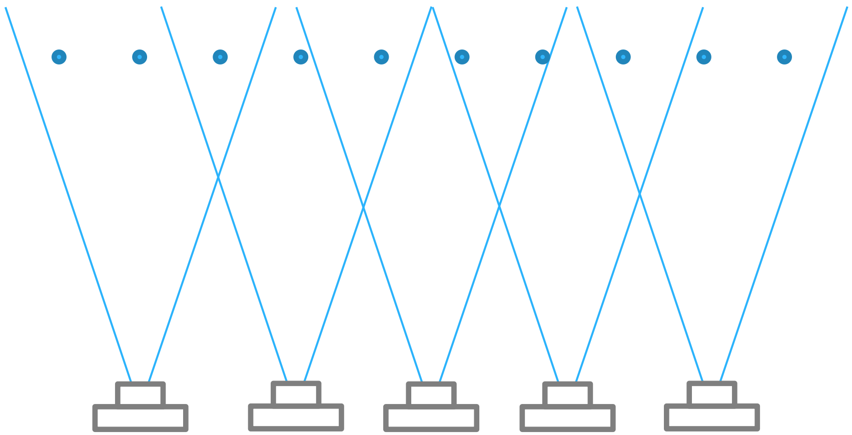
- A typical pipeline of incremental structure-from-motion (one camera case)

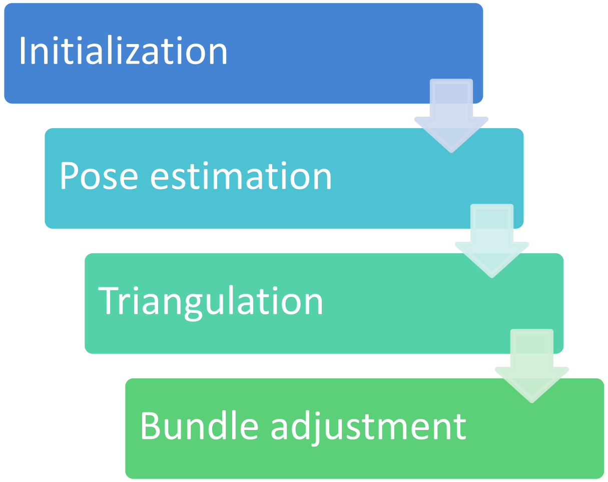




A quick overview of a SFM system

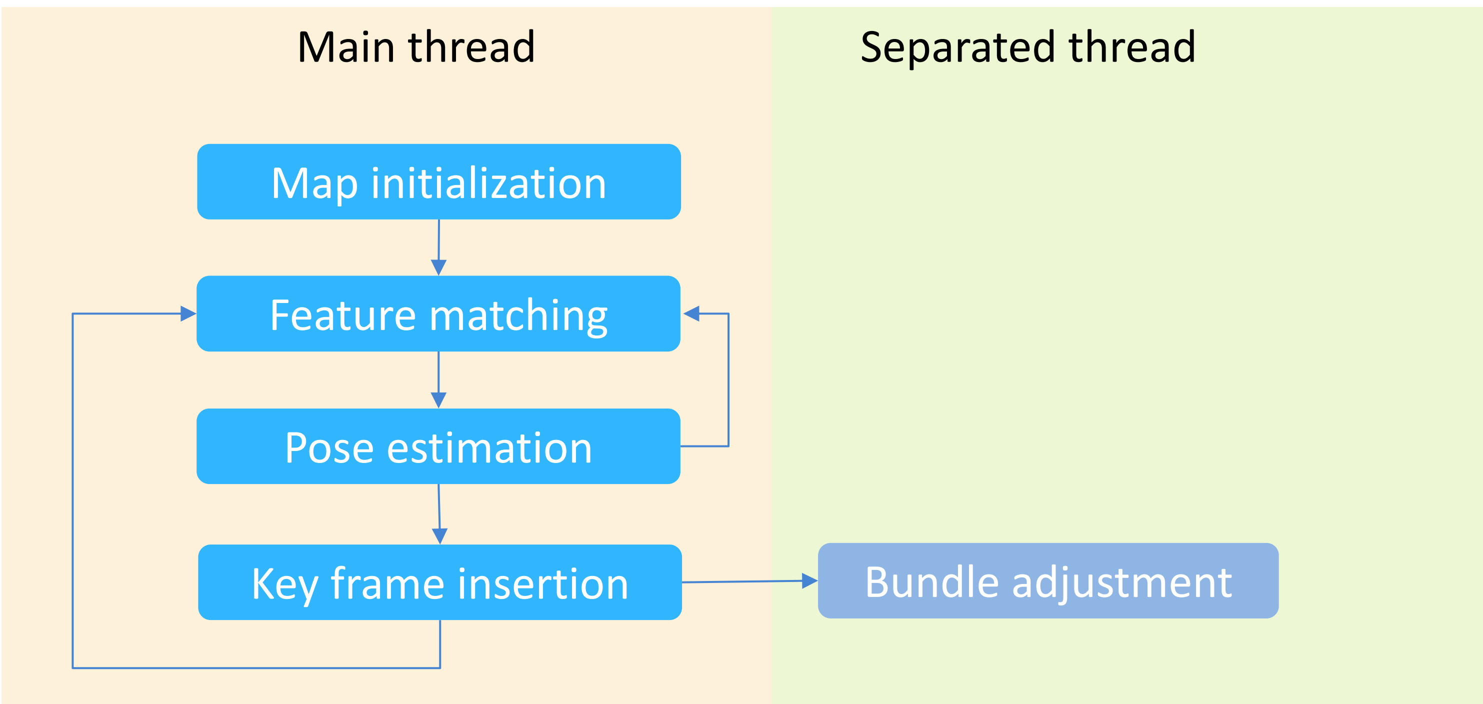
- A typical pipeline of incremental structure-from-motion (one camera case)





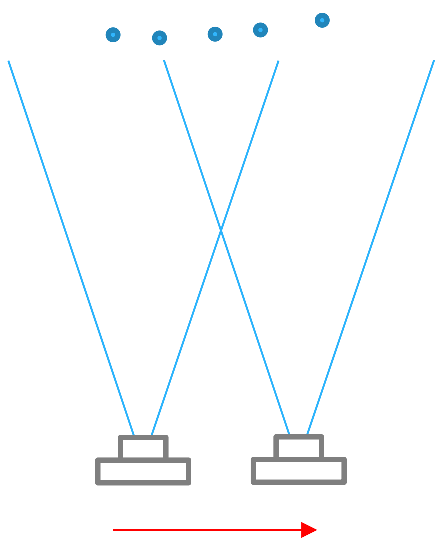
Single camera SLAM- overview

- Camera Tracking (Localization)
  - Feature matching
  - Pose estimation
- Mapping
  - Initialization
  - Key frame insertion
  - Bundle adjustment



Map initialization

- Map initialization
  - Use two images to get the **initial poses** and generate **seed map points**



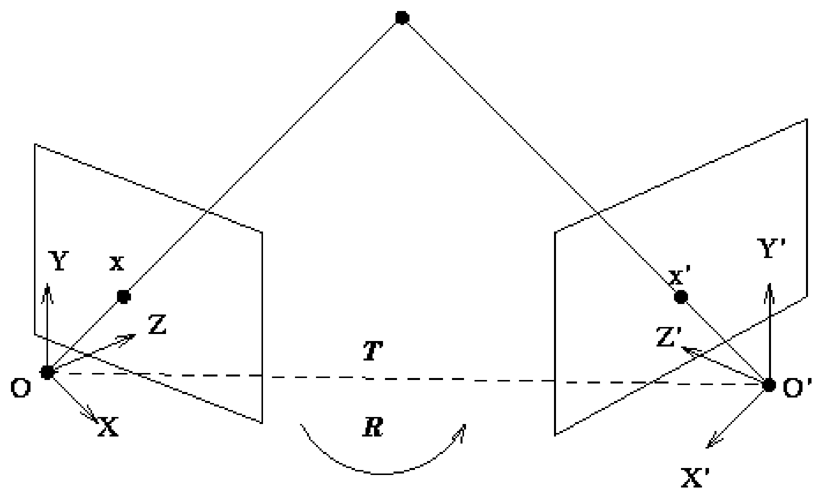
Step1:Estimate the essential matrix

Step2:Decompose the relative pose

Step3:Triangulation

Map initialization

- Estimate the essential matrix (five point algorithm)



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

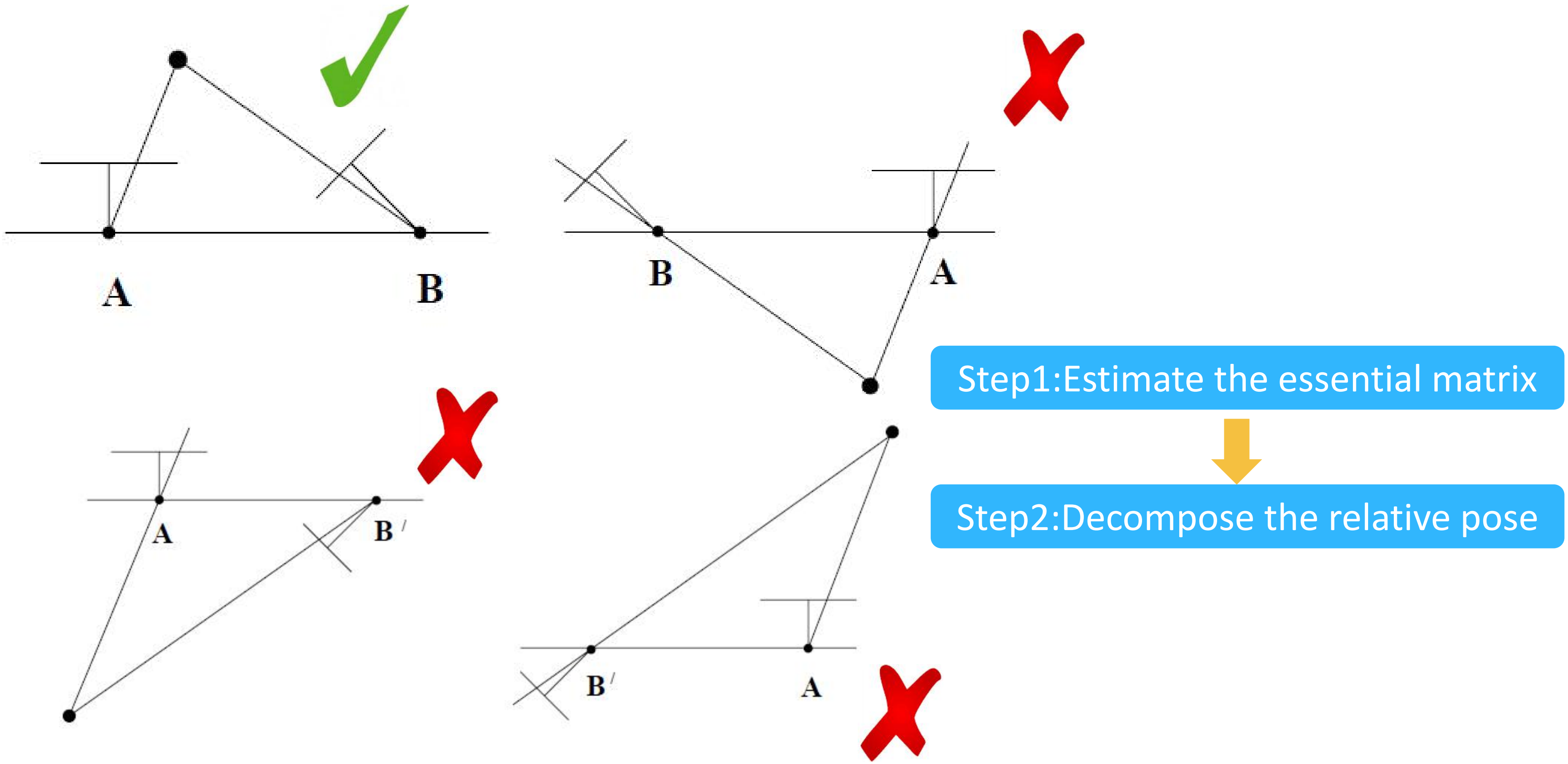
$$E = [t]_\times R$$

Step1:Estimate the essential matrix

Nistér, David. "An efficient solution to the five-point relative pose problem."*Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26.6 (2004): 756-770.

Map initialization

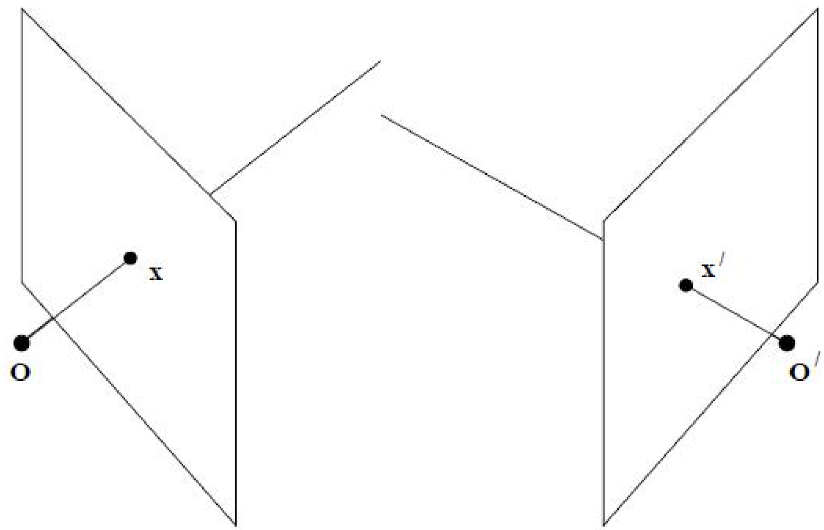
- Relative pose decomposition. As I explained previous there are four possible solutions:



Map initialization

- Triangulation - generate 3D points

$$\mathbf{x} = P\mathbf{X} \qquad \mathbf{x}' = P'\mathbf{X}$$



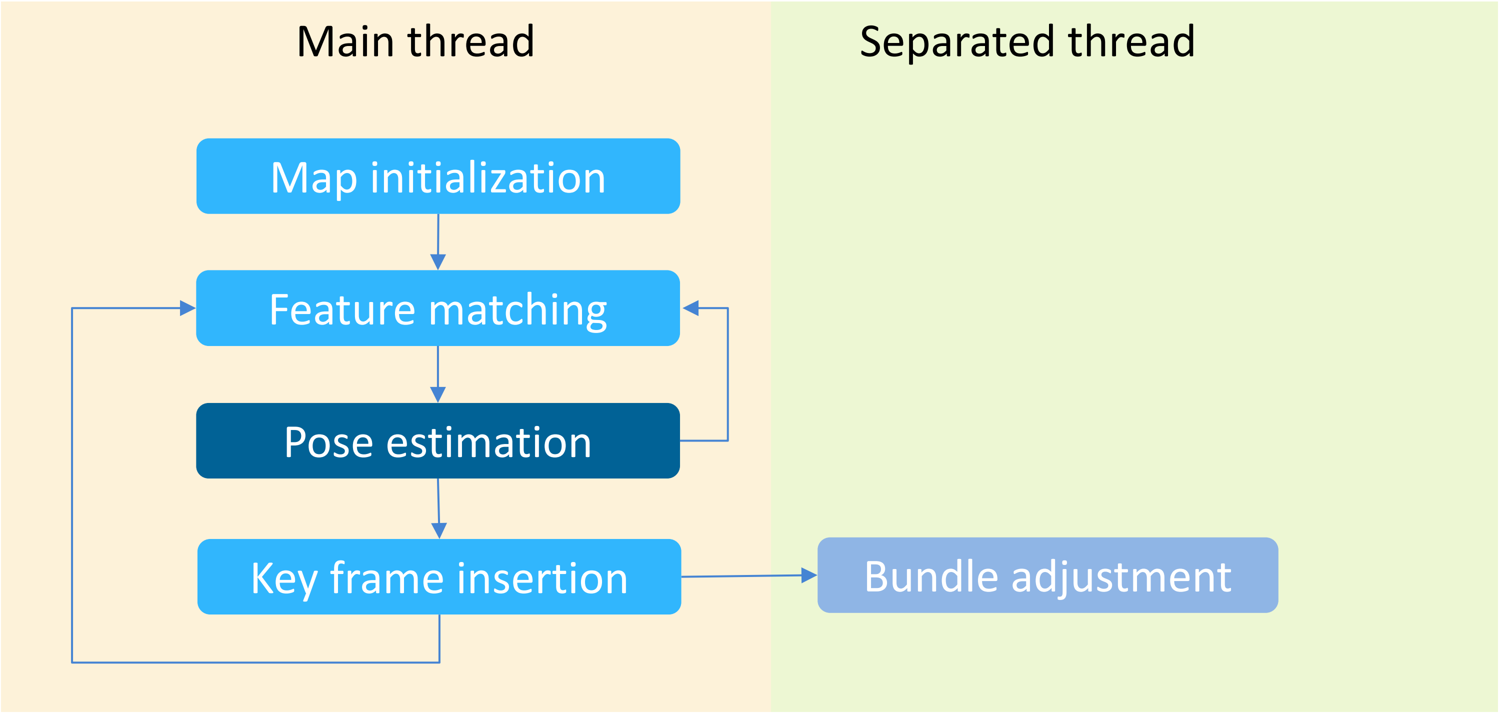
$\Rightarrow \quad \mathbf{X}$

Step1:Estimate the essential matrix

Step2:Decompose the relative pose

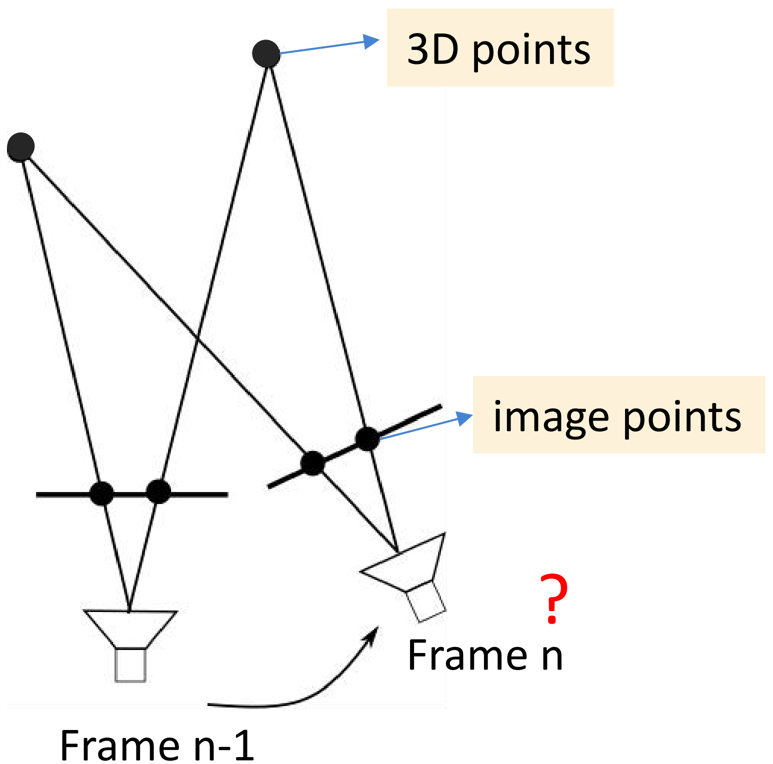
Step3:Triangulation

Pose estimation



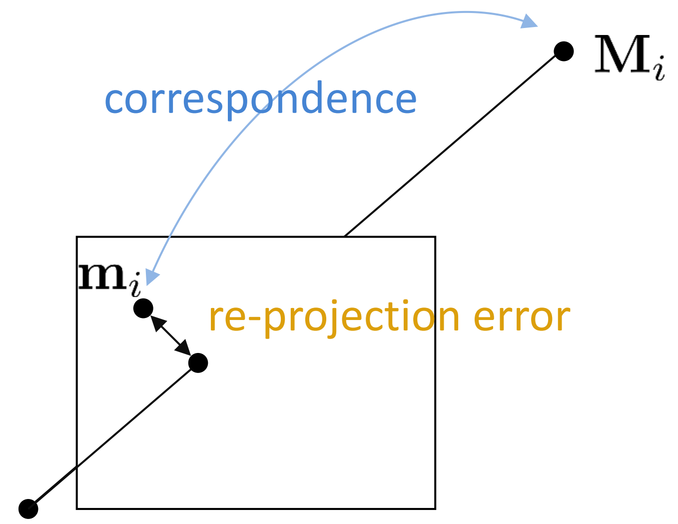
Pose estimation

- The problem:
  - Given 3D points and their corresponding images, how do we compute the camera pose ? (given that the camera is calibrated)



Pose estimation

- Denote 3D points by $\{\mathbf{M}_i\}$ and their corresponding images by $\{\mathbf{m}_i\}$.
- The re-projection error of a 3D point is defined as the distance between the image point and its projection.



Pose estimation

- Re-projection error:

$$r_i(\theta) = \mathbf{m}_i - Proj(\mathbf{M}_i, \theta)$$

Camera pose

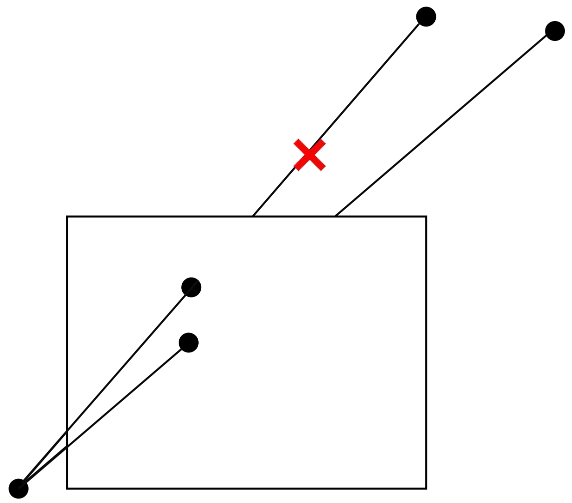
- We want find a pose that minimizes

$$\boxed{\theta^* = \arg\min_\theta \sum r_i^2(\theta)}$$

This is a standard non-linear least square problem, which can be solved by **Levenberg-Marquardt** algorithm.

Pose estimation

- How about if we get noisy correspondences?
  - Feature matching is not always correct!



$$\sum r_i^2(\theta)$$

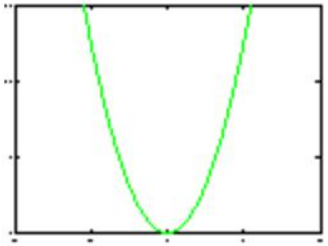
Pose estimation

- Another robust method : **M-estimator**

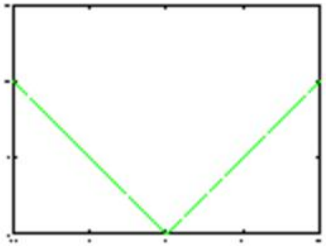
The **M-estimators** try to reduce the effect of outliers by replacing the squared residuals with another function.

$$\sum r_i^2(\theta) \Rightarrow \sum \rho(r_i(\theta))$$

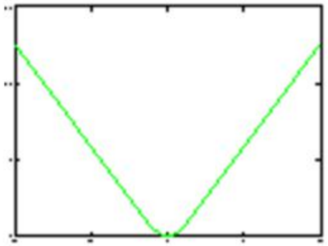
$\rho(\cdot)$



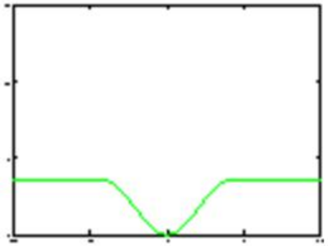
L2



L1



Huber



Tukey

Pose estimation

- **Least square V.S. M-estimator**

$$\sum r_i^2(\theta + \Delta\theta) \quad \leftrightarrow \quad \sum \rho(r_i(\theta + \Delta\theta))$$

$$\sum r_i \frac{\partial r_i}{\partial \Delta\theta} = 0 \qquad \sum \rho'(r_i) \frac{\partial r_i}{\partial \Delta\theta} = 0$$

$$\sum \underbrace{\boxed{\frac{\rho'(r_i)}{r_i}}}_{w(r_i)} r_i \frac{\partial r_i}{\partial \Delta\theta} = 0$$

This is a weighted least square problem!

M-estimator

- Reweighted least square algorithm:
  - Solve the weighted least square problem using initial weights (1s)
  - Evaluate the residual and update the weights
  - Repeat above steps for several times

$$\min \sum \rho(r_i(\theta))$$

Initialization: $\theta^{(0)} \leftarrow \min \sum r_i^2(\theta)$

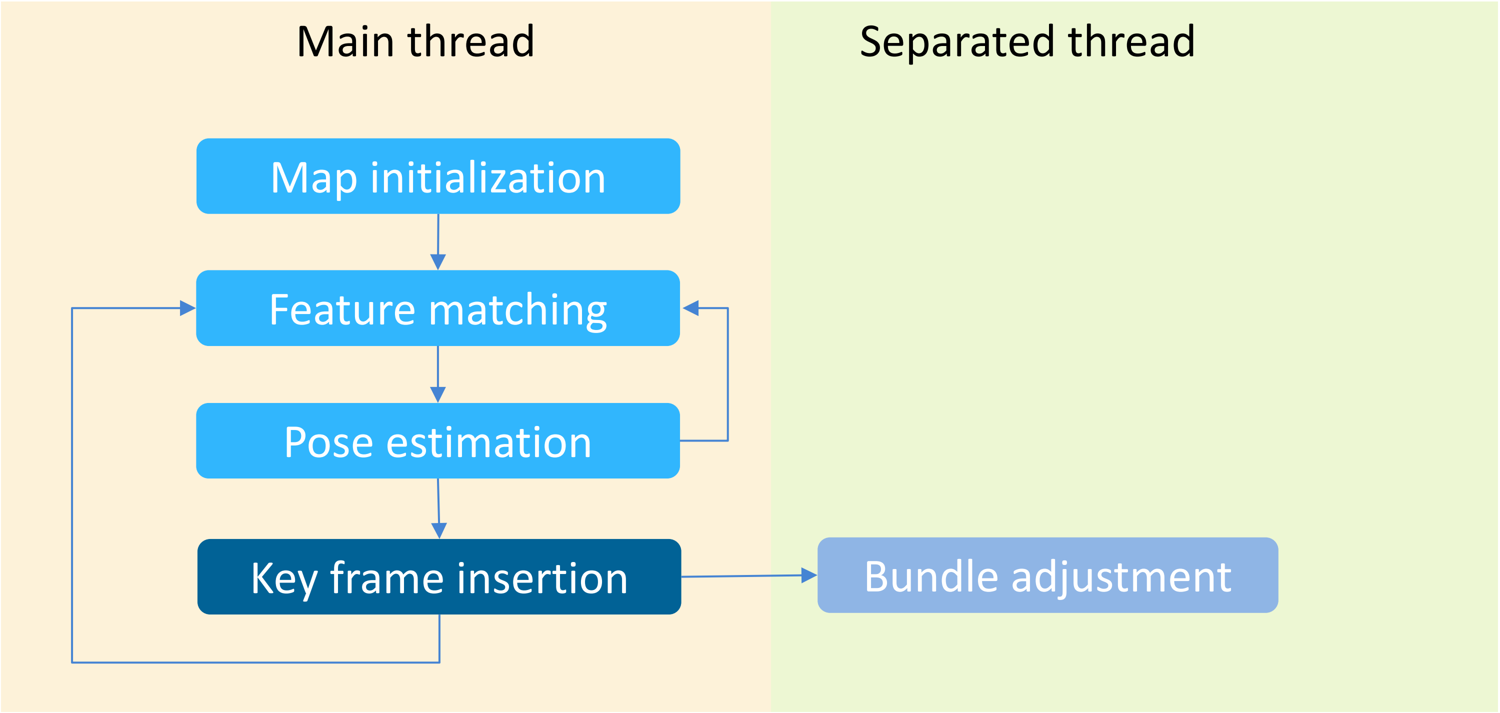
Compute new weights: $w^{(k)} \leftarrow \frac{\rho'(r_i^{(k-1)})}{r_i^{(k-1)}}$

Solve for new parameters $\theta^{k+1} \leftarrow \min \sum w^{(k+1)} r_i^2(\theta)$

M-estimator tutorial by Zhengyou Zhang

http://research.microsoft.com/en-us/um/people/zhang/INRIA/Publis/Tutorial-Estim/node24.html

Key frame selection



Key frame selection

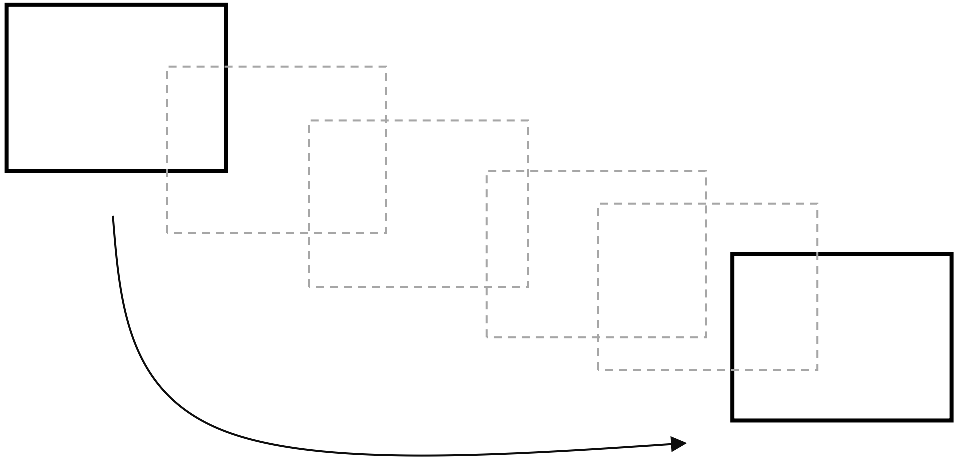
- What is key frame ?
  - An structure storing:
    - current **camera pose**
    - current **3D points** and their **image correspondences**

Question: Why not select all video frames as key frames?

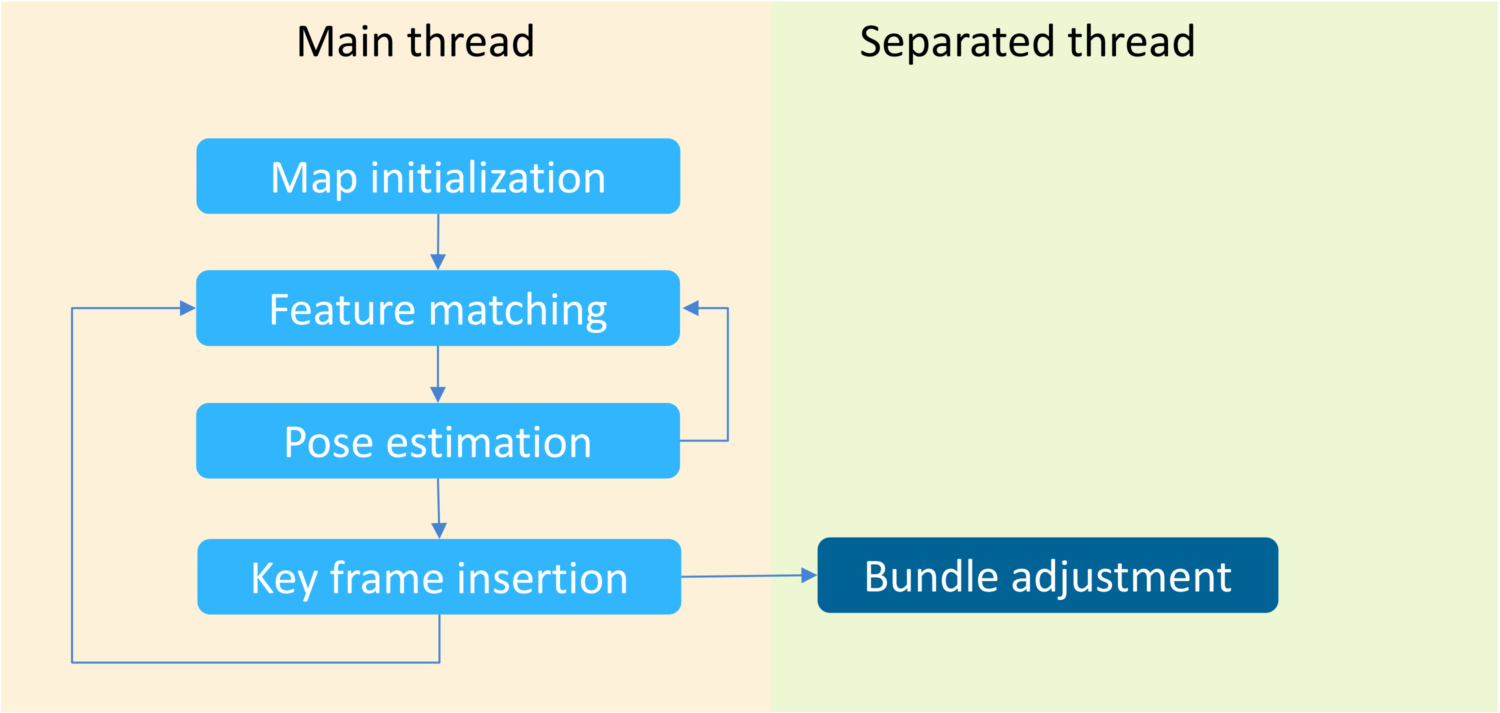
Because it is not efficient (computation time + memory request)
- two many points
- tw0 many camera poses

Key frame selection

- Some strategy to select key frame
  - A sufficient moving distance
  - Good quality of image
  - Maintain the number of features tracked



Bundle adjustment



Bundle adjustment

- What is bundle adjustment?
  - Bundle adjustment is to minimize re-projection errors in all views with respect to all 3D points and all camera poses

$$\min \sum_i \sum_j (\mathbf{m}_{ij} - Proj(\theta_i, \mathbf{M}_j))^2$$

  - This is still a non-linear least square problem

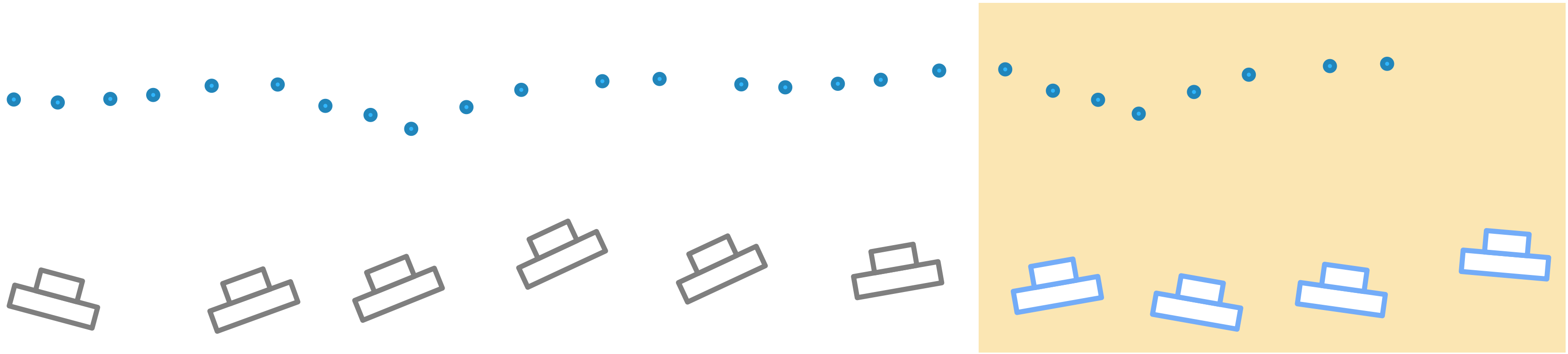
$$\min \sum_i \sum_j r_{ij}(\mathbf{x})^2$$

$\mathbf{x}$ is a vector containing all camera poses and 3D points

Software
sba: http://users.ics.forth.gr/~lourakis/sba/
mcba: http://grail.cs.washington.edu/projects/mcba/

Bundle adjustment

- Bundle adjustment with all parameters involved costs a lot of time.
  - A alternative solution is selecting only a subset of parameters to optimize. This approach is so called ***local bundle adjustment***.



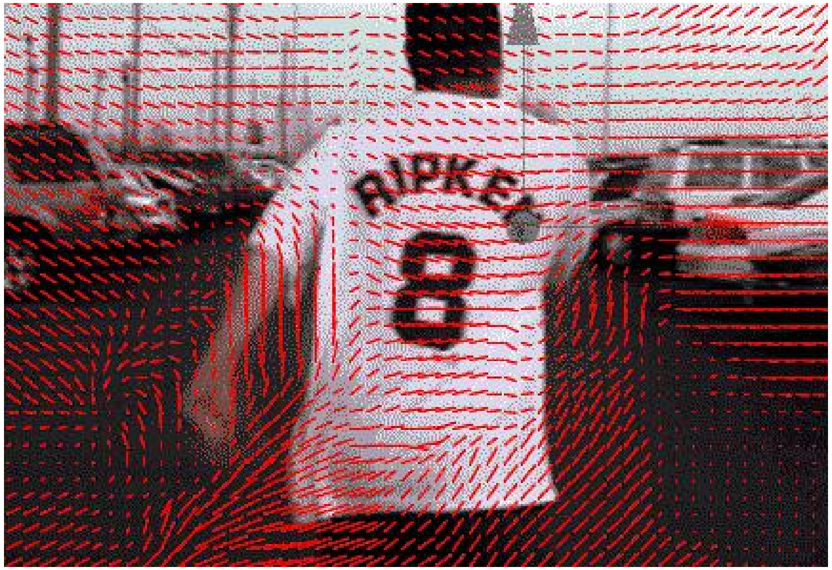
Keyframes	2-49	50-99	100-149
Local Bundle Adjustment	170ms	270ms	440ms
Global Bundle Adjustment	380ms	1.7s	6.9s

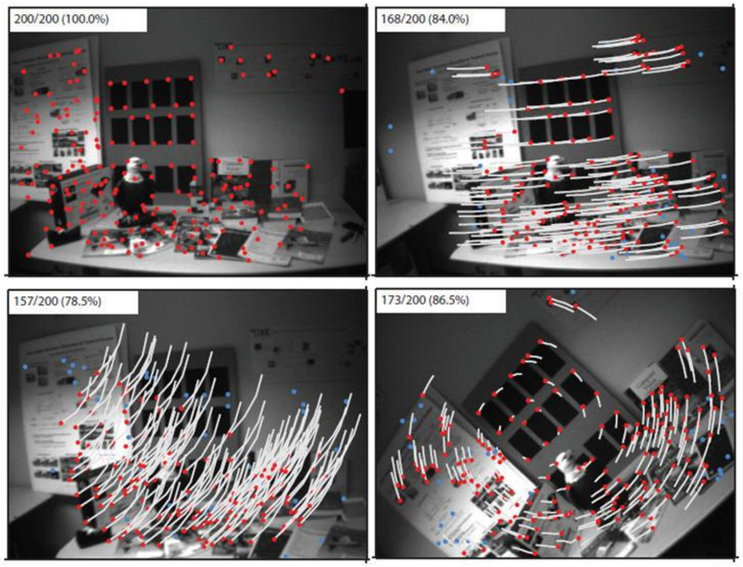
Feature detection & matching

- PTAM : Fast corner (PTAM) &ZNCC matching
- ORB-SLAM : ORB feature & ORB matching
- CoSLAM , Tango, VINS:
  - Intra-camera : Harris corner & KLT tracking
  - Inter-camera : ZNCC matching

Feature detection & matching

- Kanade-Lucas-Tomasi (KLT) feature tracker



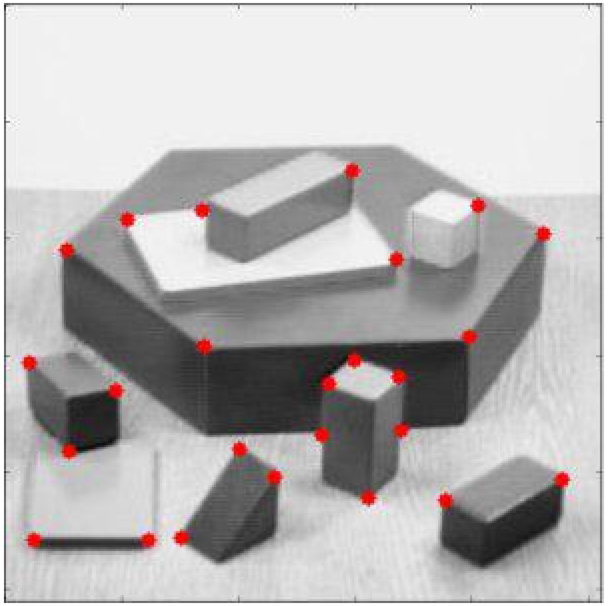


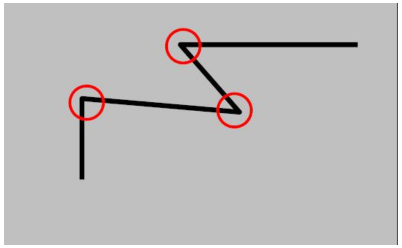
[1] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. International Joint Conference on Artificial Intelligence, pages 674–679, 1981.
[2] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.
[3] Jianbo Shi and Carlo Tomasi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition, pages 593–600, 1994.

Feature detection

- Harris corner detector

How do we define corner mathematically?

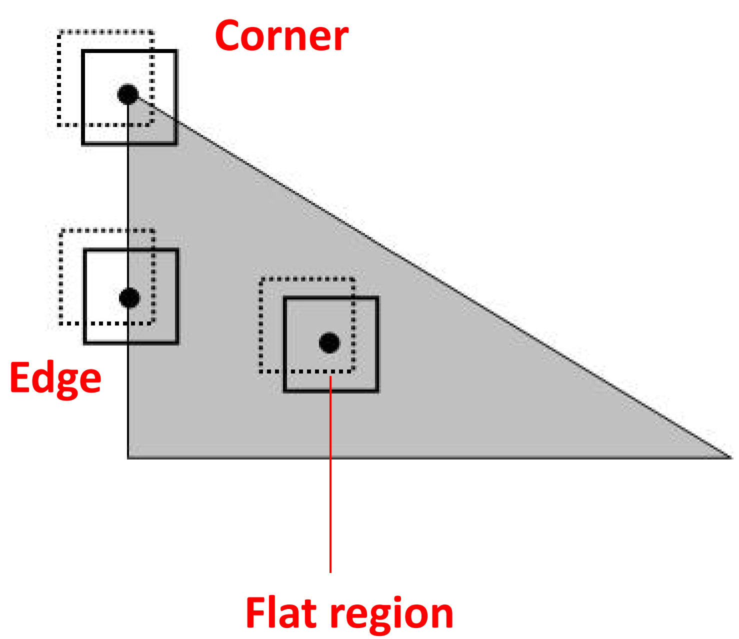




Harris, Chris, and Mike Stephens. "A combined corner and edge detector."Alvey vision conference. Vol. 15. 1988.

Feature detection

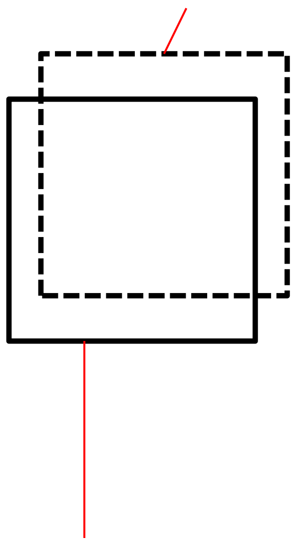
- Basic idea :
  – move the rectangle slightly around the original position and check the changes



- **Flat region**: No change in all direction
- **Edge :** No change along the edge direction
- **Corner:** Significant changes in all directions

Feature detection

- Comparing the original image patch and the shifted image patch can be mathematically written as



$$D(u,v) = \sum_{x,y\in\Omega}(\underbrace{I(x,y)}_{\text{Original patch}} - \underbrace{I(x+u,y+v)}_{\text{Shifted patch}})^2$$

Feature detection

- Assume that the image is locally smooth. Using first-order Taylor expansion, we get

$$I(x+u, y+v) = I(x,y) + \nabla I(x,y) \begin{bmatrix} u \\ v \end{bmatrix} = I(x,y) + [I_x, I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

- Therefore, we can write

$$D(u,v) = \sum_{x,y\in\Omega} (I(x,y) - I(x+u, y+v))^2$$

as

$$= [u, v] \boxed{\begin{pmatrix} \sum_{x,y\in\Omega} I_x(x,y) I_x(x,y) & \sum_{x,y\in\Omega} I_x(x,y) I_y(x,y) \\ \sum_{x,y\in\Omega} I_x(x,y) I_y(x,y) & \sum_{x,y\in\Omega} I_y(x,y) I_y(x,y) \end{pmatrix}} \begin{bmatrix} u \\ v \end{bmatrix}$$

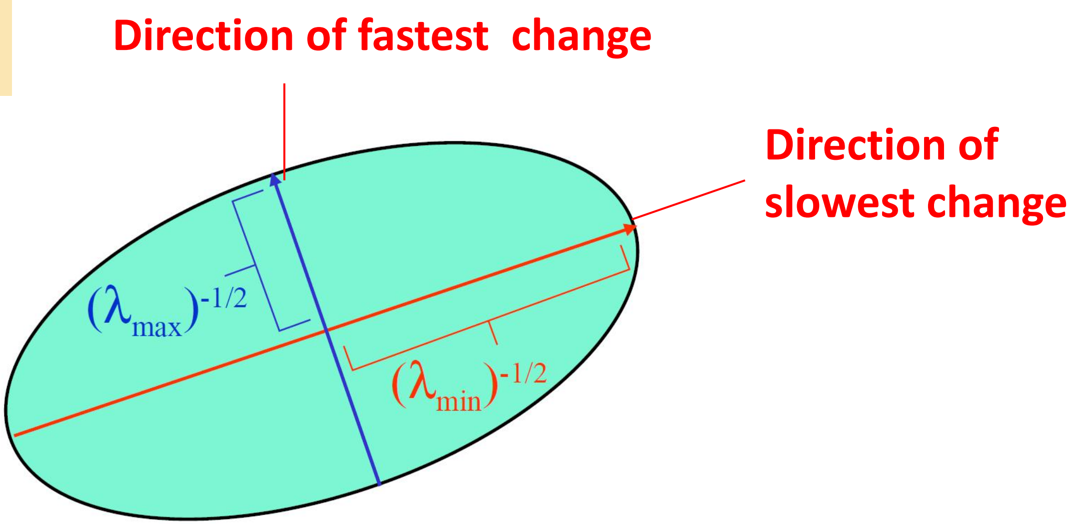
$$= [u, v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

Feature detection

- The intensity change is a quadratic function of the shift vector (u, v) $D(u,v) \approx [u,v] M \begin{bmatrix} u \\ v \end{bmatrix}$

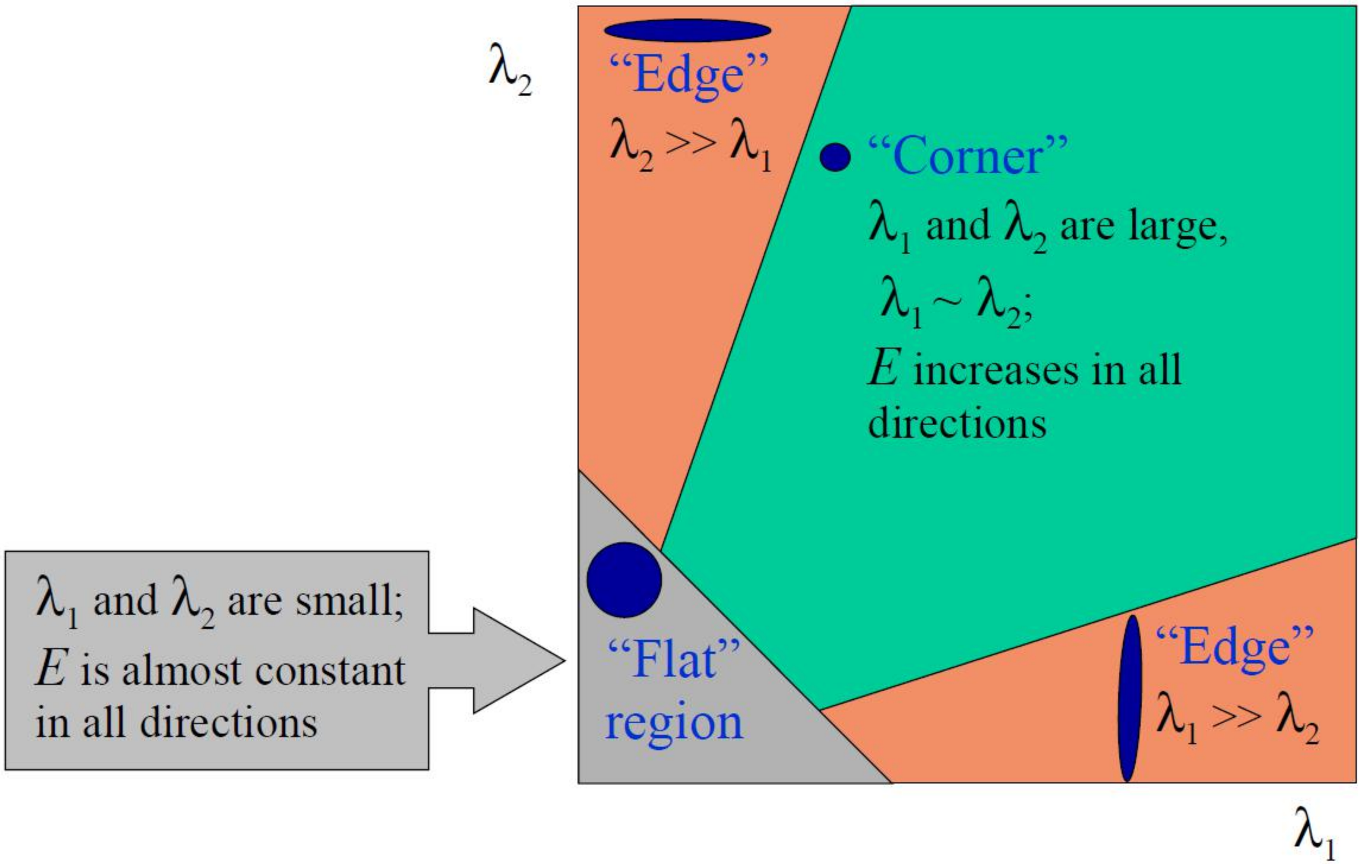
where $M = \sum_{x,y\in\Omega} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$

if we let D(u,v) = const, we can draw an ellipse as:



Feature detection

- Classify image points using eigenvalues of M:



Feature detection

- Measure of corner response
  - Harris method

$$R = det(M) - k(trace(M))^2$$
$$det(M) = \lambda_1$$
$$trace(M) = \lambda_1 + \lambda_2$$

k is an empirical constant (0.04 ~ 0.06)

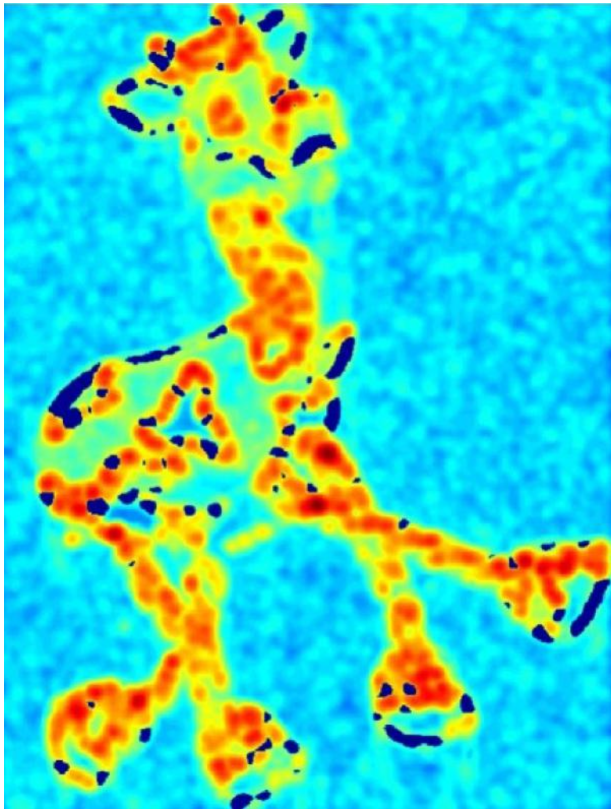
  - Shi-Tomasi method

$$R = min(\lambda_1, \lambda_2)$$

Shi, Jianbo, and Carlo Tomasi. "Good features to track." *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994.

Feature detection

- Harris detector : work flow



compute corner response

Feature detection

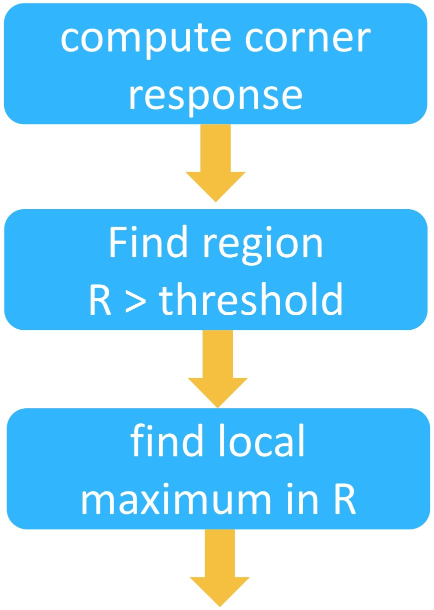
- Harris detector : work flow

compute corner response

Find region R > threshold

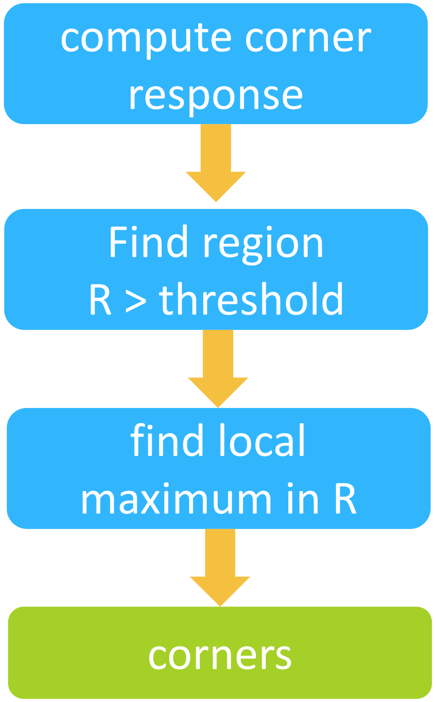
Feature detection

- Harris detector : work flow



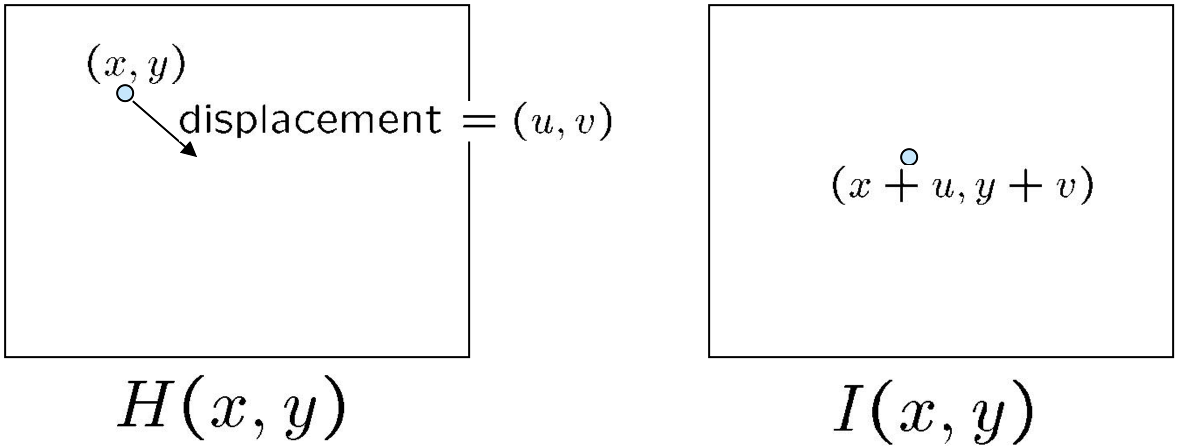
Feature detection

- Harris detector : work flow



Feature matching

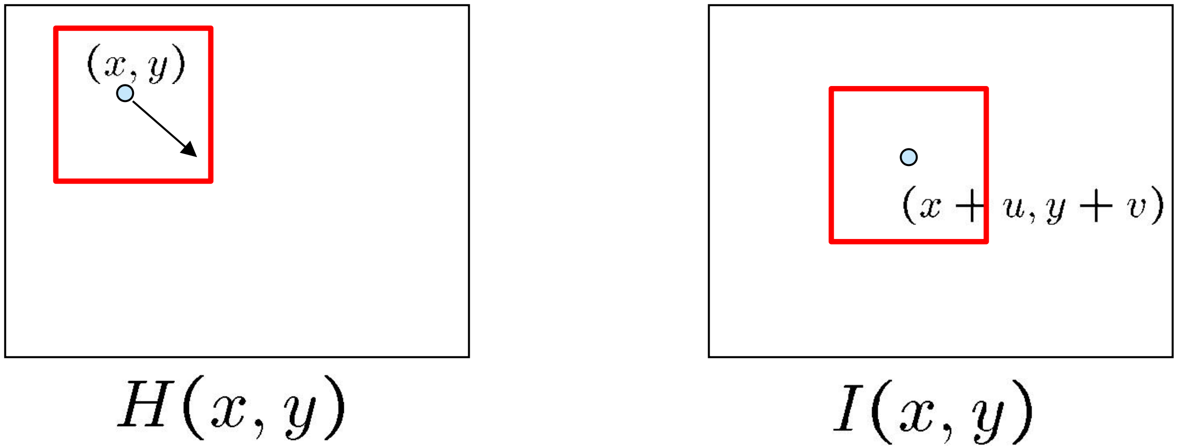
- Matching by tracking (for video sequence)
  - Lucas-Kanade optical flow algorithm



Given a feature point in the previous frame H, how to infer its position in the current frame I?

Feature matching

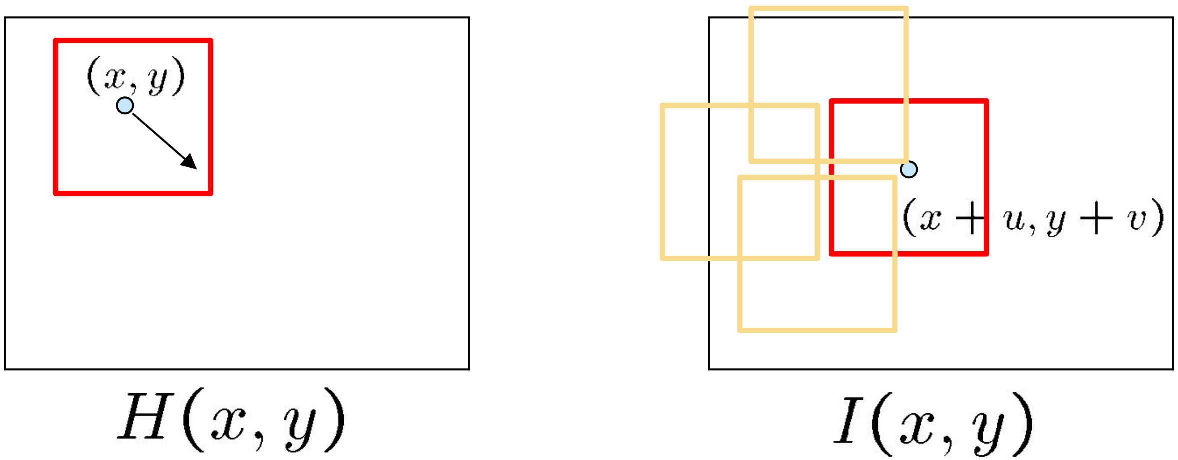
- Assumption
  - Displacement is small.
  - Color has little change.



Compare color difference between two patches.

Feature matching

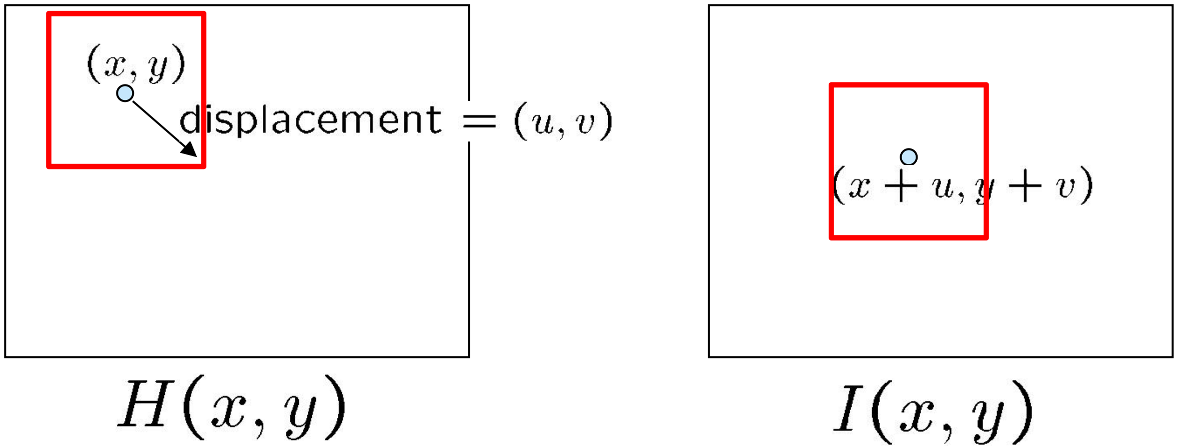
- A brute force searching method:
  - search the patch with the smallest color difference from original patch.



- **Inefficient:** searching in a 10x10 window requires 100 times of patch comparison.
- **Inaccurate:** operates on pixels

Feature matching

- Lucas-Kanade optical flow method



- We are trying to minimize the **S**um of **S**quared **D**ifference (**SSD**)

$$S(u, v) = \sum_{x,y\in\Omega}(I(x + u, y + v) - H(x, y))^2$$

Feature matching

- Local minimum: let the gradient be zero!

$$\nabla S = \begin{bmatrix} S_u \\ S_v \end{bmatrix} = 0$$

$$S(u,v) = \sum_{x,y\in\Omega} (\boxed{I(x+u, y+v)} - H(x,y))^2$$

Feature matching

- First-order Taylor expansion

$$I(x+u,y+v) = I(x,y) + \nabla I(x,y) \begin{bmatrix} u \\ v \end{bmatrix} = I + I_x u + I_y v$$

$$S(u,v) = \sum_{x,y\in\Omega} (\boxed{I(x+u,y+v)} - H(x,y))^2$$

$$= \sum_{x,y\in\Omega} (I - H + I_x u + I_y v)^2$$

$$\nabla S = \begin{bmatrix} S_u \\ S_v \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{x,y} I_x(I-H) + \sum_{x,y} I_x^2 u + \sum_{x,y} I_x I_y v \\ \sum_{x,y} I_y(I-H) + \sum_{x,y} I_x I_y u + \sum_{x,y} I_y^2 v \end{bmatrix}$$

Feature matching

- Finally we get the following equation

$$\begin{bmatrix} \sum_{x,y} I_x^2 + \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y + \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_{x,y} I_x (I - H) \\ \sum_{x,y} I_y (I - H) \end{bmatrix}$$

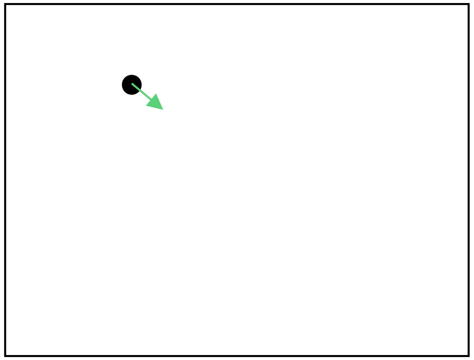
$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\Rightarrow \boxed{\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}}$$

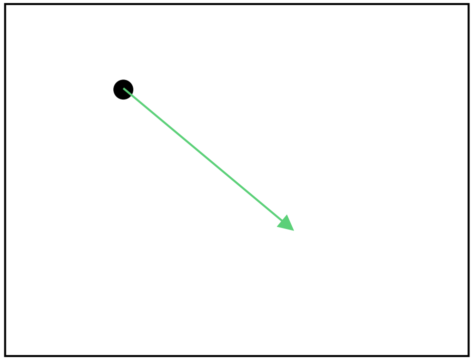
Finally, we get the displacement (flow)!

Feature matching

- Problem:
  - It assumes that the displacement is small (~ 1 pixel)
- How about **large** displacement value?



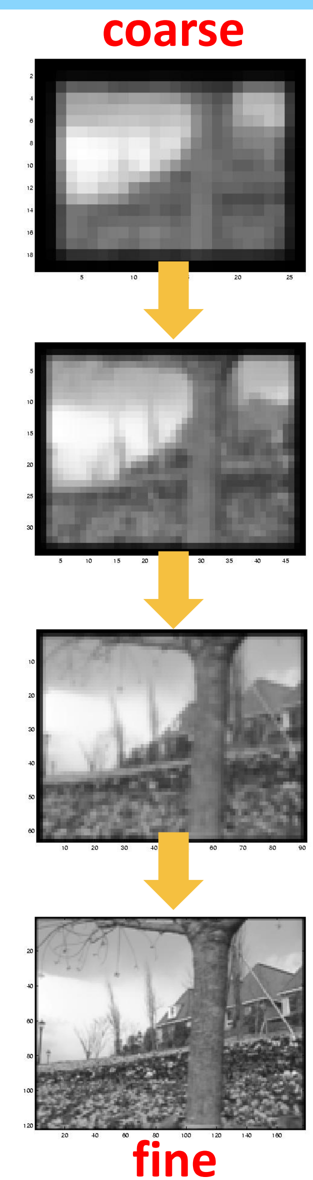
Small displacement



Large displacement

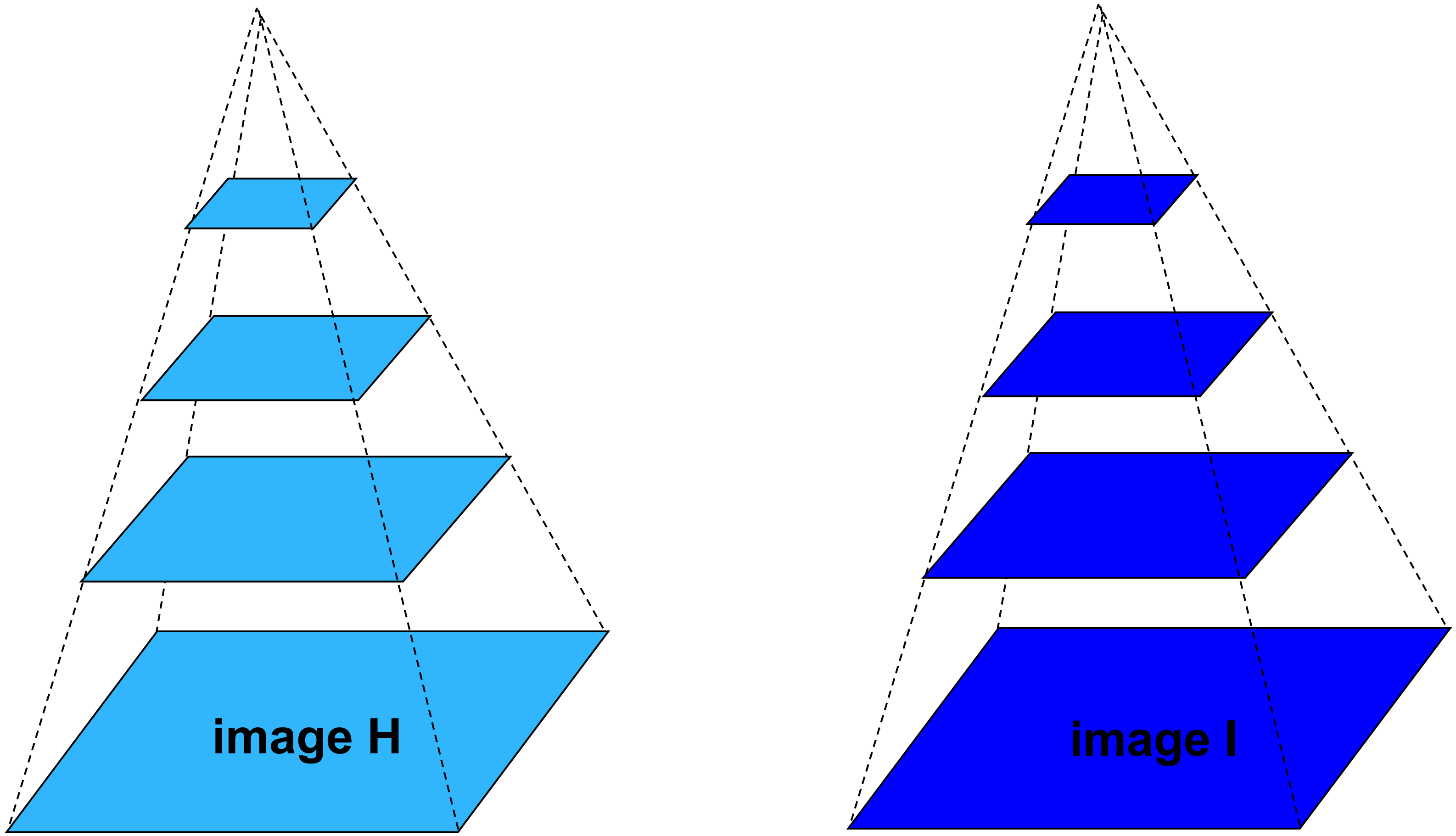
Feature matching

- Coarse-to-find manner
  - Compute displacement in low resolution images (coarse)
  - Use the coarse displacement value as an initialization and refine the displacement in high resolution images(fine)
  - repeatedly run above steps until the finest level has been reached.



Feature matching

- Build image pyramid

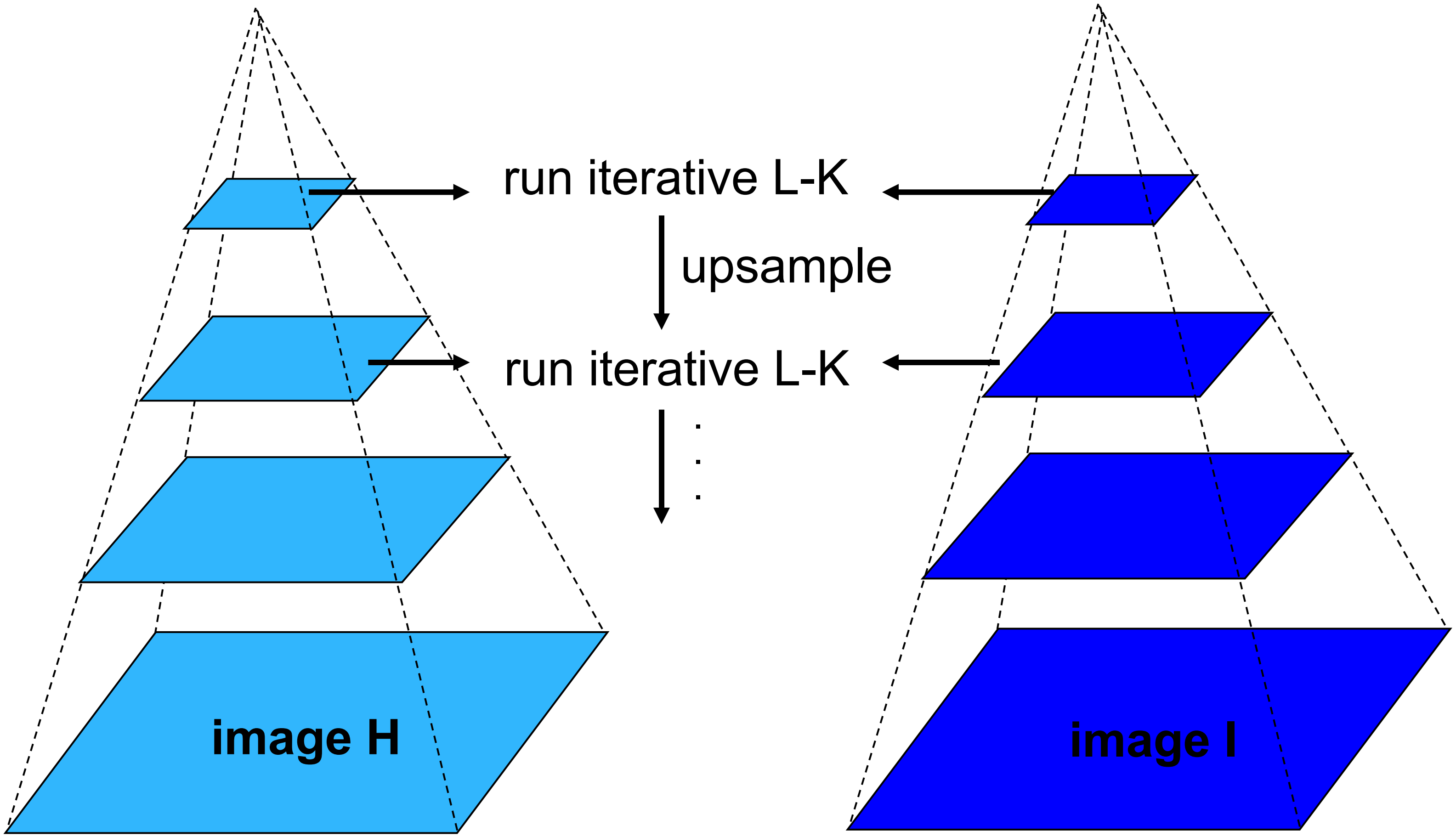


Gaussian pyramid of image H

Gaussian pyramid of image I

Feature matching

- Run Lucuas-Kanade algorithm in coarse-to-fine manner



Gaussian pyramid of image H

Gaussian pyramid of image I

Feature detection& matching

- OpenCV implementation

```
void goodFeaturesToTrack(InputArray image,
                         OutputArray corners,
                         int maxCorners,
                         double qualityLevel,
                         double minDistance,
                         InputArray mask=noArray(),
                         int blockSize=3,
                         bool useHarrisDetector=false,
                         double k=0.04 )
```

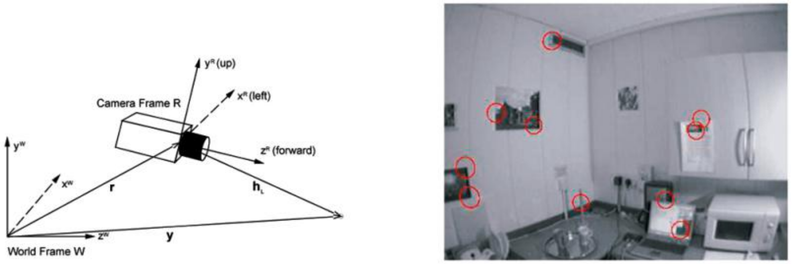
```
void calcOpticalFlowPyrLK(InputArray prevImg,
                          InputArray nextImg,
                          InputArray prevPts,
                          InputOutputArray nextPts,
                          OutputArray status,
                          OutputArray err,
                          Size winSize=Size(21,21),
                          int maxLevel=3,
                         TermCriteria   criteria,
                          int flags=0,
                          double minEigThreshold=1e-4 )
```

Outline

- Basic Theory
  - Projective geometry
  - Pinhole camera model
  - Camera calibration
  - Two camera geometry

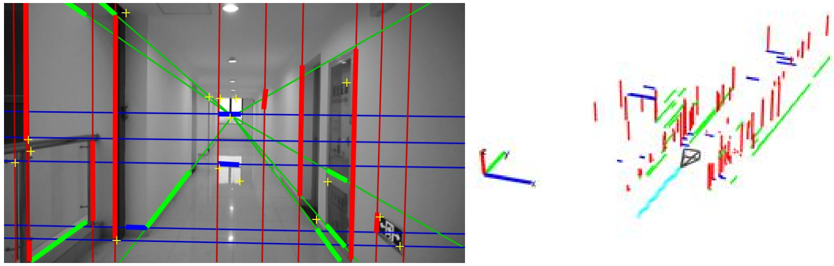
- Design a typical Visual SLAM system
- Two Visual SLAM systems:
  - **Extended Kalman Filter approach:**
    - **StructSLAM**
  - Visual SLAM for a group of robots:
    - CoLSAM

Extended Kalman Filter approach



MonoSLAM,2003

Davison, Andrew J., Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. "MonoSLAM: Real-time single camera SLAM." IEEE transactions on pattern analysis and machine intelligence 29, no. 6 (2007): 1052-1067.



StructSLAM, 2015

Zhou, Huizhong, Danping Zou, Ling Pei, Rendong Ying, Peilin Liu, and Wenxian Yu. "StructSLAM: Visual SLAM With Building Structure Lines."Vehicular Technology, IEEE Transactions on 64, no. 4 (2015): 1364-1375.

Kalman filter

- What is Kalman filter?
  - A Kalman filter is an **estimator** – i.e. infers parameters of interest from indirect, inaccurate and uncertain observations
  - It is **recursive** so that new measurements can be processed as they arrive.
  - It is **optimal** – i.e. if the noise is Gaussian, Kalman filter minimizes the mean square error of the estimated parameters.

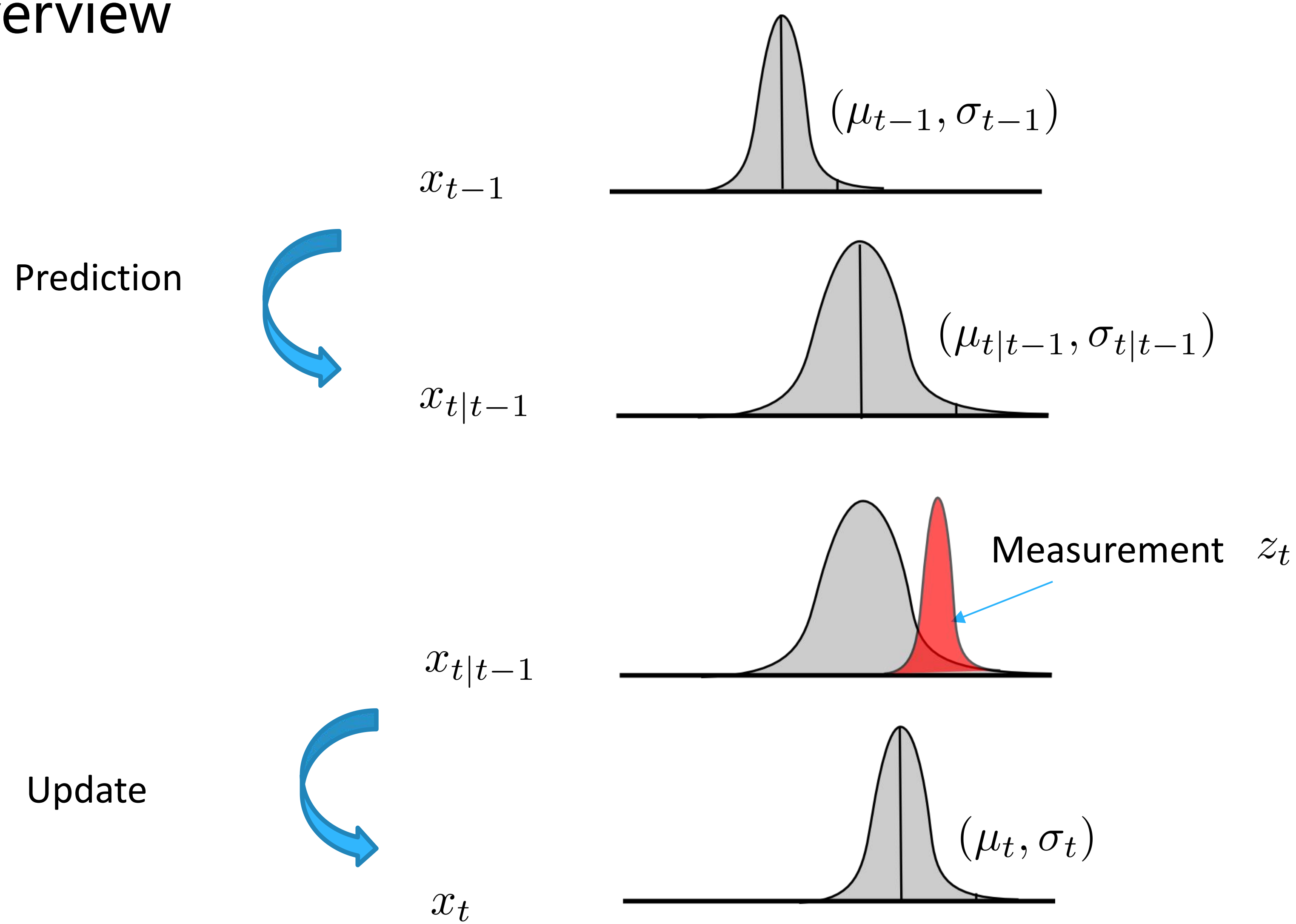
Rudolf Emil Kálmán, co-inventor and developer of the Kalman filter.

Kalman filter

- Why is Kalman filter so popular?
  - Good results in practice due to optimality and structure.
  - Convenient form for online real time processing.
  - Easy to formulate and implement given a basic understanding.
  - Measurement equations need not be inverted.

Kalman filter

- Overview



Kalman filter

- Linear dynamic model

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- F_t is the state transition model which is applied to the previous state x_{t-1};
- B_t is the control-input model which is applied to the control vector u_t;
- w_t is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_t.

$$w_t \sim \mathcal{N}(0, Q_t)$$

Kalman filter

- Observation model (measurement model)

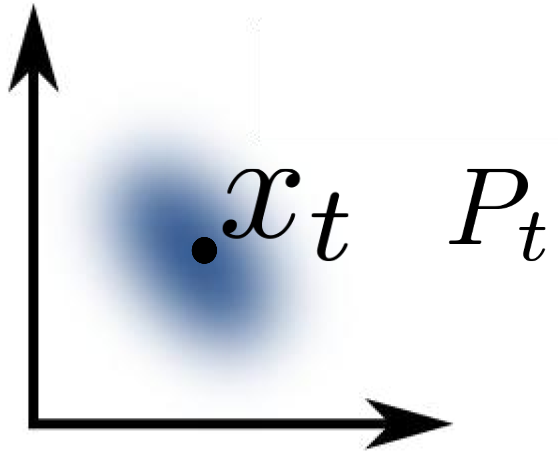
$$z_t = H_t x_t + n_t$$

  - H_t is the observation model which maps the true state space into the observed space.
  - n_t is the observation noise which is assumed to be zero mean Gaussian white noise with covariance R_t

$$n_t \sim \mathcal{N}(0, R_t)$$

Kalman filter

- At each time step, Kalman filter try to compute both the state estimation and the state covariance



Kalman filter

- Prediction
  - State prediction - use the dynamic model to predict the state in the next time step:

$$x_{t|t-1} = F_t x_{t-1} + B_t u_t$$

  - Uncertainty of prediction – propagate the covariance

$$P_{t|t-1} = F_t P_{t-1} F_t^T + Q_t$$

Kalman filter

- Correction/Update
  - Compute innovation (measurement residual)

$$y_t = z_t - H_t x_{t|t-1}$$

  - Get innovation covariance

$$S_t = H_t P_{t|t-1} H_t^T + R_t$$

Kalman filter

- Correction/Update
  - Compute Kalman Gain

$$K_t = P_{t|t-1} H_t^T S_t^{-1}$$

  - Update state estimate

$$x_t = x_{t|t-1} + K_t y_t$$

  - Update state covariance

$$P_t = (I - K_t H_t) P_{t|t-1}$$

Extended Kalman filter

- Nonlinear dynamic model
- Nonlinear observation model

$$x_t = f(x_{t-1}, u_t) + w_t$$
$$z_t = h(x_t) + v_t$$

Prediction

$$x_{t|t-1} = f(x_{t-1}, u_t)$$

$$P_{t|t-1} = F_t P_{t-1} F_t^T + Q_t$$

$$F_t = \frac{\partial f}{\partial x}\Big|_{x_t, u_t}$$

Observation

$$y_t = z_t - h(x_{t|t-1})$$

$$S_t = H_t P_{t|t-1} H_t^T + R_t$$

$$H_t = \frac{\partial h}{\partial x}\Big|_{x_t}$$

Extended Kalman Filter approach

- EKF VSLAM tries to estimate *a state variable* that contains current robot state(orientation, position, velocity) and all landmarks .

$$\mathbf{x} = \begin{bmatrix} \mathcal{R} \\ \mathcal{M} \end{bmatrix} = \begin{bmatrix} \mathcal{R} \\ \mathcal{L}_1 \\ \vdots \\ \mathcal{L}_n \end{bmatrix}$$

Robot state

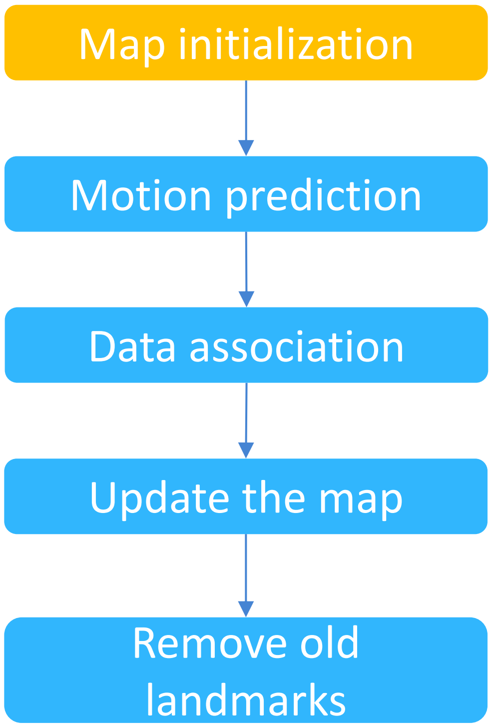
Landmarks (points, lines)

This is also called ***map***

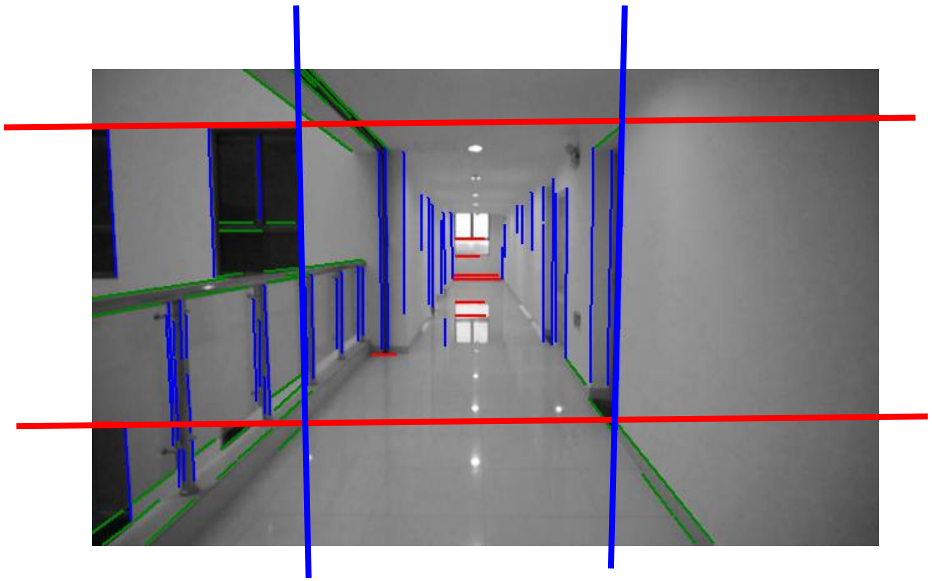
$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{\mathcal{R}\mathcal{R}} & \mathbf{P}_{\mathcal{R}\mathcal{M}} \\ \mathbf{P}_{\mathcal{M}\mathcal{R}} & \mathbf{P}_{\mathcal{M}\mathcal{M}} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{\mathcal{R}\mathcal{R}} & \mathbf{P}_{\mathcal{R}\mathcal{L}_1} & \cdots & \mathbf{P}_{\mathcal{R}\mathcal{L}_n} \\ \mathbf{P}_{\mathcal{L}_1\mathcal{R}} & \mathbf{P}_{\mathcal{L}_1\mathcal{L}_1} & \cdots & \mathbf{P}_{\mathcal{L}_1\mathcal{L}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{\mathcal{L}_n\mathcal{R}} & \mathbf{P}_{\mathcal{L}_n\mathcal{L}_1} & \cdots & \mathbf{P}_{\mathcal{L}_n\mathcal{L}_n} \end{bmatrix}$$

Extended Kalman Filter approach

- The workflow of a typical EKF vSLAM system.



- Initialize the state variable and the covariance



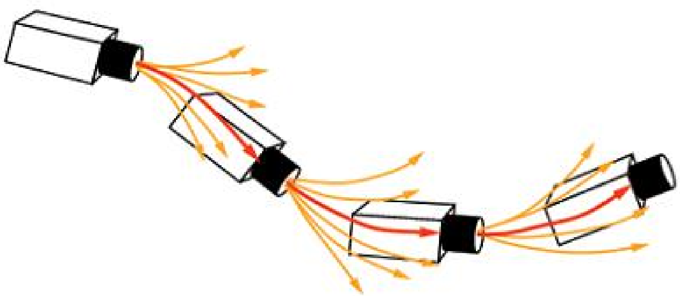
Extended Kalman Filter approach

- The workflow of a typical EKF vSLAM system.

Map initialization
↓
Motion prediction
↓
Data association
↓
Update the map
↓
Remove old landmarks

- Predict the state and propagate the covariance

$$\overline{\mathbf{x}} \leftarrow f(\overline{\mathbf{x}}, \mathbf{u}, 0)$$
$$\mathbf{P} \leftarrow \mathbf{F_x} \mathbf{P} \mathbf{F_x}^\top + \mathbf{F_n} \mathbf{N} \mathbf{F_n}^\top$$

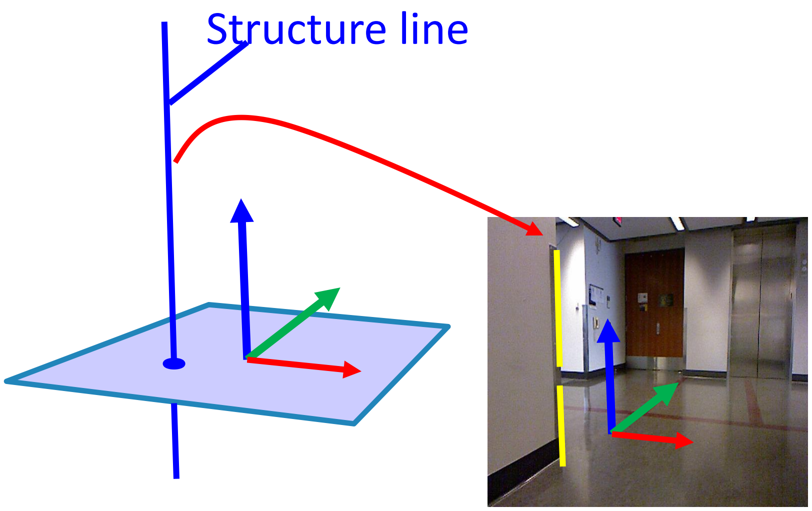


Extended Kalman Filter approach

- The workflow of a typical EKF vSLAM system.

Map initialization → Motion prediction → **Data association** → Update the map → Remove old landmarks

- Find the corresponding features of the landmarks in the image



Extended Kalman Filter approach

Map initialization

Motion prediction

Data association

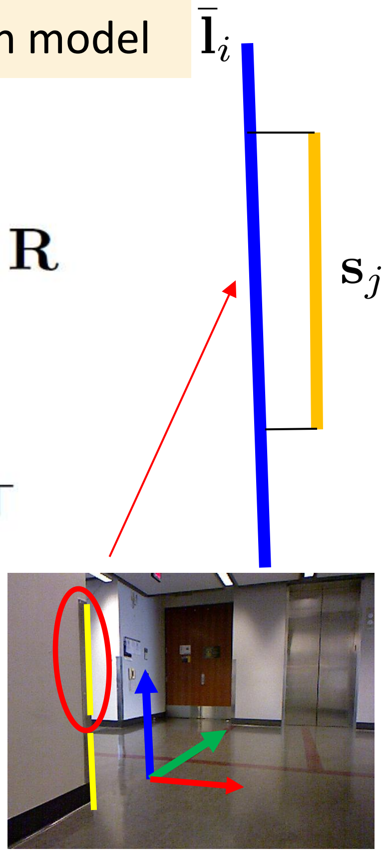
Update the map

Remove old landmarks

- Compute Kalman Gain according to the observation model and use it to update the state.

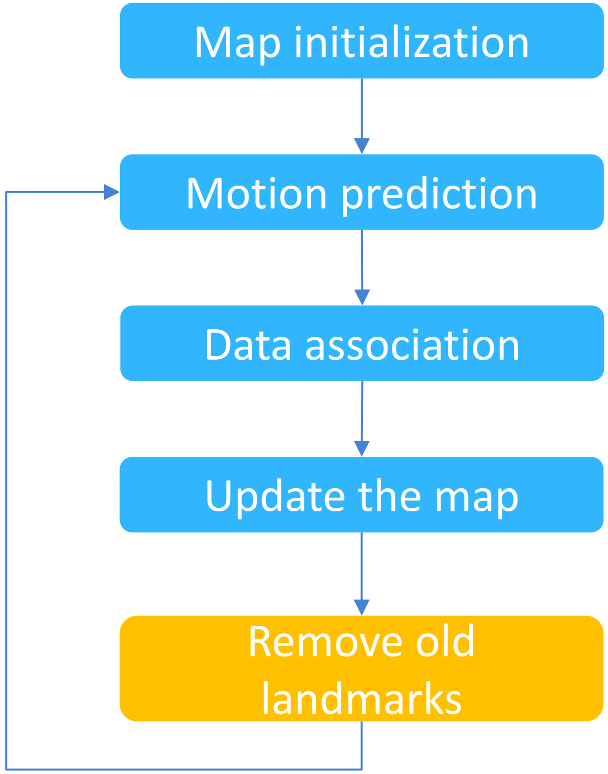
$h(\bar{\mathbf{x}})$ is the observation model

$$
\begin{aligned}
\bar{\mathbf{z}} &= \mathbf{y} - \boxed{h(\bar{\mathbf{x}})} \\
\mathbf{Z} &= \mathbf{H}_{\mathbf{x}}\mathbf{P}\mathbf{H}_{\mathbf{x}}^{\top} + \mathbf{R} \\
\mathbf{K} &= \mathbf{P}\mathbf{H}_{\mathbf{x}}^{\top}\mathbf{Z}^{-1} \\
\bar{\mathbf{x}} &\leftarrow \bar{\mathbf{x}} + \mathbf{K}\bar{\mathbf{z}} \\
\mathbf{P} &\leftarrow \mathbf{P} - \mathbf{K}\mathbf{Z}\mathbf{K}^{\top}
\end{aligned}
$$



Extended Kalman Filter approach

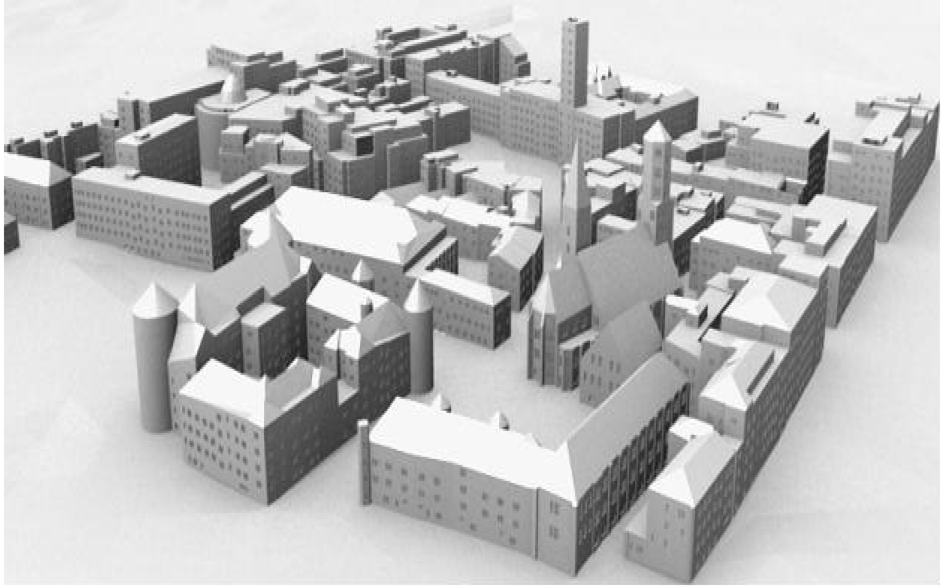
- The workflow of a typical EKF vSLAM system.



- To limit the dimension of the map without growing to a very large value.

StructSLAM

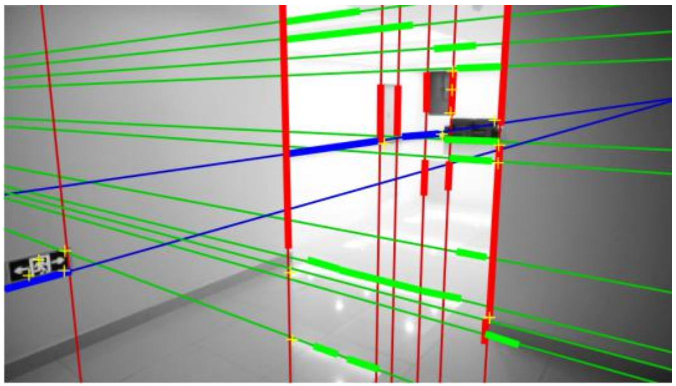
- Basic idea: Most man-made scenes exhibit strong regularity in structures, especially the indoor spaces.
- This regularity can be simply described as 'Manhattan world'.



- Perpendicular surfaces
- Have several dominant directions

StructSLAM

- A new kind of line features named as *Structure Lines*

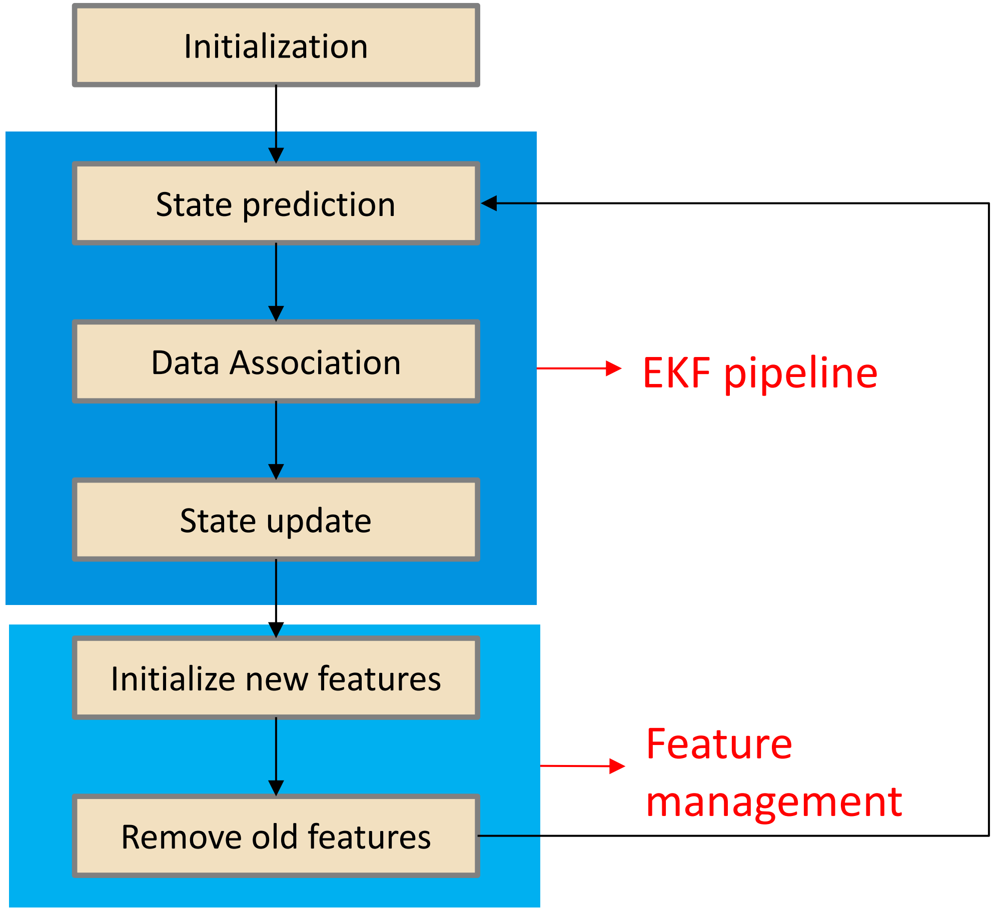


- **Structure Lines here are those lines who are aligned with x,y,z axes.**

- Motivations:
  - Structure lines encode the **global orientation information** in the image
  - Lines are better landmarks in texture-less scenes (like many indoor scenes with only white walls) than points.

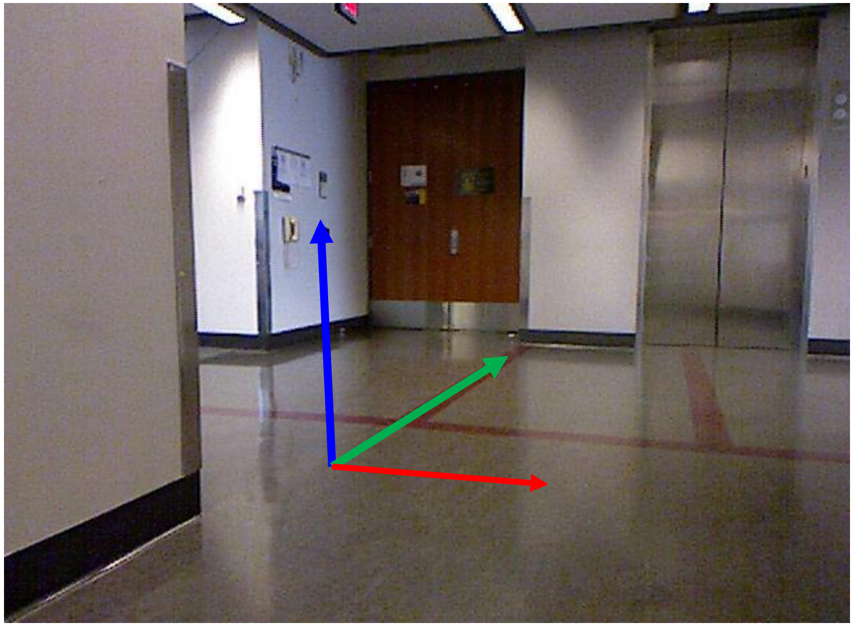
Pipeline of StructSLAM

- Extended Kalman filter:
  - Simple to be implemented
  - Easily fused with other sensors (IMU, odometers)
  - More robust than structure-from-motion pipeline.



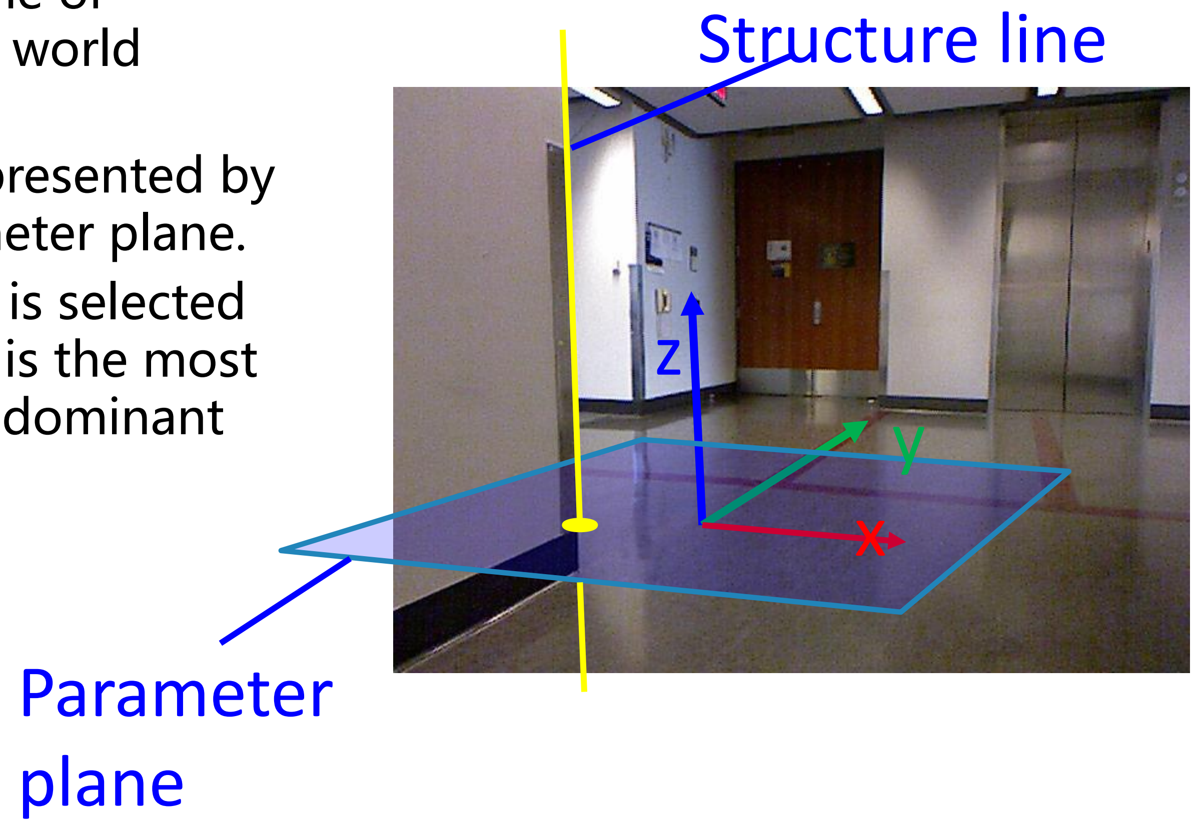
Dominant directions

- The man-made world are generally dominated by several directions.
  - Vertical direction (always points to the sky)
  - Horizontal directions (are usually perpendicular to each other, although not always)



Parameter planes

- Parameter plane is one of xy,yz,xz planes of the world frame.
- A structure line is represented by a point on the parameter plane.
- The parameter plane is selected so as to make sure it is the most perpendicular to the dominant direction.



Structure line

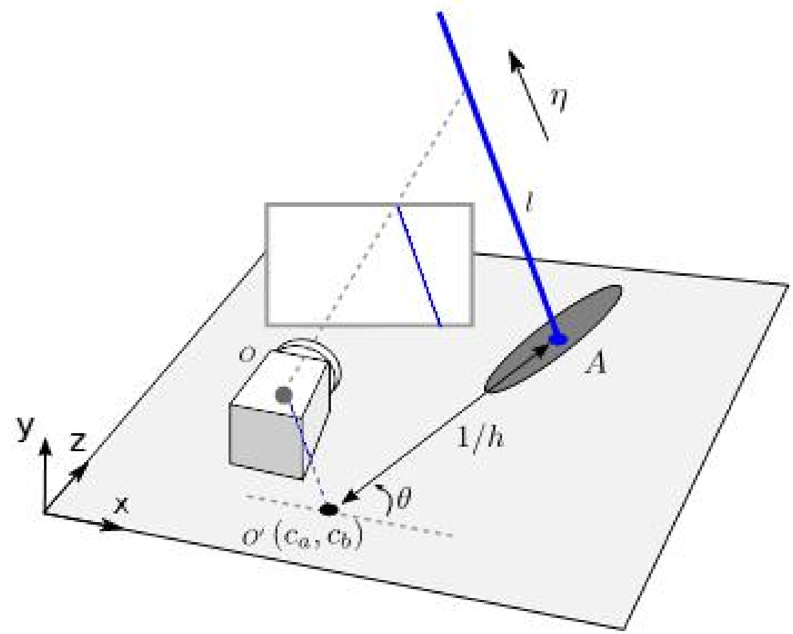
- Each structure line is represented by a point on the parameter plane, denoted by a 4x1 vector.
- It is in fact a 2D inverse depth representation*

$$\mathbf{l} = \begin{pmatrix} c_a \\ c_b \\ \theta \\ h \end{pmatrix}$$

Projection of camera center

Direction

Inverse depth



* Montiel, J. M. M., Javier Civera, and Andrew J. Davison. "Unified inverse depth parametrization for monocular SLAM." *analysis* 9 (2006): 1.

StructSLAM – State representation

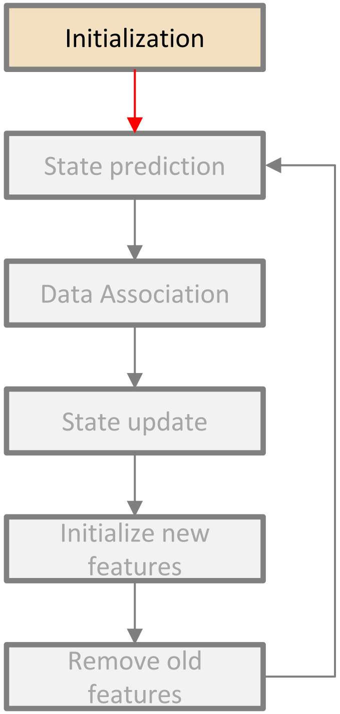
- State vector and covariance matrix (MonoSLAM)

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{x}_p \\ \boxed{\mathbf{x}_l} \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{cc} & \boldsymbol{\Sigma}_{cp} & \boldsymbol{\Sigma}_{cl} \\ \boldsymbol{\Sigma}_{pc} & \boldsymbol{\Sigma}_{pp} & \boldsymbol{\Sigma}_{pl} \\ \boldsymbol{\Sigma}_{lc} & \boldsymbol{\Sigma}_{lp} & \boldsymbol{\Sigma}_{ll} \end{bmatrix}$$

Camera pose + Points + Structure Lines

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{p}^w \\ \mathbf{q}^{wc} \\ \mathbf{v}^w \\ \omega^c \end{bmatrix} \qquad \mathbf{x}_p = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \end{bmatrix} \qquad \boxed{\mathbf{x}_p = \begin{bmatrix} \mathbf{l}_1 \\ \mathbf{l}_2 \\ \vdots \end{bmatrix}}$$

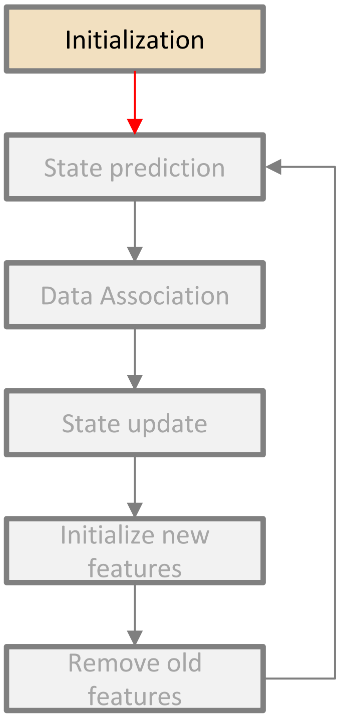
Initialization



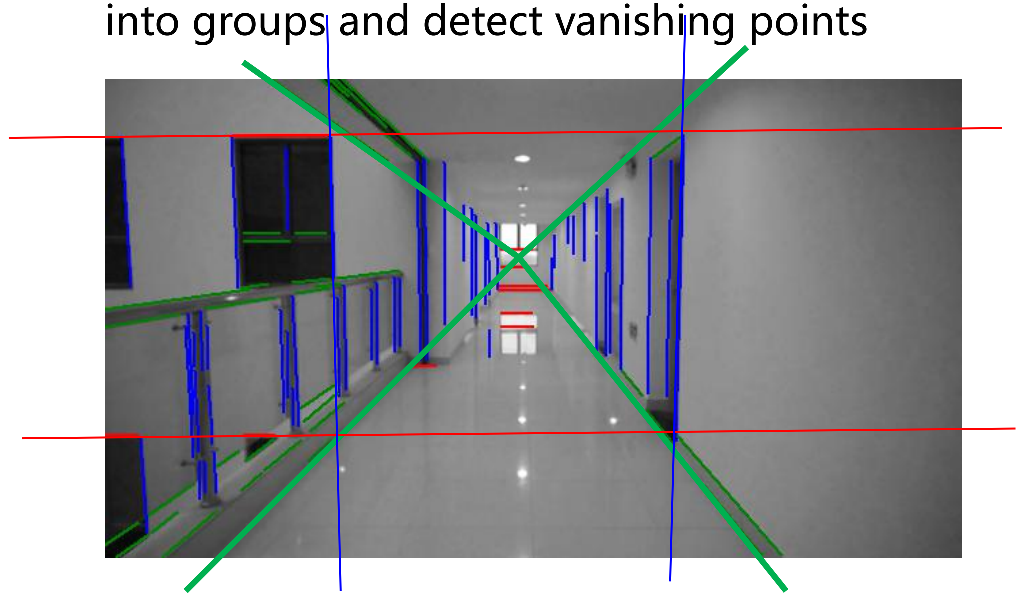
- Step 1:
  - Use LSD line detector* to detect line segments on the image.

* Von Gioi, Rafael Grompone, et al. "LSD: A fast line segment detector with a false detection control." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 4 (2008): 722-732.

Initialization

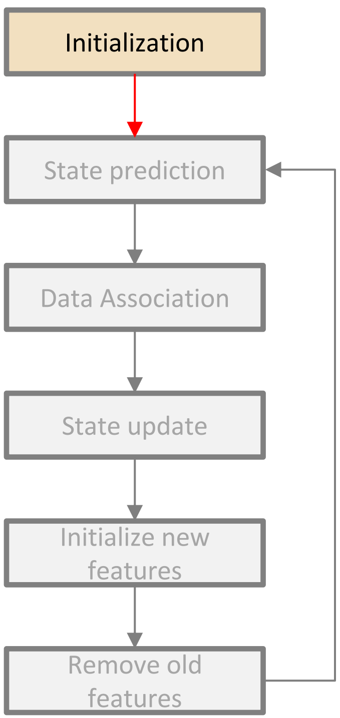


- Step 2:
  - Apply J-linkage* to classify parallel line segments into groups and detect vanishing points

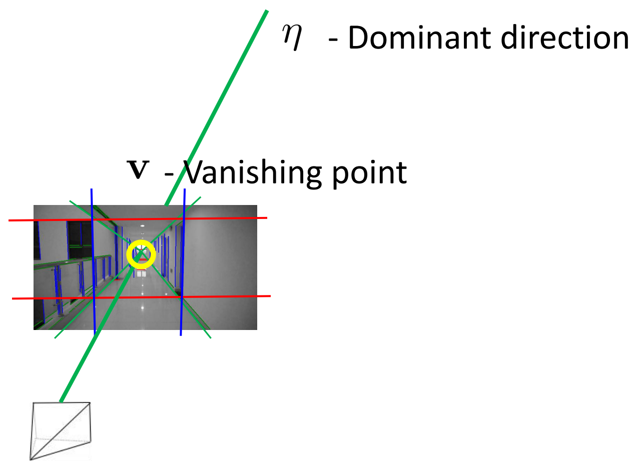


*Toldo, Roberto, and Andrea Fusiello. "Robust multiple structures estimation with j-linkage." *Computer Vision–ECCV 2008*. Springer Berlin Heidelberg, 2008. 537-547.

Initialization



- Step 3:
  - Estimate the dominant direction from the vanishing points.



$$\mathbf{v} = \mathbf{K}\mathbf{R}^{cw}\eta$$

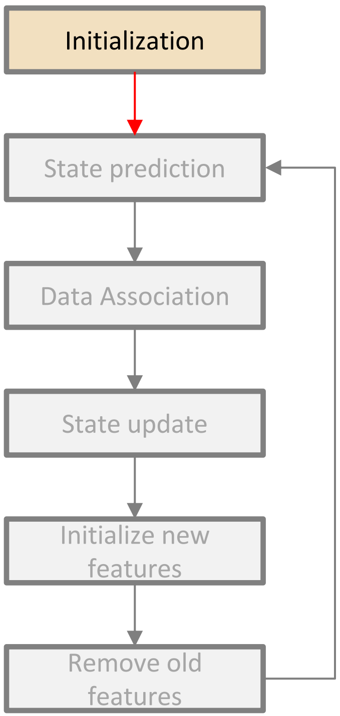
$$\eta \propto \mathbf{R}^{wc}\mathbf{K}^{-1}\mathbf{v}$$

$\mathbf{K}$: Camera intrinsic

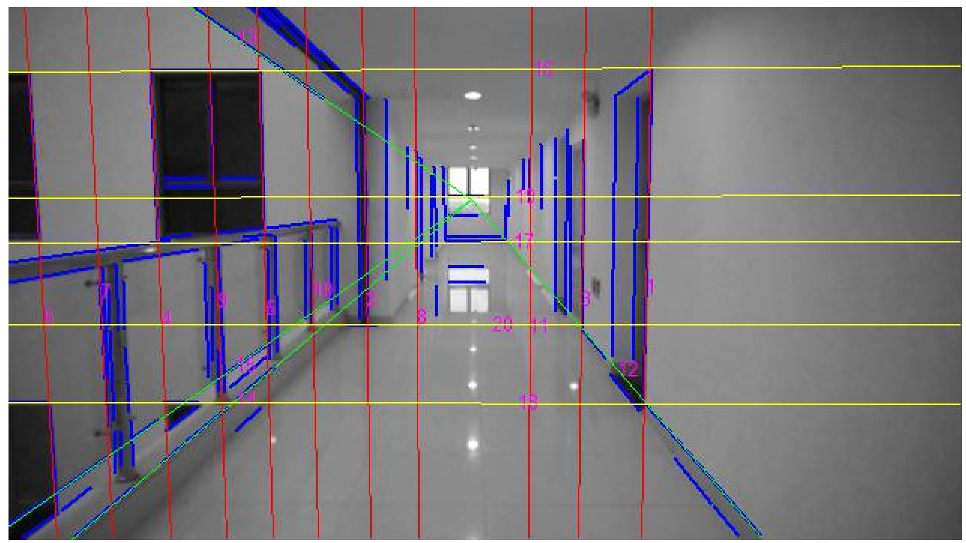
$\mathbf{R}^{cw}$: Rotation from the world frame to the camera frame

$$\mathbf{R}^{wc} = (\mathbf{R}^{cw})^T$$

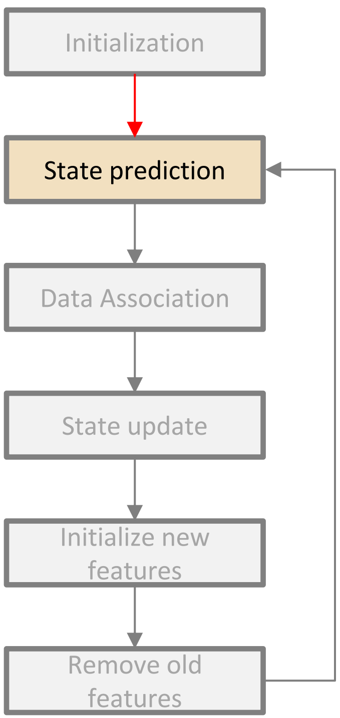
Initialization



- Step 4:
  - Refine the dominant directions by non-linear least square optimization
  - Initialize new lines (See the feature management section)



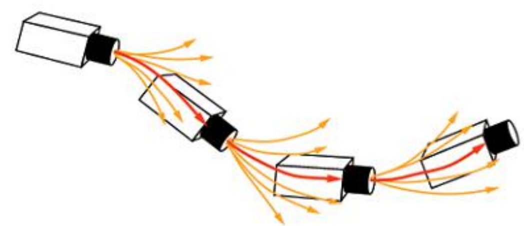
State prediction



- Camera – constant velocity or odometer data (if available)

$$\mathbf{f}_c(\mathbf{x}_c) = \begin{pmatrix} \bar{\mathbf{p}}^w \\ \bar{\mathbf{q}}^{wc} \\ \bar{\mathbf{v}}^w \\ \bar{\omega}^c \end{pmatrix} = \begin{pmatrix} \mathbf{p}^w + \mathbf{v}^w \Delta t \\ \mathbf{q}^{wc} \cdot \mathbf{q}(\omega^c) \Delta t \\ \mathbf{v}^w \\ \omega^c \end{pmatrix}.$$

Odometer velocity

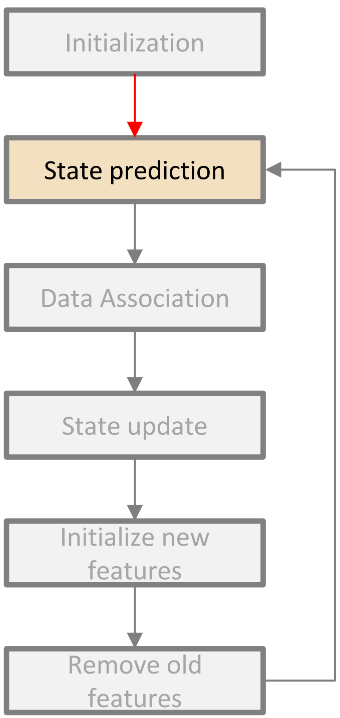


- Landmarks – static

$$\mathbf{f}_p(\mathbf{x}_p) = \mathbf{x}_p \qquad \mathbf{f}_l(\mathbf{x}_l) = \mathbf{x}_l$$

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} \mathbf{f}_c(\mathbf{x}_c) \\ \mathbf{x}_p \\ \mathbf{x}_l \end{bmatrix} + \mathbf{n}$$

State prediction



- Covariance propagation

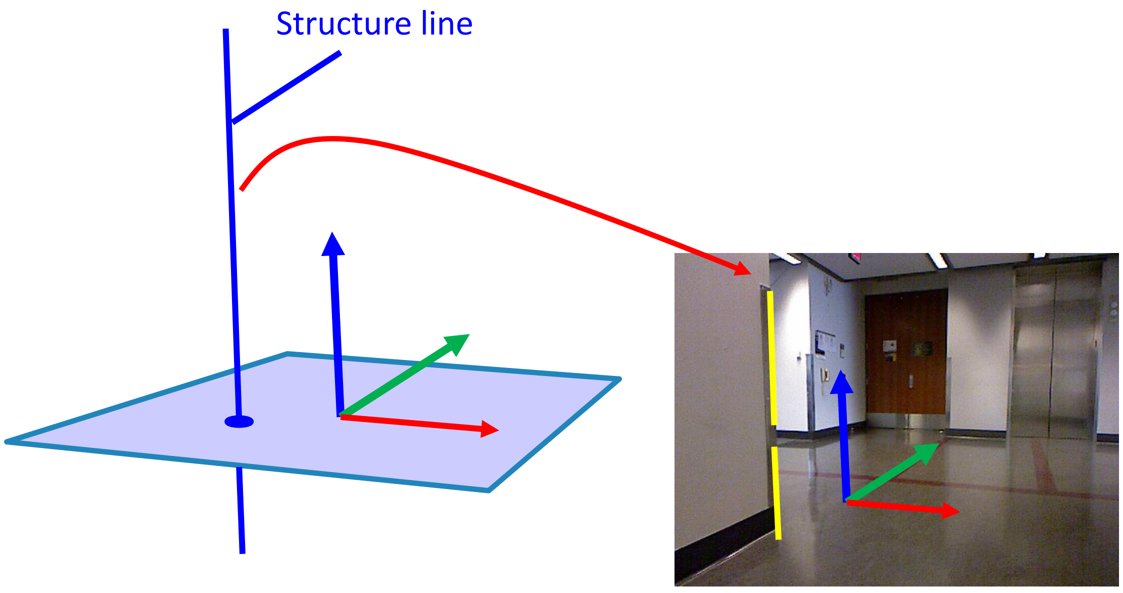
$$\bar{\mathbf{\Sigma}} = \mathbf{F_x}\mathbf{\Sigma}\mathbf{F_x}^T + \mathbf{F_n}\mathbf{\Sigma_n}\mathbf{F_n}^T$$

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} \mathbf{f}_c(\mathbf{x}_c) \\ \mathbf{x}_p \\ \mathbf{x}_l \end{bmatrix} + \mathbf{n}$$

$$\mathbf{F_x} = \begin{bmatrix} \frac{\partial \mathbf{f}_c}{\partial \mathbf{x}_c} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \qquad \mathbf{F_n} = \begin{bmatrix} \frac{\partial \mathbf{f}_c}{\partial \mathbf{n}} \\ \mathbf{0} \end{bmatrix}$$

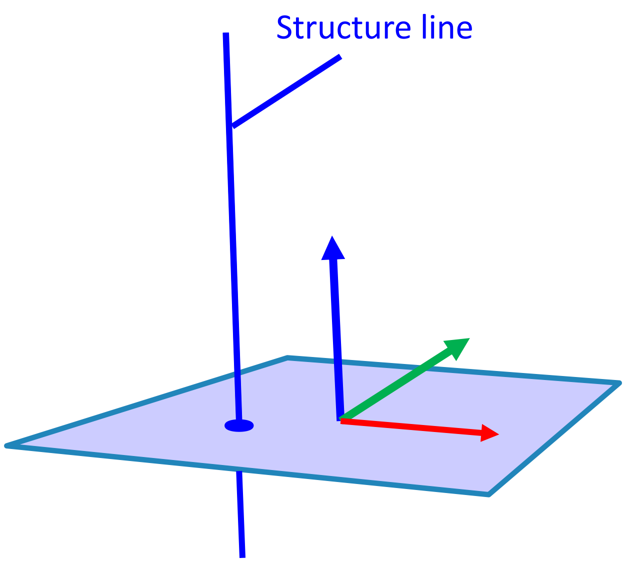
Data association

- Find the line segments corresponding to the structure line

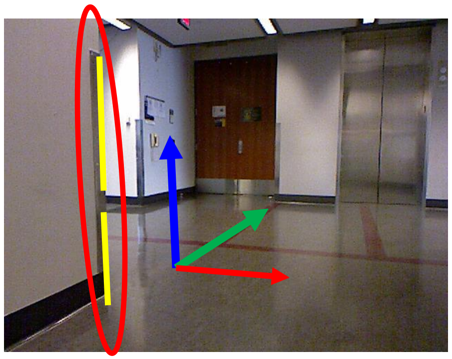


Data association

- Find the line segments corresponding to the structure line

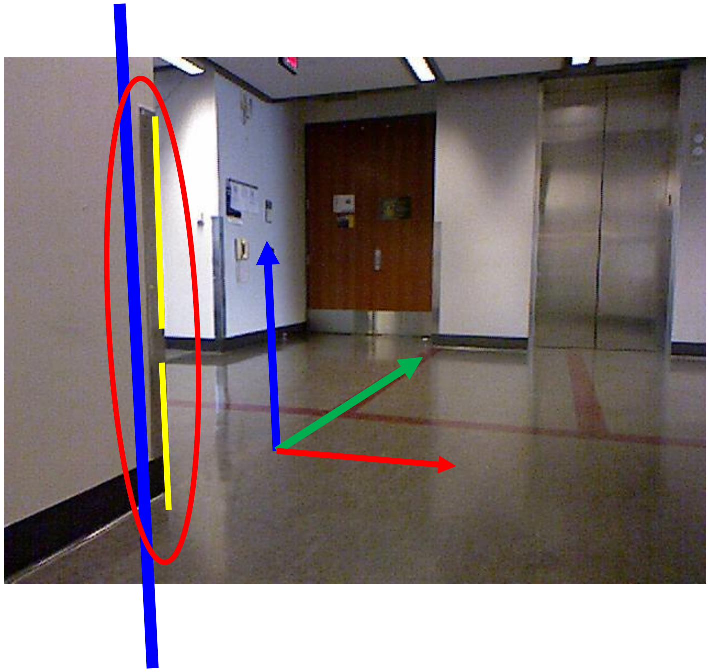


- One-to-multiple matching
- There could be false matchings (outliers).



Data association

- Step1 : Get candidate matching by



χ^2-distance

$$\chi^2 = \mathbf{r}_i^T (\mathbf{H}_i \mathbf{H}_i^T)^{-1} \mathbf{r}_i^T$$

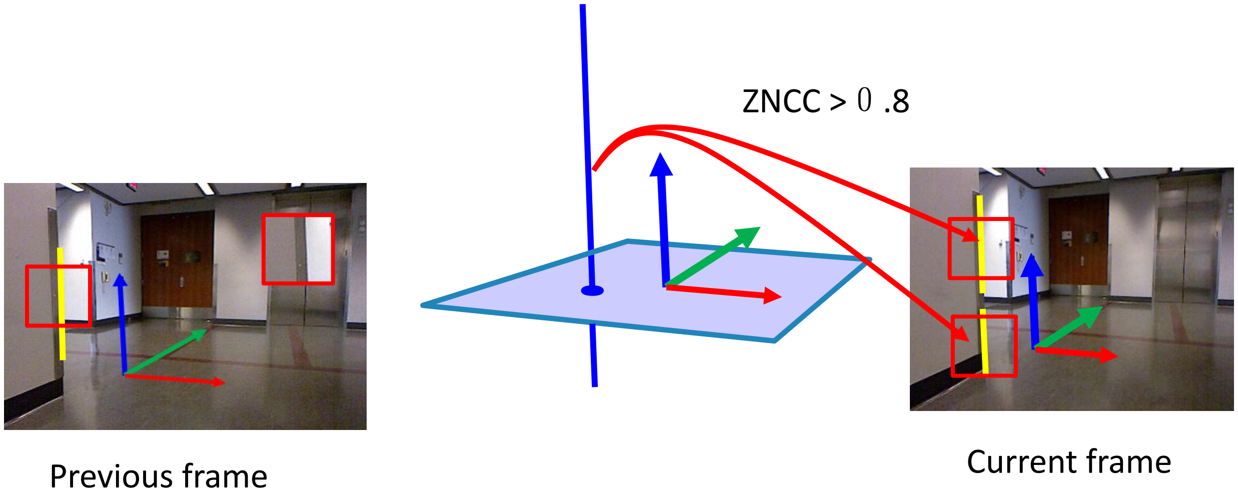
$\mathbf{r}_i$: residual vector

$\mathbf{H}_i$:Jacobian of observation function

$$\boxed{\chi^2 < 5.99 \quad (Probablity > 95\%)}$$

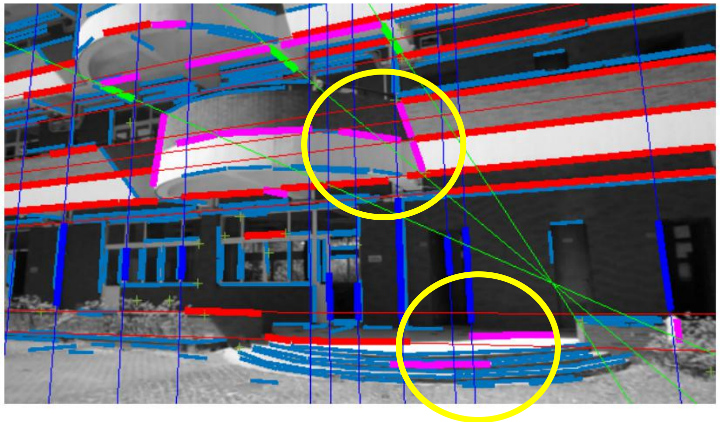
Data association

- Step 2: Comparing appearance by ZNCC (zero mean normalized cross-correlation)

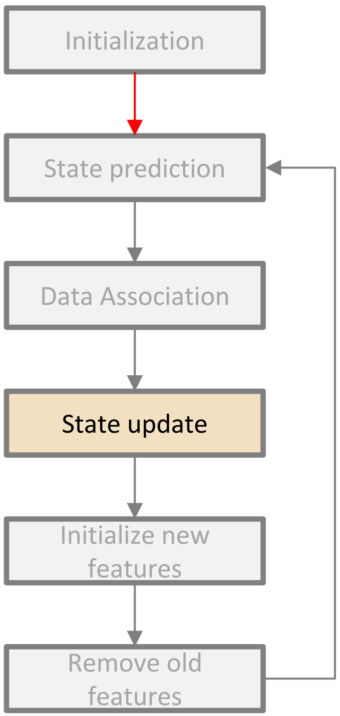


Data association

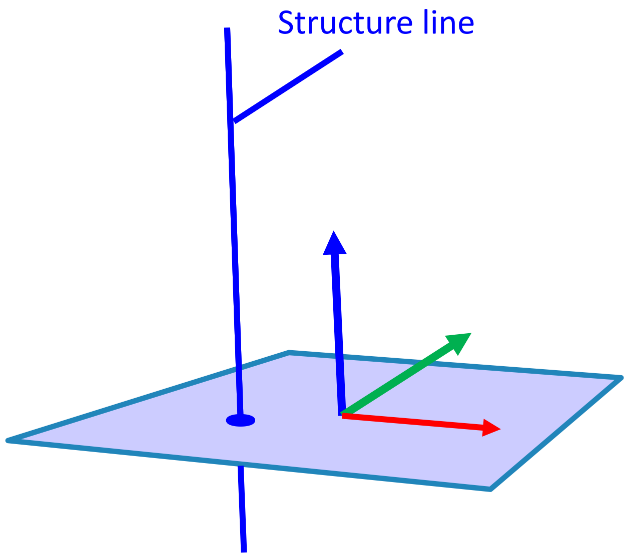
- Step3 : One-feature RANSAC to eliminate false matchings (outliers):
  - Randomly sample a candidate matching
  - Run a tentative EKF update using the sampled matching and check the number of inliers
  - Keep the inlier set with the maximum number
  - Repeat the above steps
  - Use the inlier set to run EKF u

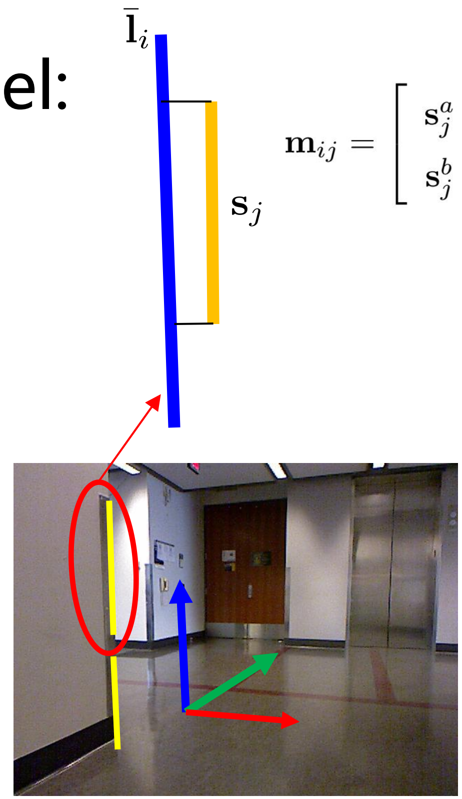


State update



- Observation model:

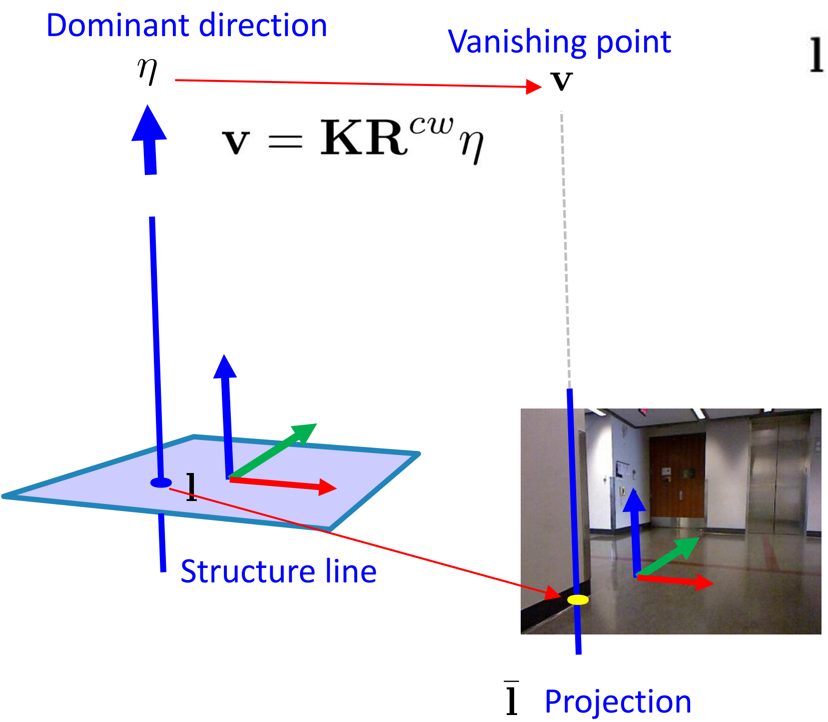




$$\mathbf{m}_{ij} = \begin{bmatrix} \mathbf{s}_j^a \cdot \bar{\mathbf{l}}_i / \sqrt{(\bar{l}_i^1)^2 + (\bar{l}_i^2)^2} \\ \mathbf{s}_j^b \cdot \bar{\mathbf{l}}_i / \sqrt{(\bar{l}_i^1)^2 + (\bar{l}_i^2)^2} \end{bmatrix}$$

State update

- Project a structure line onto the image



$$\mathbf{l} = [c_a, c_b, \theta, h]^T$$

World frame

$$\mathbf{l}^w h = \mathbf{P}^T \left([\ c_a, c_b\ ]^T h + [\ \cos(\theta), \sin(\theta)\ ]^T \right)$$

Camera frame

$$\mathbf{l}^c = \mathbf{R}^{cw}\mathbf{l}^w h - \mathbf{R}^{cw}\mathbf{p}^w h$$

Image

$$\mathbf{l}^i = \mathbf{K}\mathbf{l}^c$$

$$\bar{\mathbf{l}} = \mathbf{v} \times \mathbf{l}^i$$

Connect with vanishing point

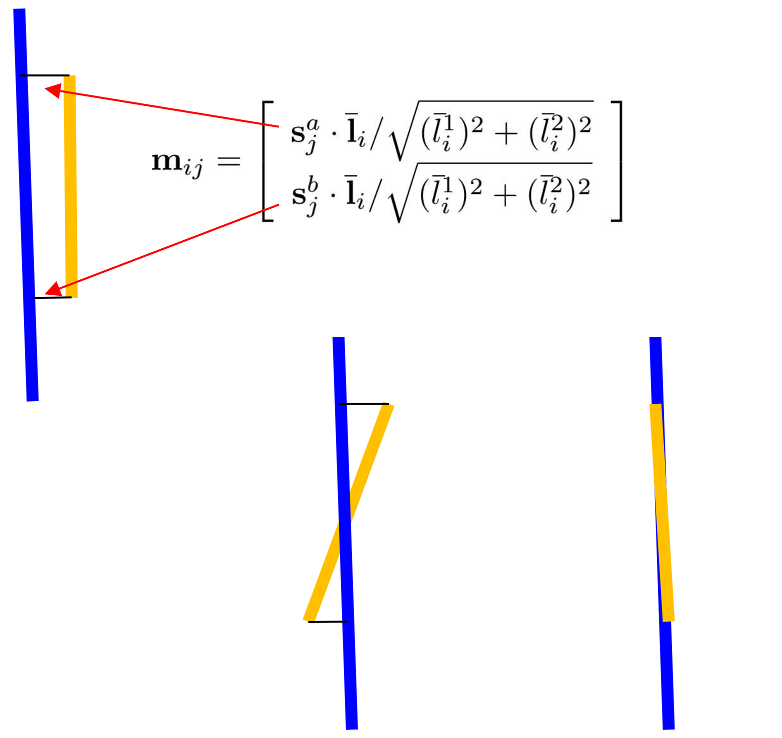
State update

- Observation model
  - Observation function (measurement function):

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \vdots \\ \mathbf{m}_{ij} \\ \vdots \end{bmatrix}$$

  - Since the desired distance is zero, the residual is computed as : $\mathbf{r}(\mathbf{x}) = -\mathbf{h}(\mathbf{x})$

$$\mathbf{m}_{ij} = \begin{bmatrix} \mathbf{s}_j^a \cdot \bar{\mathbf{l}}_i / \sqrt{(\bar{l}_i^1)^2 + (\bar{l}_i^2)^2} \\ \mathbf{s}_j^b \cdot \bar{\mathbf{l}}_i / \sqrt{(\bar{l}_i^1)^2 + (\bar{l}_i^2)^2} \end{bmatrix}$$



State update

- Standard Extended Kalman Filter:

Predicted state and covariance: $\bar{\mathbf{x}}, \bar{\boldsymbol{\Sigma}}$

$$\begin{aligned} \text{Innovation covariance:} \quad & \mathbf{S} = \mathbf{H}\bar{\boldsymbol{\Sigma}}\mathbf{H}^T + \mathbf{N} \\ \text{Kalman gain:} \quad & \mathbf{K} = \bar{\boldsymbol{\Sigma}}\mathbf{H}^T\mathbf{S}^{-1} \end{aligned}$$

$$\begin{aligned} \text{State update:} \quad & \mathbf{x} \leftarrow \bar{\mathbf{x}} + \mathbf{K}\mathbf{r} \\ \text{Covariance update:} \quad & \boldsymbol{\Sigma} \leftarrow \bar{\boldsymbol{\Sigma}} - \mathbf{K}\mathbf{S}\mathbf{K}^T \end{aligned}$$

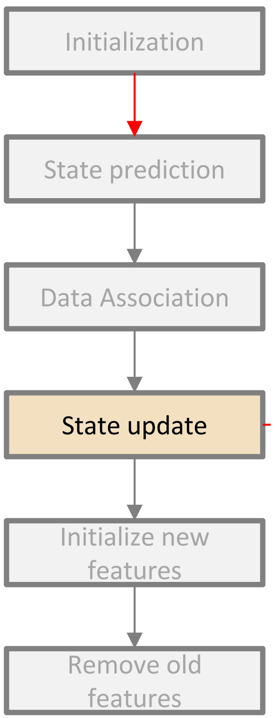
$$\mathbf{x}, \boldsymbol{\Sigma}$$

$\mathbf{H} = \frac{\partial \mathbf{m}}{\partial \mathbf{x}}$:Jacobian of observation function

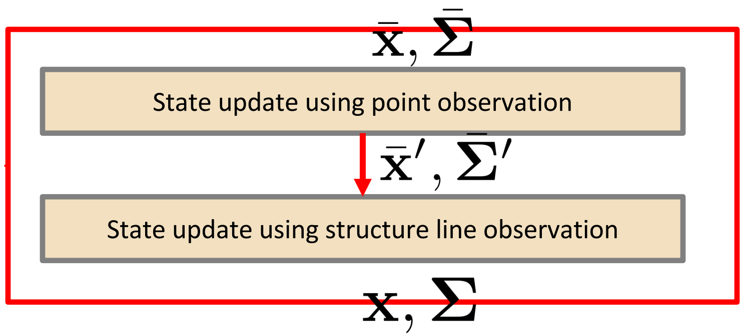
N : Observation noise - Uncertainty of detected line segments

$$\begin{bmatrix} \ddots & & & \\ & 4 & & \\ & & 4 & \\ & & & \ddots \end{bmatrix}$$

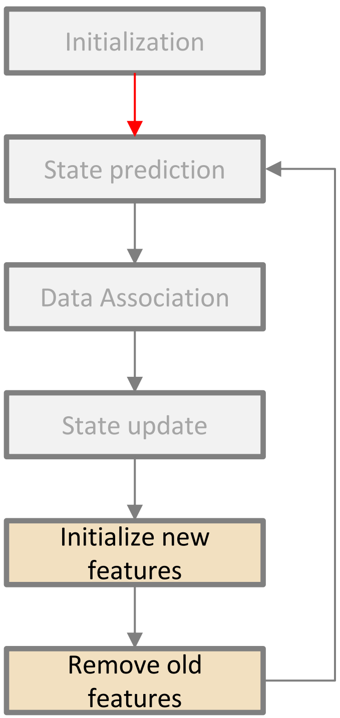
State update



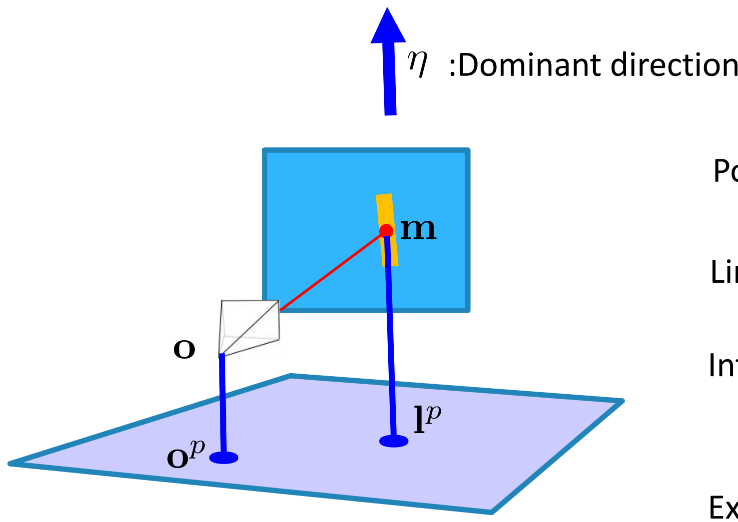
- Multiple-pass EKF update (Point + Structure lines)



Feature management



- Initialize new structure lines



Point in world frame: $\mathbf{m} = \mathbf{R}^{wc}\mathbf{K}^{-1}\tilde{\mathbf{m}} + \mathbf{p}^w$

Line though the point: $\mathbf{L} = \mathbf{m}\eta^T - \eta\mathbf{m}^T$

Intersection with parameter plane:

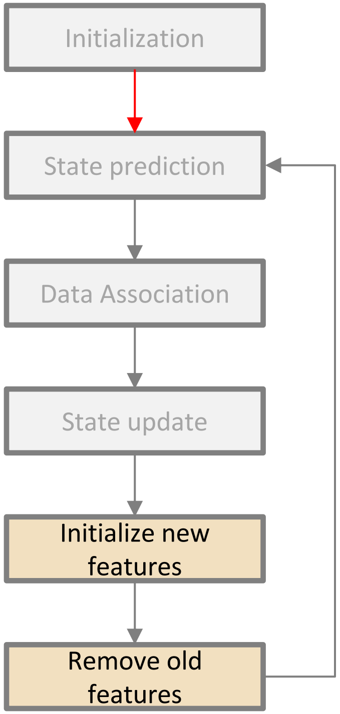
$$\tilde{\mathbf{l}}^w = \mathbf{L}\pi$$

Expressed in parameter plane (xy-plane):

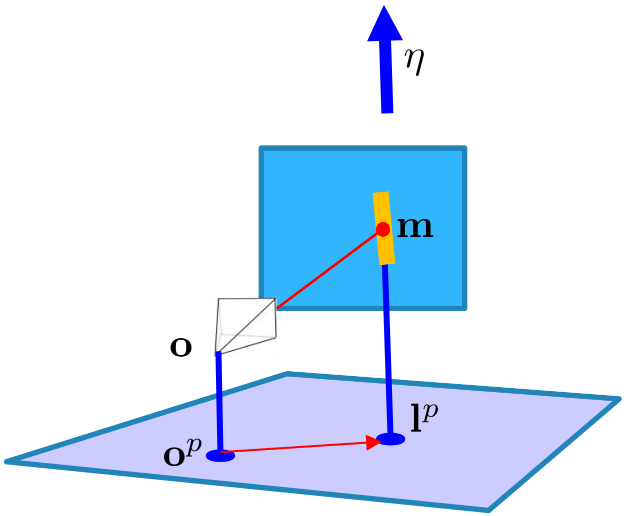
$$\mathbf{l}^p = \mathbf{P}\mathbf{l}^w$$

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

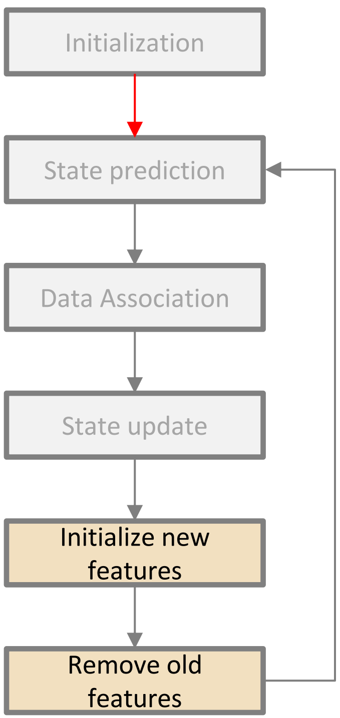
Feature management



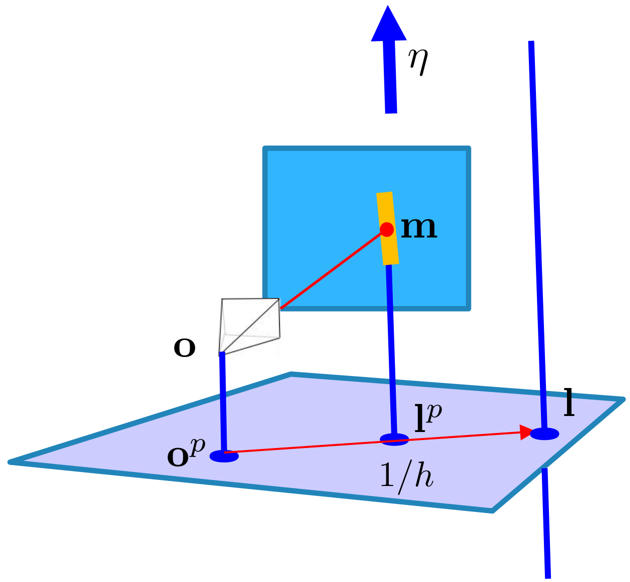
- Initialize new structure lines



Feature management

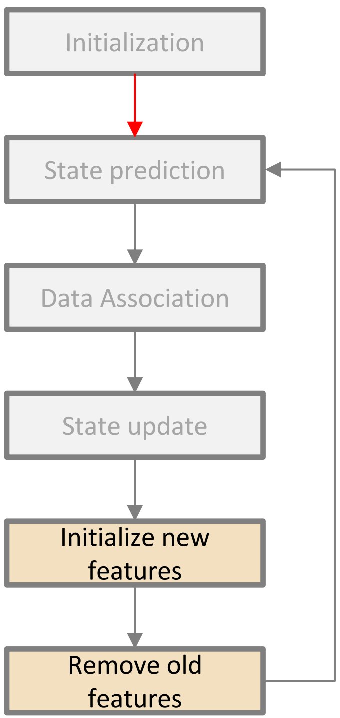


- Initialize new structure lines



$$\begin{aligned}\mathbf{l} &= \left[\ c_a, c_b, \theta, h\ \right]^T \\ &= \left[\ \mathbf{o}^p(1), \mathbf{o}^p(2), atan\left(\frac{\mathbf{l}^p(2)-\mathbf{o}^p(2)}{\mathbf{l}^p(1)-\mathbf{o}^p(1)}\right), h_0\ \right]^T\end{aligned}$$

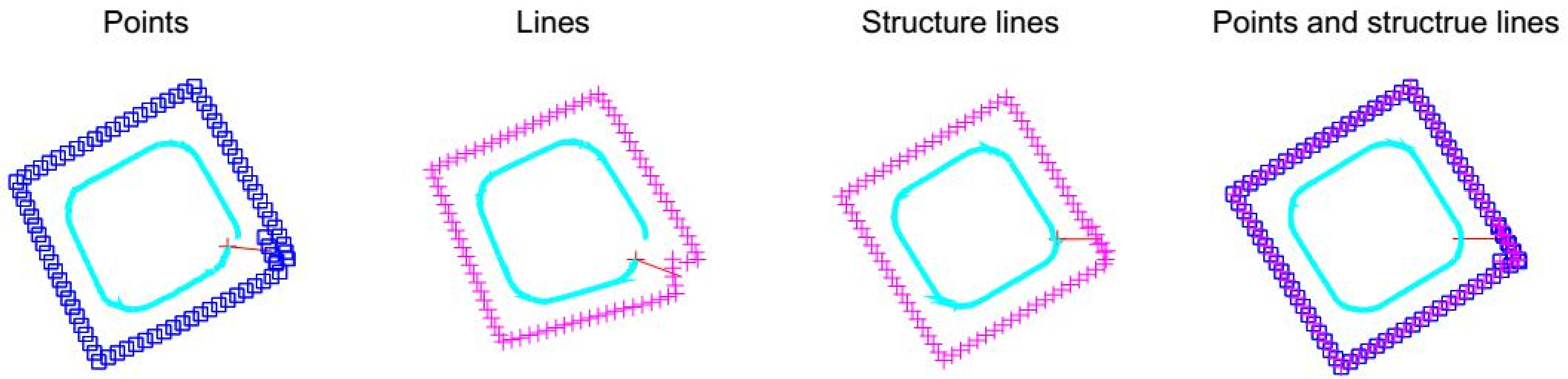
Feature management



- The number of features is limited in the state.
- For each dominant direction, we keep a maximum number of structure lines.
- Old features are removed according to the number of matching failure (NOF)

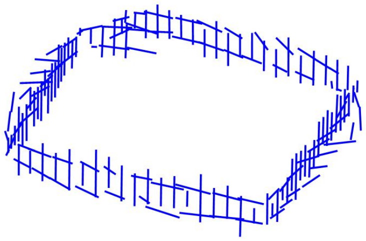
Results

- Simulated case

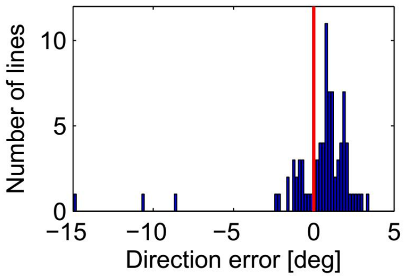


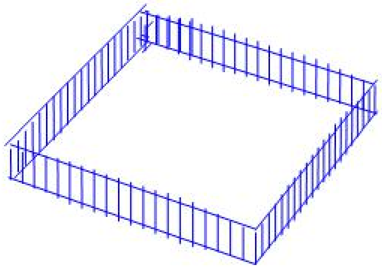
Results

- Simulated case



Lines



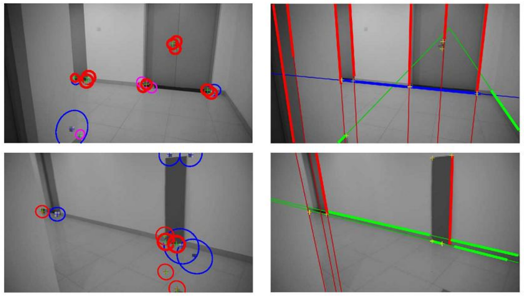


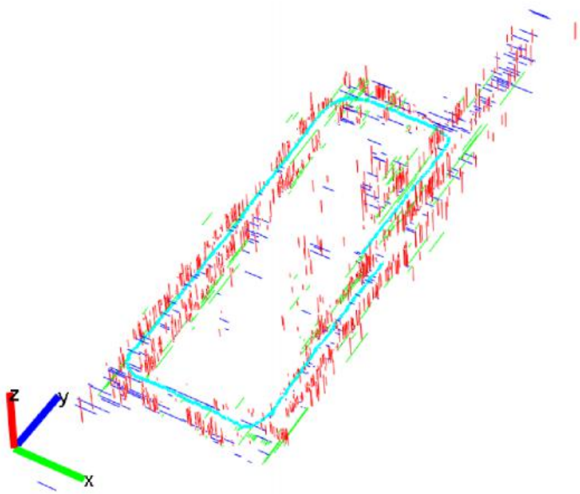
Structure lines

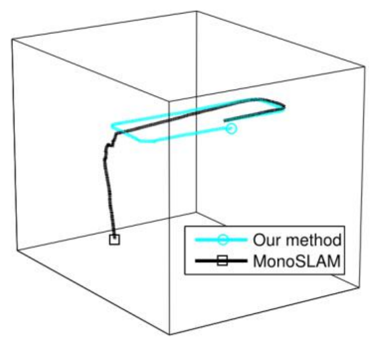
Lines V.S. structure lines

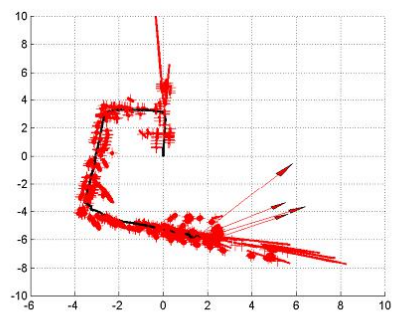
Results

- Real-world case (Using one video camera)
  - Indoor texture-less scenes



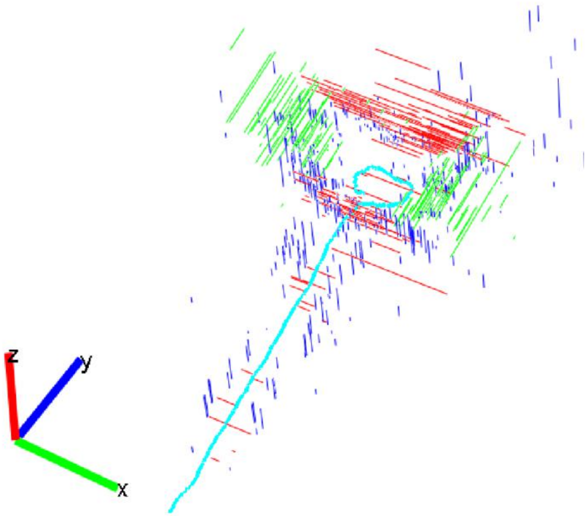


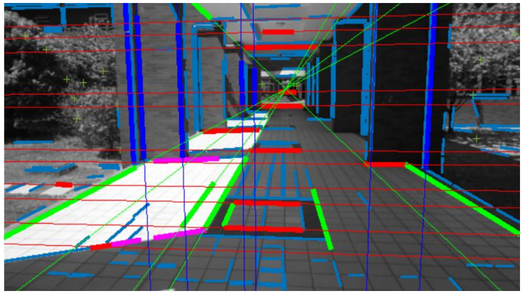


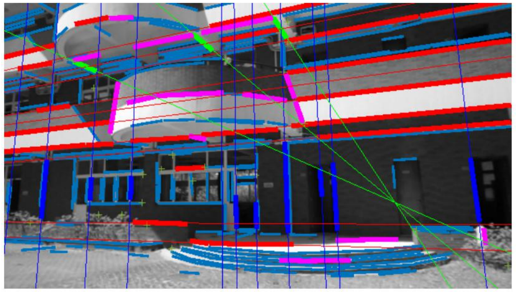


Results of real-word indoor scenes

- Real world case (Using one video camera)
  - Outdoor texture rich scenes

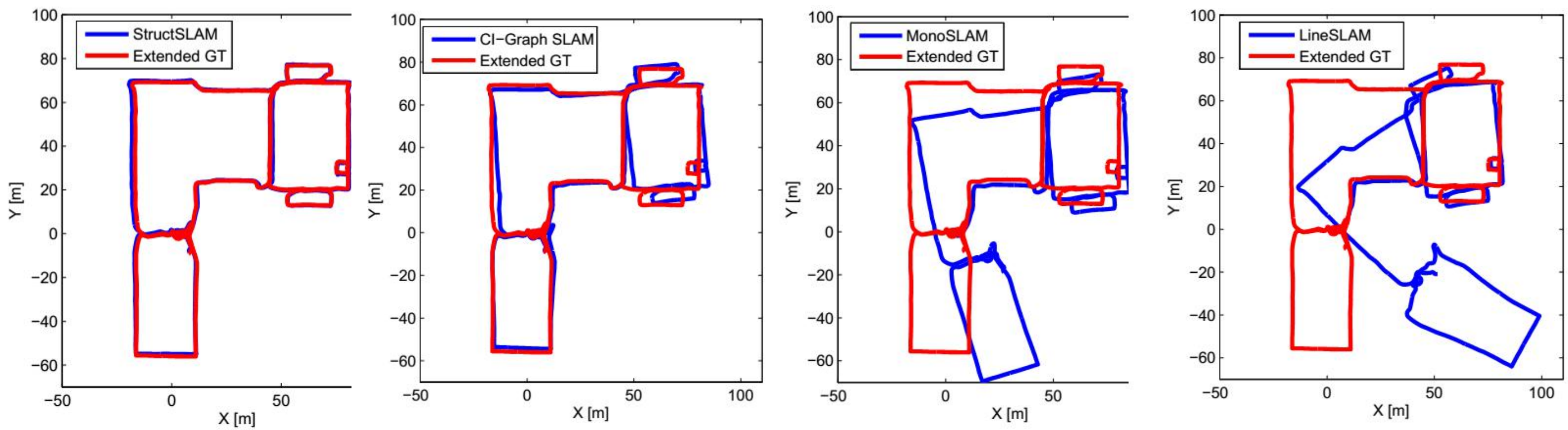






Benchmark datasets

- Rawseeds datasets (all methods fused the odometer information)
  - **Bicacco-02-25b** : A 774m trajectory in the indoor scene



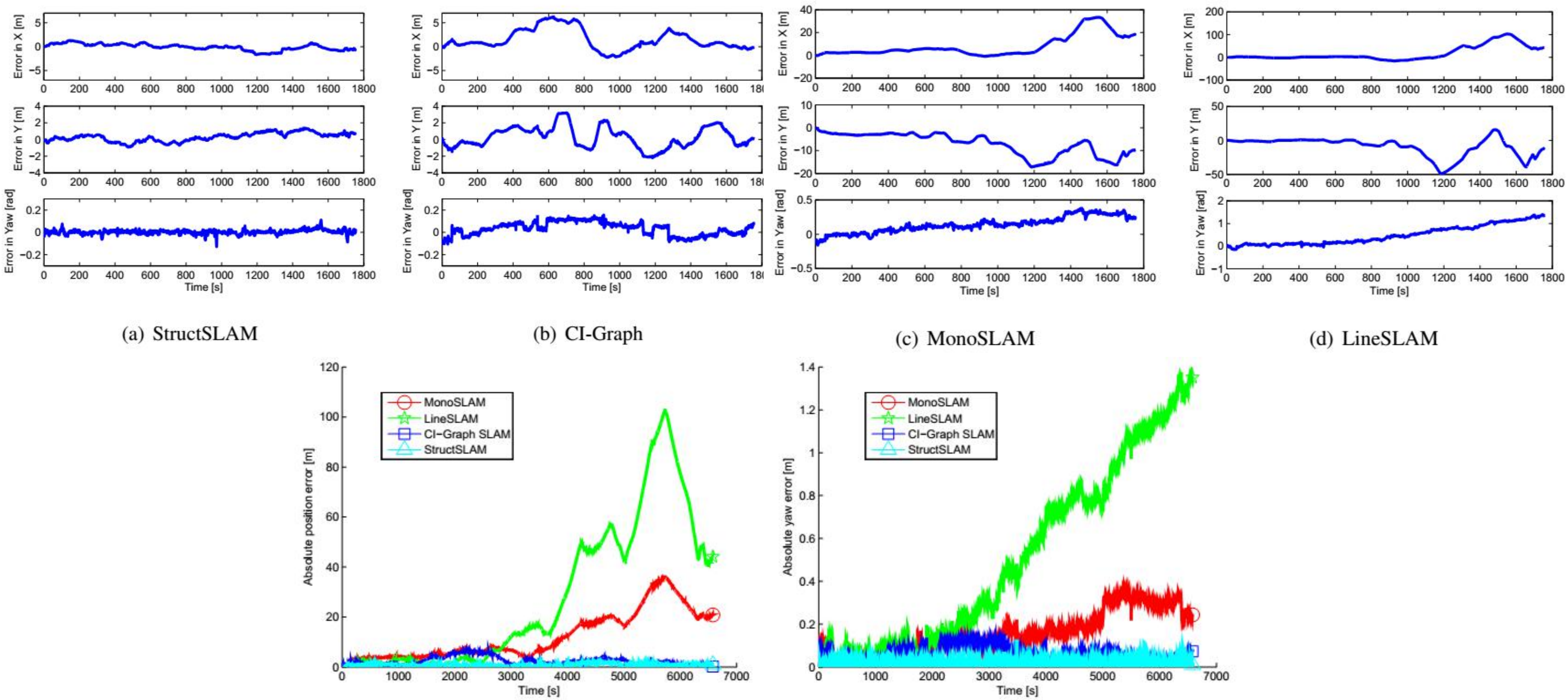
(a) StructSLAM vs GT

(b) CI-Graph vs GT

(c) MonoSLAM vs GT

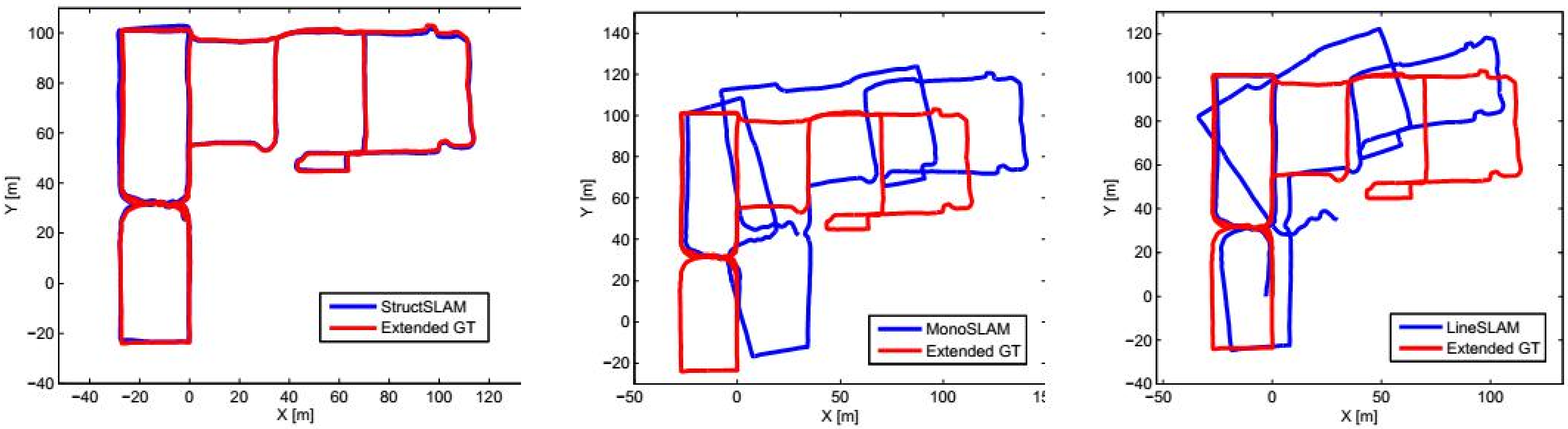
(d) LineSLAM vs GT

Benchmark results



Benchmark datasets

- **Bicacco-02-27a :**



(a) StructSLAM (b) MonoSLAM (c) LineSLAM

Benchmark datasets

- Comparision

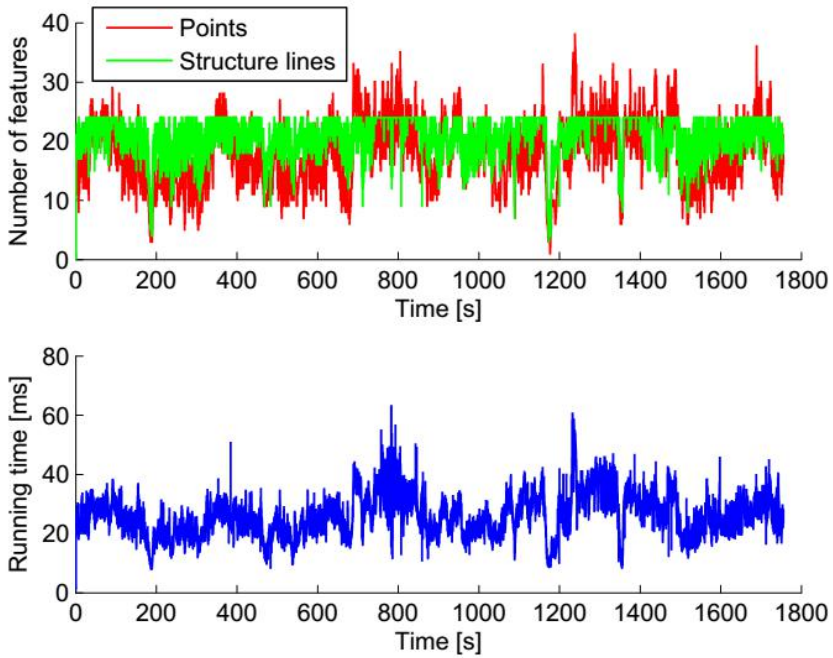
	Biccoca 25b [774 m]						Biccoca 27a [967m]					
	Position[m]			Yaw[rad]			Position[m]			Yaw[rad]		
	Mean	Max	Std	Mean	Max	Std	Mean	Max	Std	Mean	Max	Std
MonoSLAM	12.22	36.12	9.469	0.154	0.371	0.102	24.85	37.90	11.66	0.187	0.392	0.104
LineSLAM	27.63	102.7	29.65	0.509	1.393	0.420	12.77	32.35	10.17	0.208	0.706	0.206
CI-GraphSLAM	2.236	6.453	1.675	0.055	0.155	0.035	-	-	-	-	-	-
StructSLAM	**0.797**	**1.916**	**0.447**	**0.012**	**0.129**	**0.011**	**0.793**	**1.913**	**0.493**	**0.017**	**0.214**	**0.017**

TABLE II. ERROR COMPARISON.

StructSLAM : Without any loop-closing algorithms being applied!

Running time efficiency

- A common PC with i7 4-core cpu (2.7GHz) (c++ implementation)



Average running time: **25.8 ms**
Peak running time: **62.9 ms**

Conclusion & Discussion

- StructSLAM is more robust in texture-less indoor scenes than conventional SLAM methods
- With global orientation information encoded in structure lines, StructSLAM produces much less drift error.
- It is well fit for robotic and augmented reality (AR) applications in indoor scenes.

StructSLAM:

Visual SLAM with Building Structure Lines

Hui Zhong Zhou, Danping Zou et al.

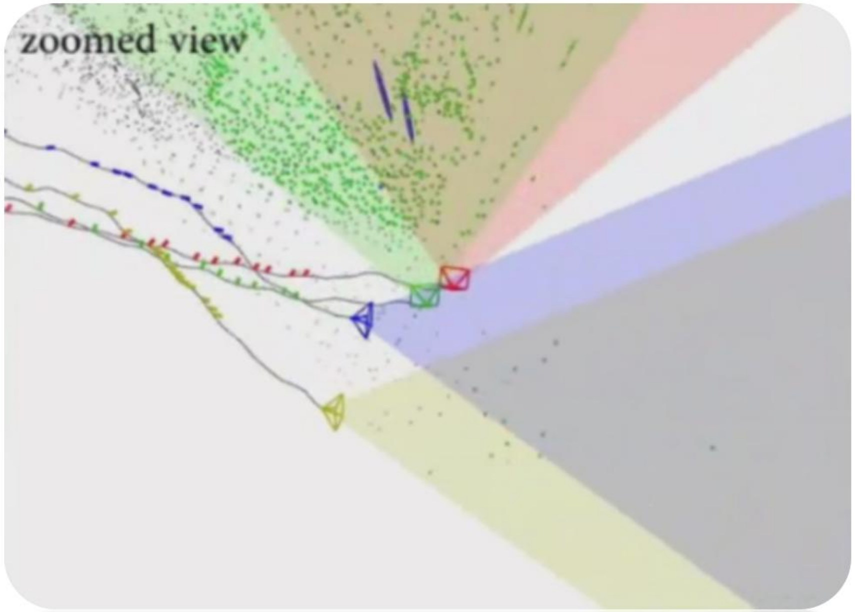
Shanghai Key Laboratory of Navigation and Location Based Services
Shanghai Jiao Tong University
Apirl,2014

Outline

- Basic Theory
  - Projective geometry
  - Pinhole camera model
  - Camera calibration
  - Two camera geometry

- Design a typical Visual SLAM system
- Two Visual SLAM systems:
  - Extended Kalman Filter approach:
    - StructSLAM
  - **Visual SLAM for a group of robots:**
    - **CoLSAM**

CoSLAM



https://github.com/danping/CoSLAM

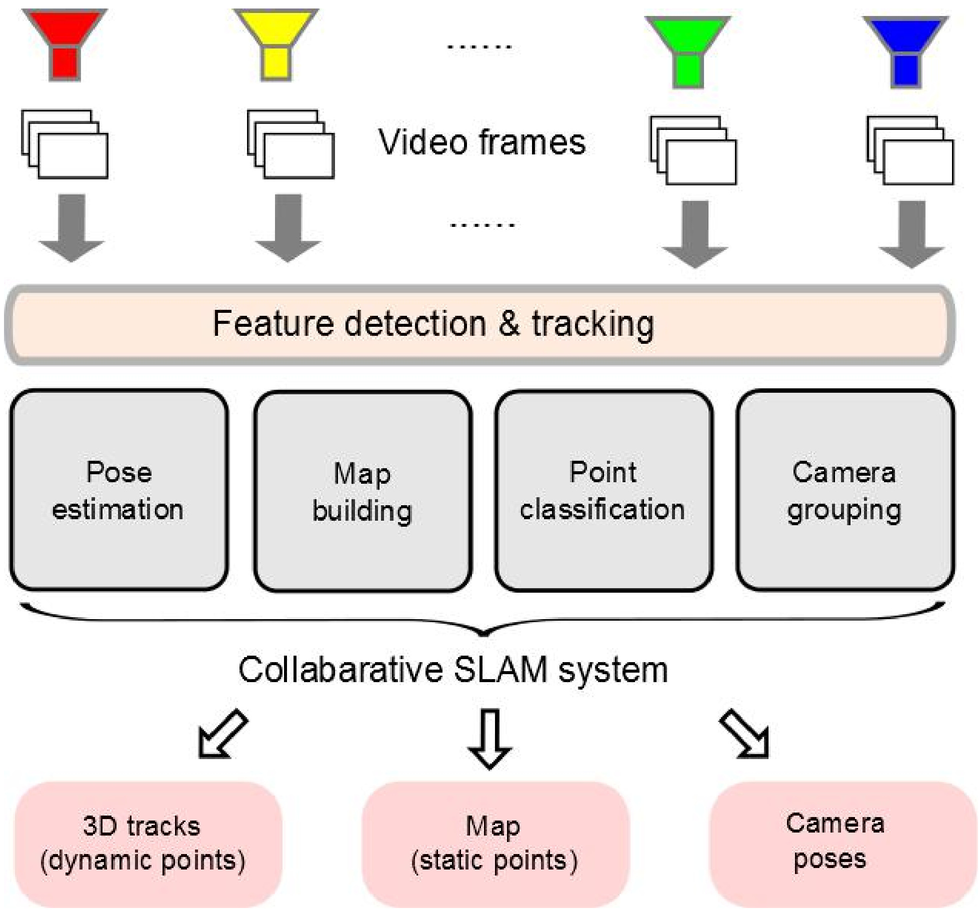
Zou D, Tan P. **CoSLAM**: Collaborative visual slam in dynamic environments[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 35(2): 354-366.

Team of drones

SLAM with multiple freely moving cameras

1. **Efficiency** – Distributed exploration
2. **Robustness** - Wider view angle
3. **Bonus** - Reconstruct moving points

CoSLAM: Collaborative Visual SLAM



Incremental SFM approach

Issues need to be addressed

- **Pose estimation**
- **Point management**
- **Mapping**
- **Grouping**

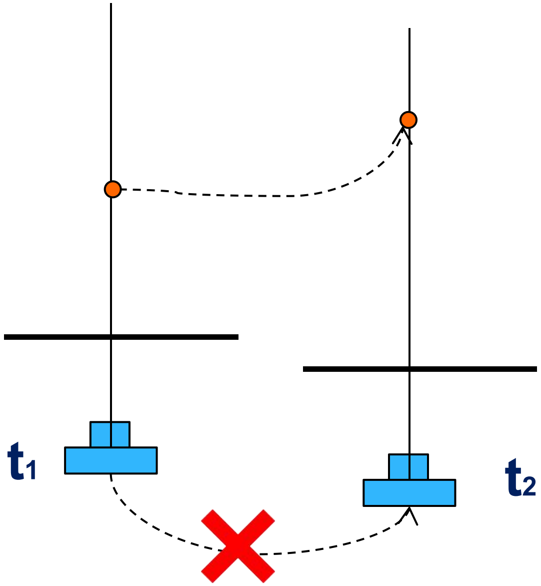
Pose estimation

Pose estimation

Single view pose estimation

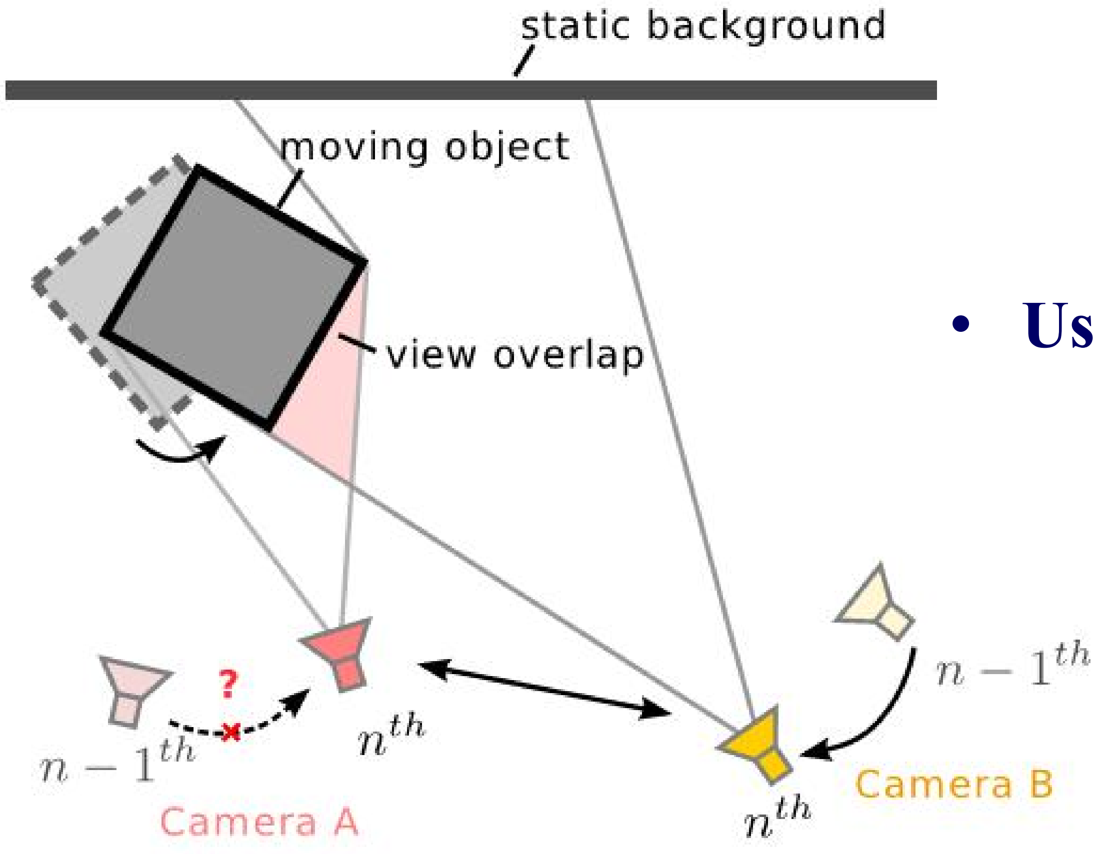
Intra-camera pose estimation

Inter-camera pose estimation



Pose estimation

- Inter-camera pose estimation



- **Use moving objects as a reference**

Pose estimation

- Inter-camera pose estimation

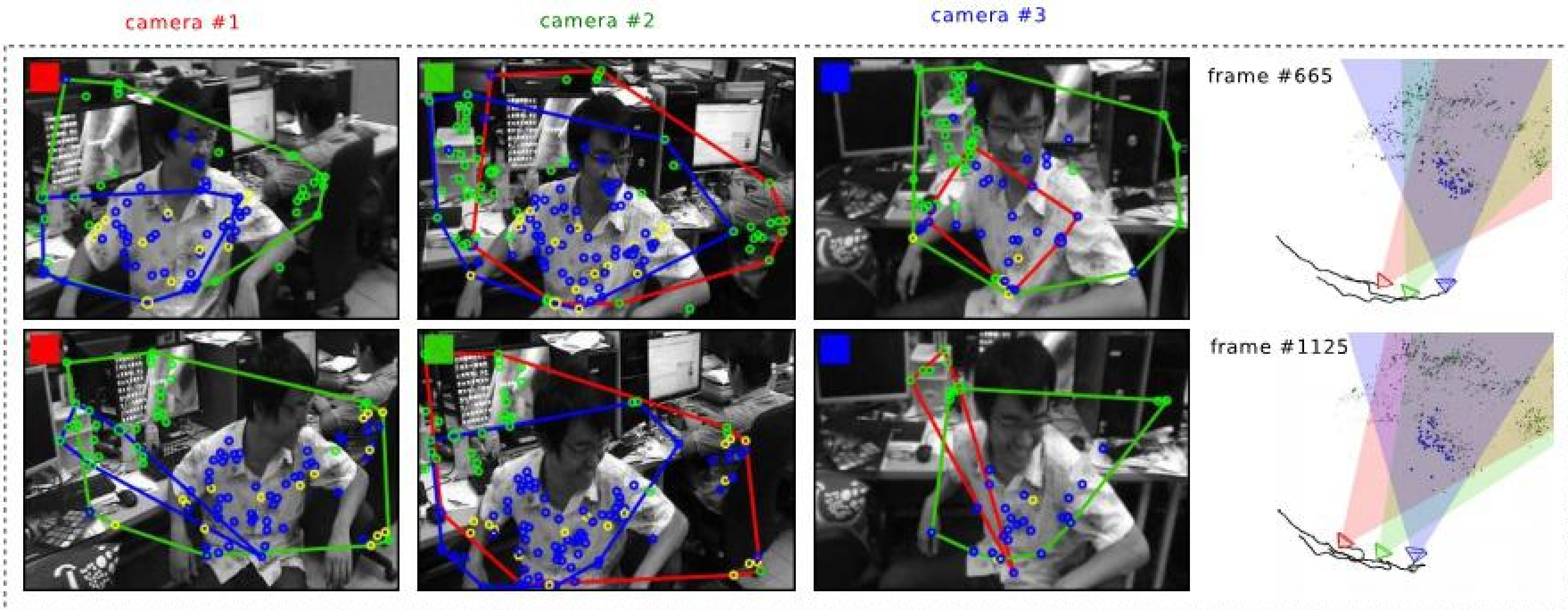
$$\{\Theta_c\}^* = \underset{\mathbf{M}_D, \{\Theta_c\}}{\arg\min} \sum_c \Bigg\{ \sum_{i \in S} v_i^c \underbrace{\boxed{\rho\left(\|\mathbf{m}_i - \mathcal{P}(\mathbf{M}_i, \Theta_c)\|\right)}}_{\text{Reprojection error of static points}} + \sum_{j \in D} v_j^c \underbrace{\boxed{\rho\left(\|\mathbf{m}_j - \mathcal{P}(\mathbf{M}_j, \Theta_c)\|\right)}}_{\text{Reprojection error of dynamic points}} \Bigg\}.$$

Θ_c : **Pose of camera C**

$\mathbf{M}_D$: **3D coordinates of the dynamic points**

- **Use both static and moving points to estimate camera poses**
- **3D coordinates of dynamic points are updated**

Intra-camera pose estimation fails



Inter-camera pose estimation succeeds in highly dynamic sce

Point management

- **Four types of points**
  - **Static** - 3D points in static backgrounds
  - **Dynamic** – 3D points on moving objects
  - **False** – False stereo matching
  - **Uncertain** - Intermediate state for further investigation

Point classification

Intra-camera outlier (Time outlier) :

reprojection error is large at some frame

Inter-camera outlier (Spatial outlier):

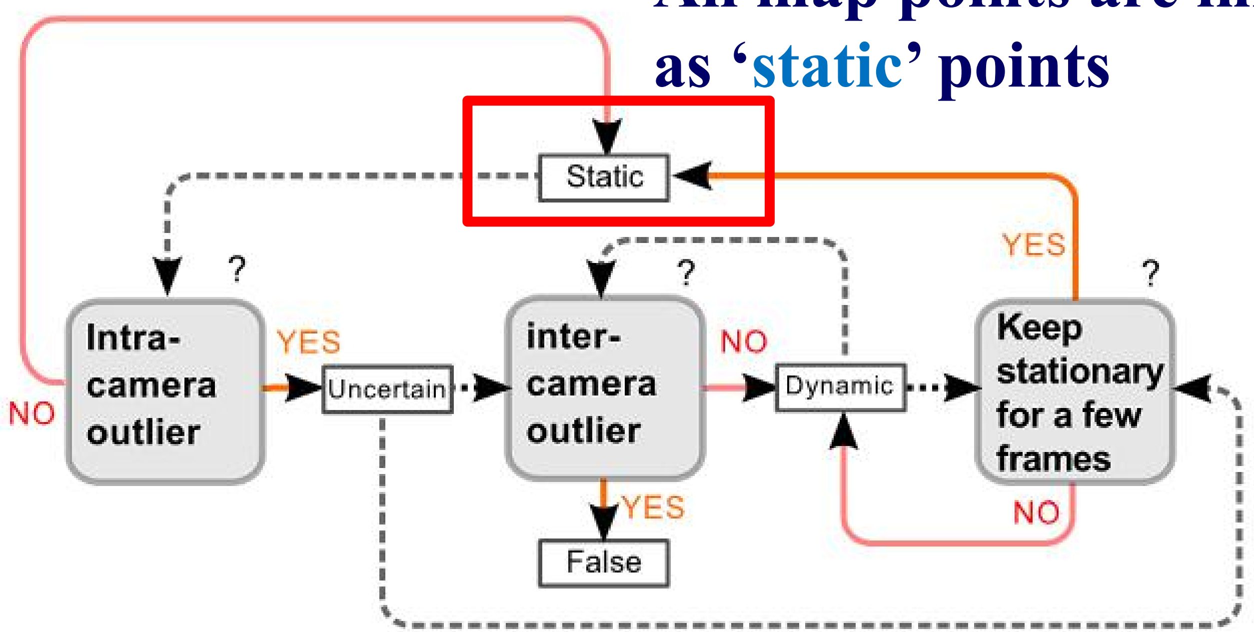
reprojection error is large in some camera

Point type	Intra-camera outlier	Inter-camera outlier
Static points	No	No
Dynamic points	Yes	No
False points	Yes	Yes

Map point classification

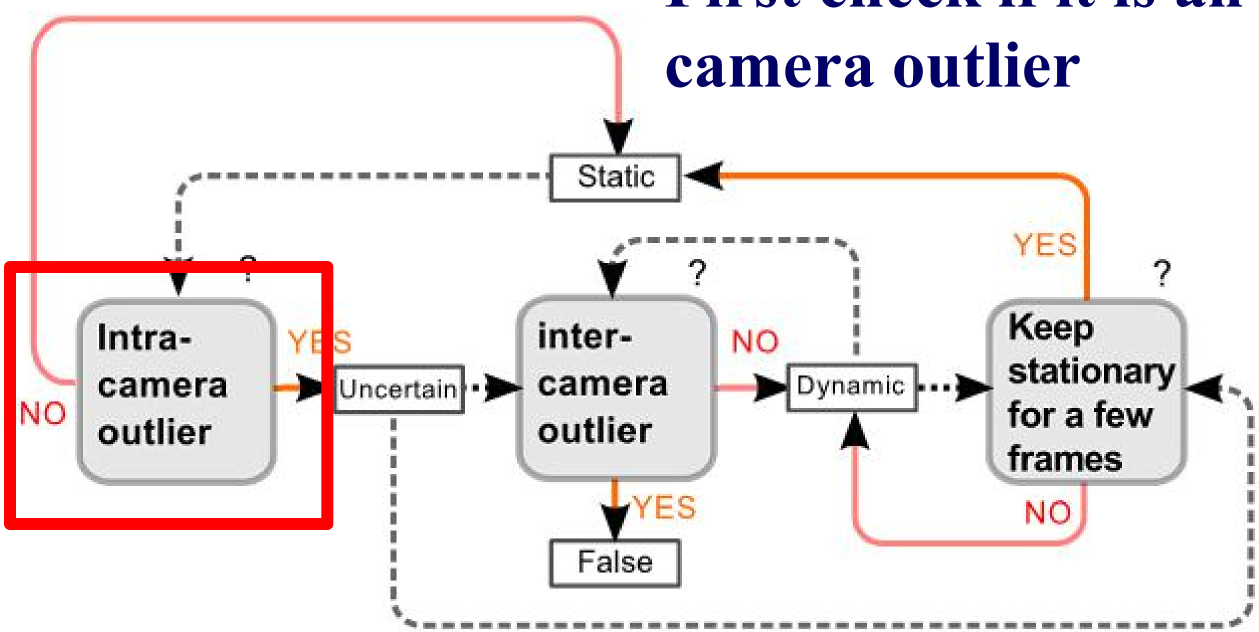
Pipeline for point classification:

All map points are initialized as 'static' points



Map point classification

Pipeline for point classification:

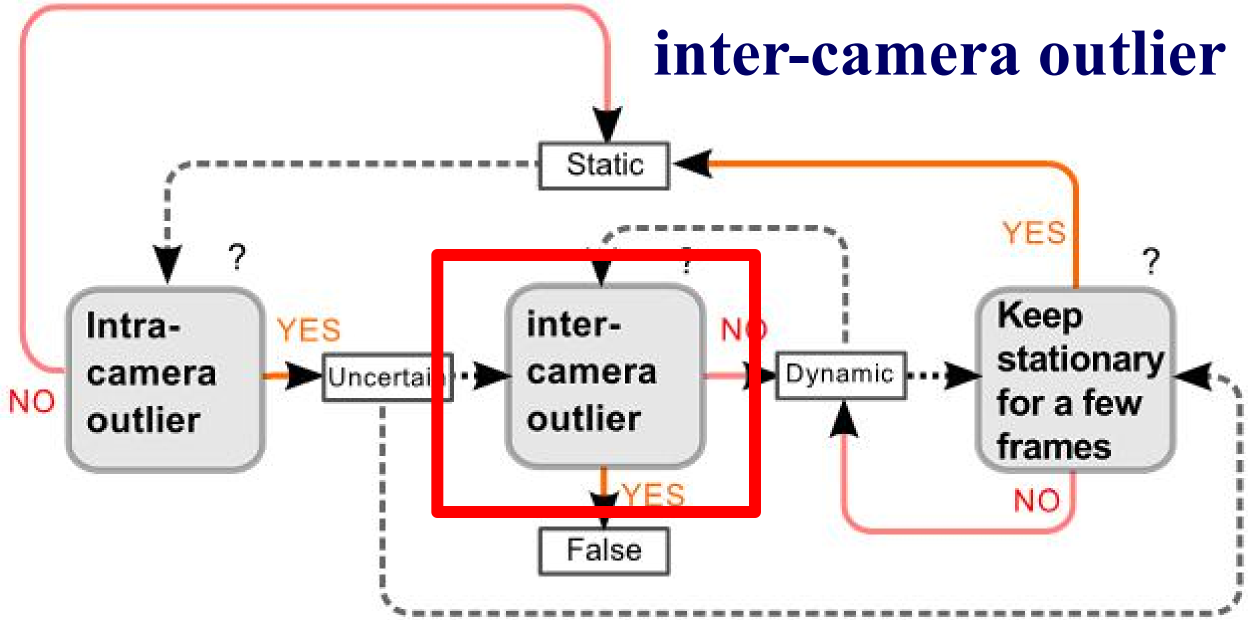


First check if it is an intra-camera outlier

Map point classification

Pipeline for point classification:

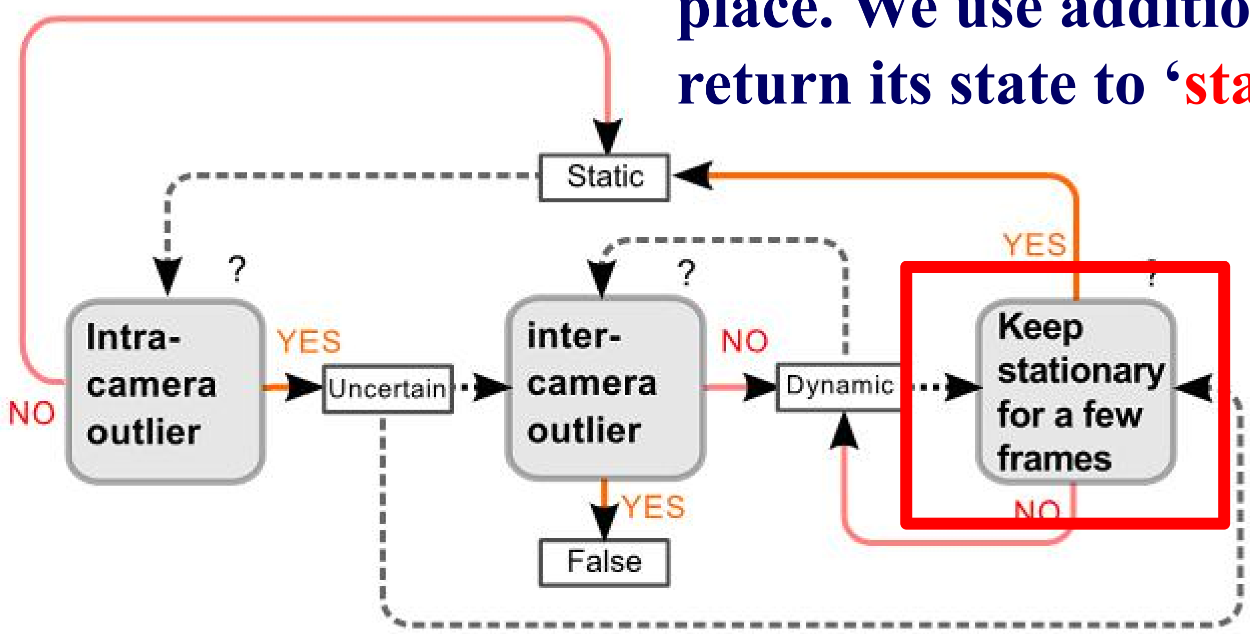
Further check if it is an inter-camera outlier

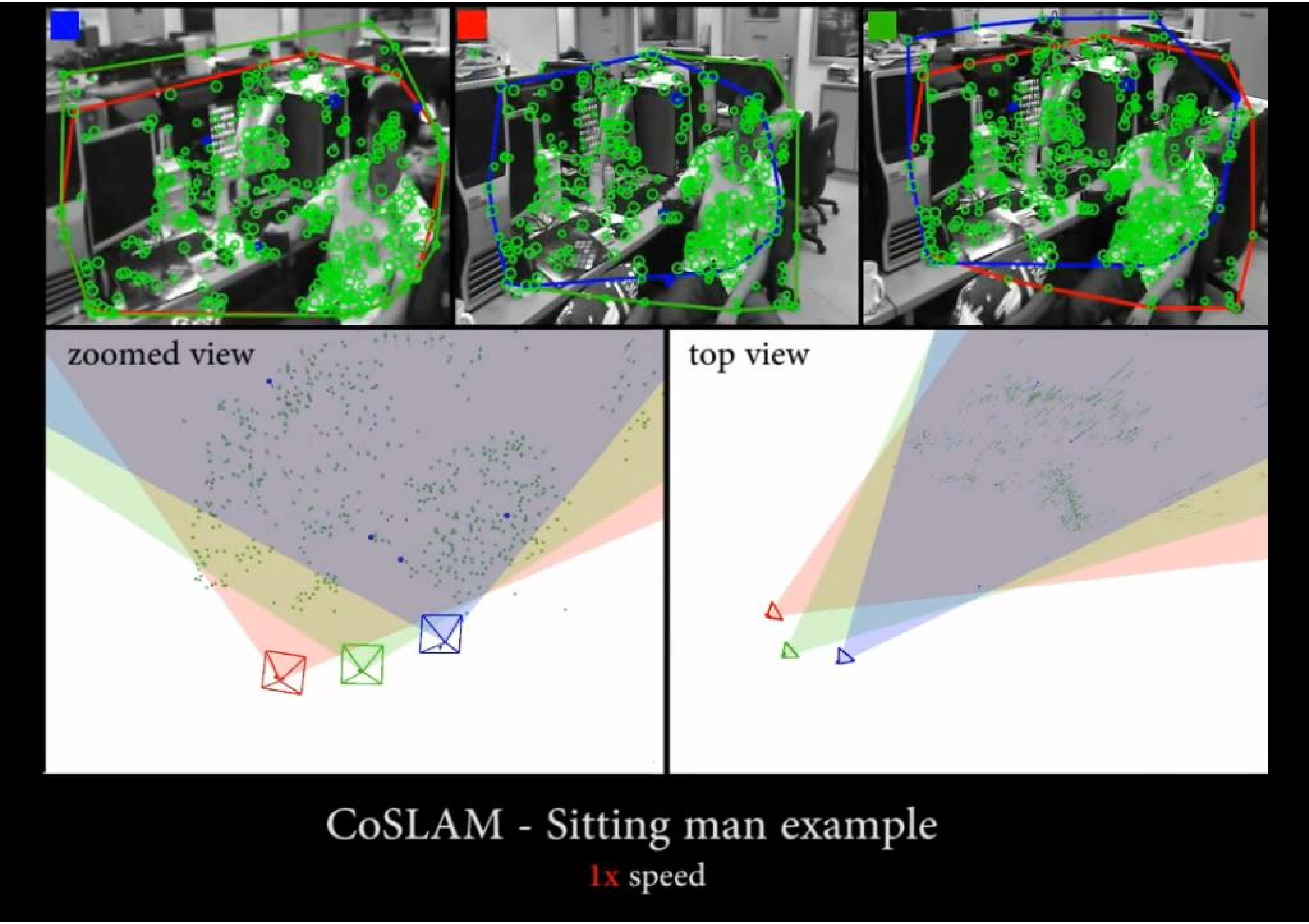


Map point classification

Pipeline for point classification:

Moving objects may stop at some place. We use additional step to return its state to 'static'




zoomed view
top view
CoSLAM - Sitting man example
1x speed

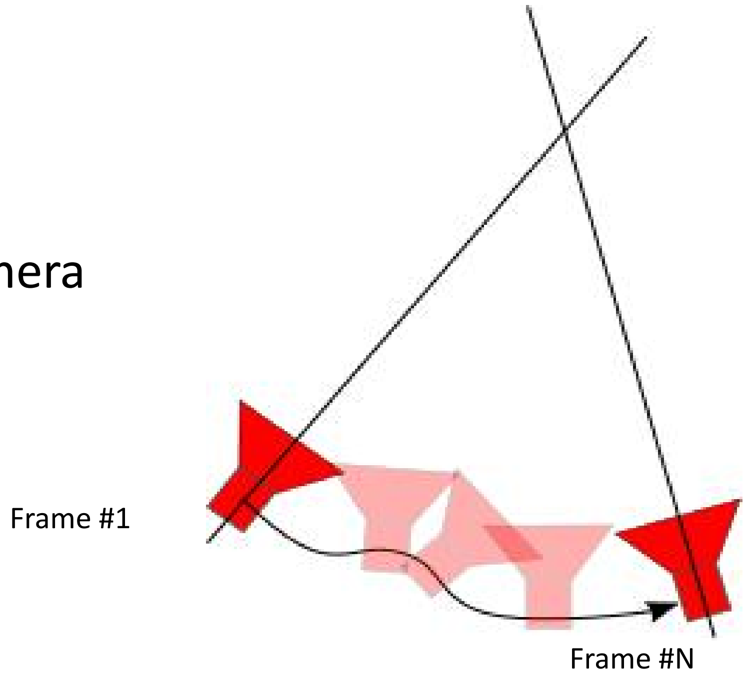
Mapping

Mapping

- **Intra-camera mapping**
- **Inter-camera mapping**

Intra-camera mapping :

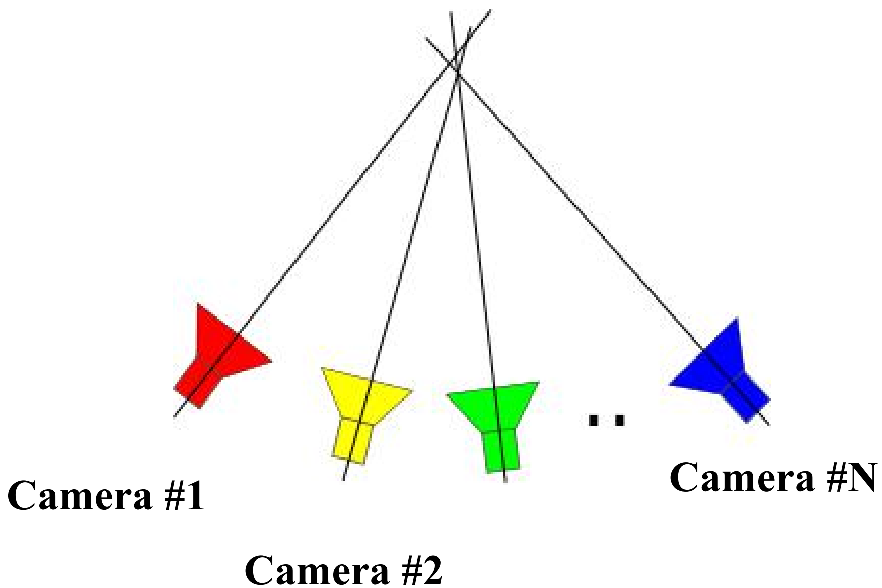
Triangulation is done in the same camera



Mapping

Inter-camera mapping:

✓ reconstruct map points from different cameras



Exhaustive matching:

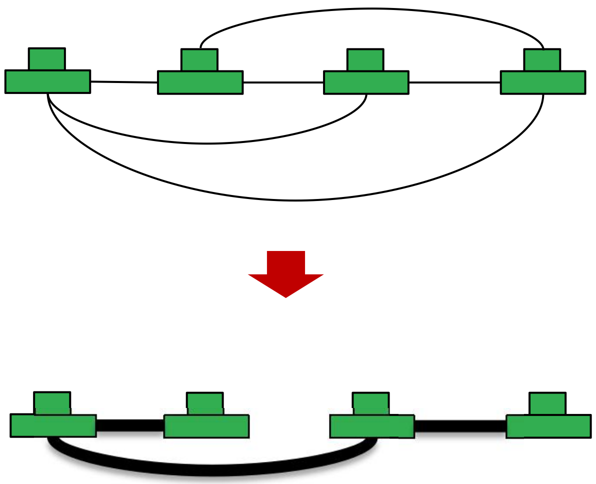
$O(N^2)$

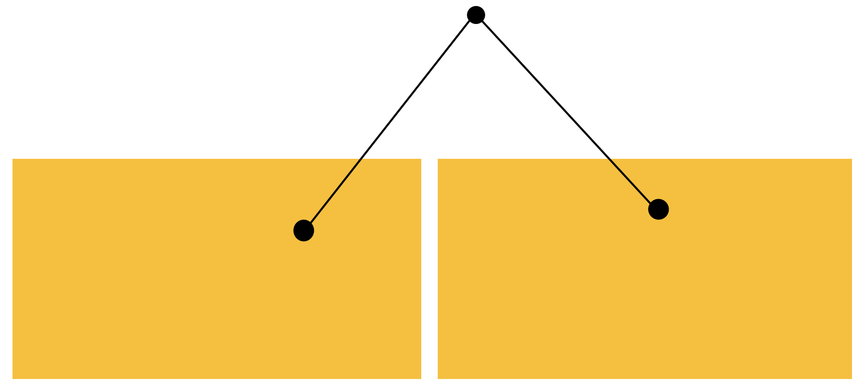
#Camera	Time for feature matching
2	200ms
3	1.2s
4	2.4s
5	4.0s

Mapping

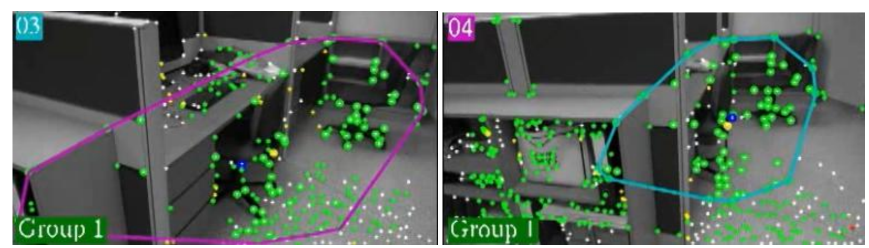
Inter-camera mapping:

✓ minimum spanning tree





Weight = Number of shared feature points

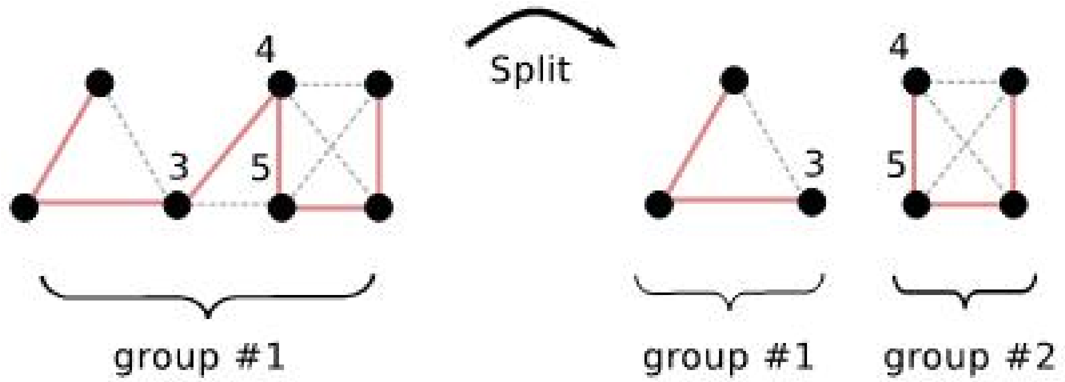


Grouping

- **A Group** – cameras with similar views

Grouping

- Use a graph to connect all cameras:



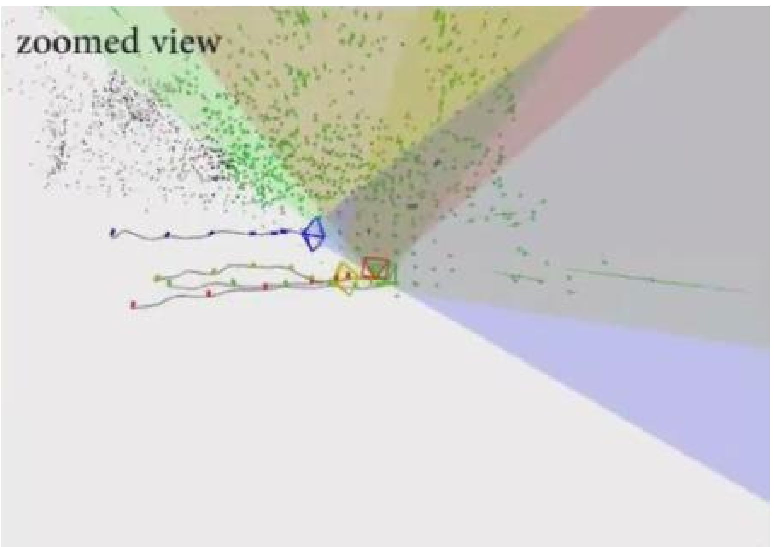
The red lines indicate the maximum spanning tree

A group splits when the weight of the edge on the spanning tree drop to zero.

Grouping

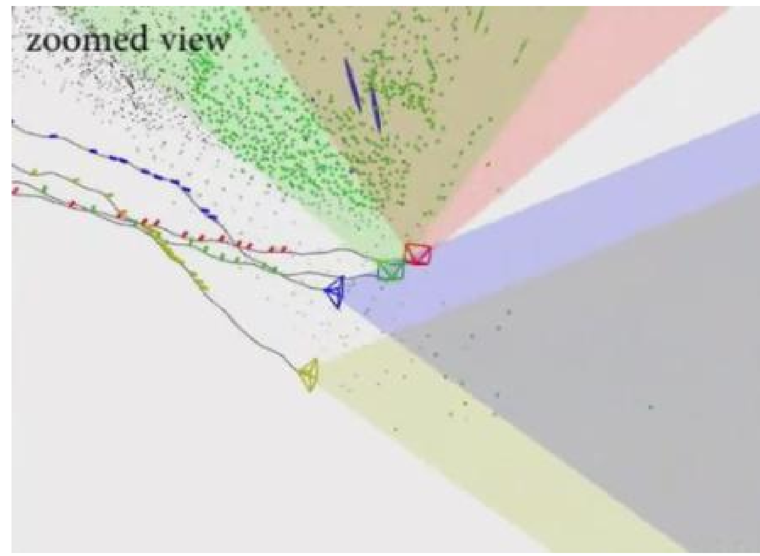
- **Example** - A group splits

Group #0: 1 3 2 4



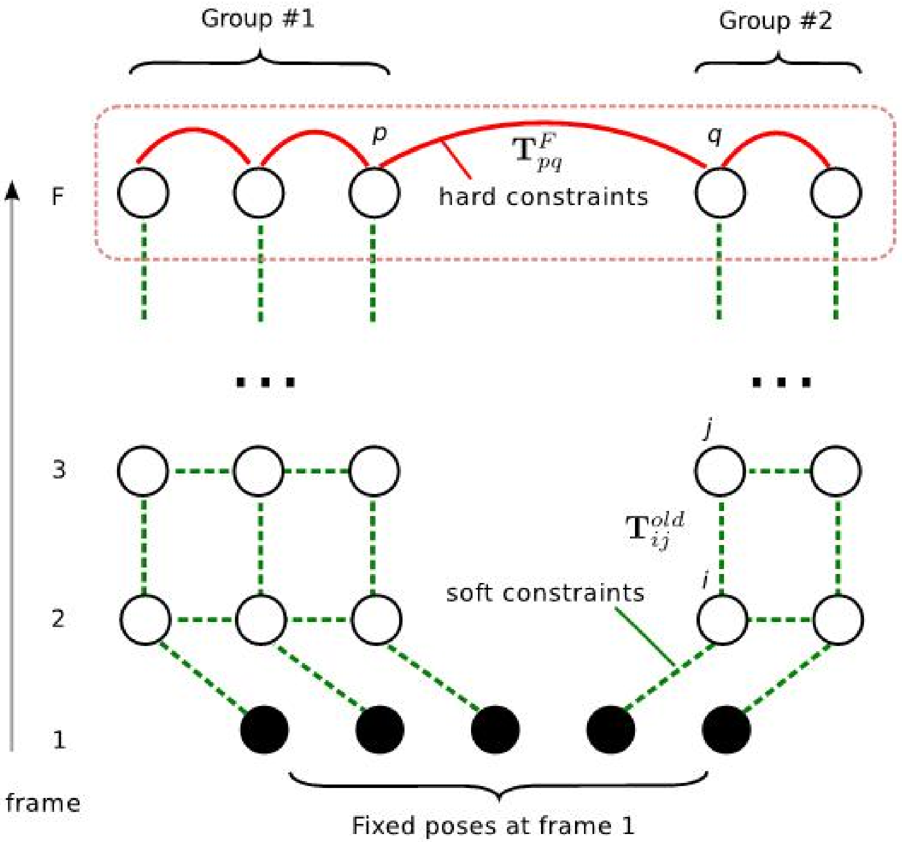
split

Group #0: 1 3 Group #1: 2 4



Grouping

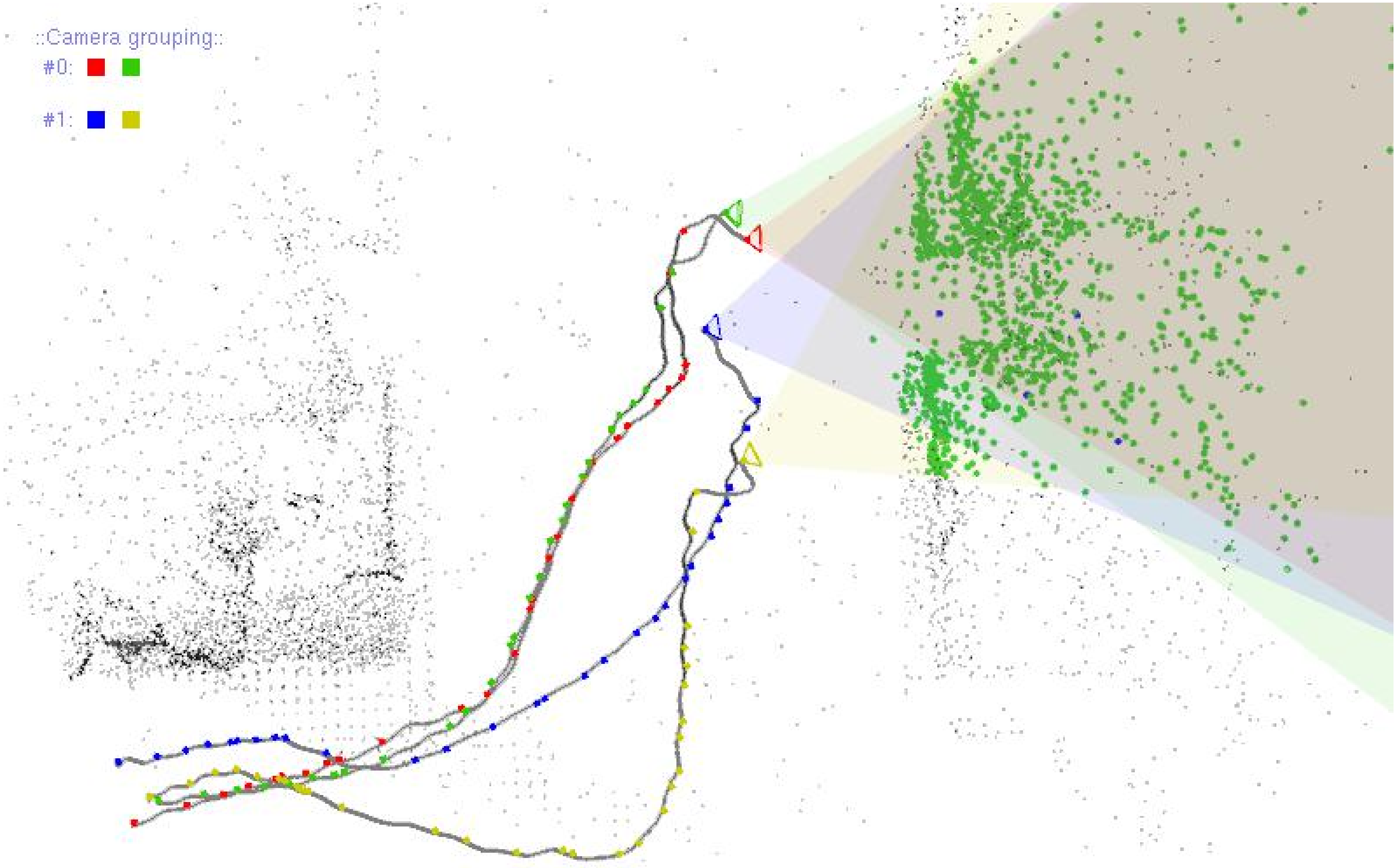
- **Group merging** – similar to loop closing
  - Step1 - Detect the loop
  - Step2 - Pose adjustment
  - Step3 - Retriangulation / Point merging



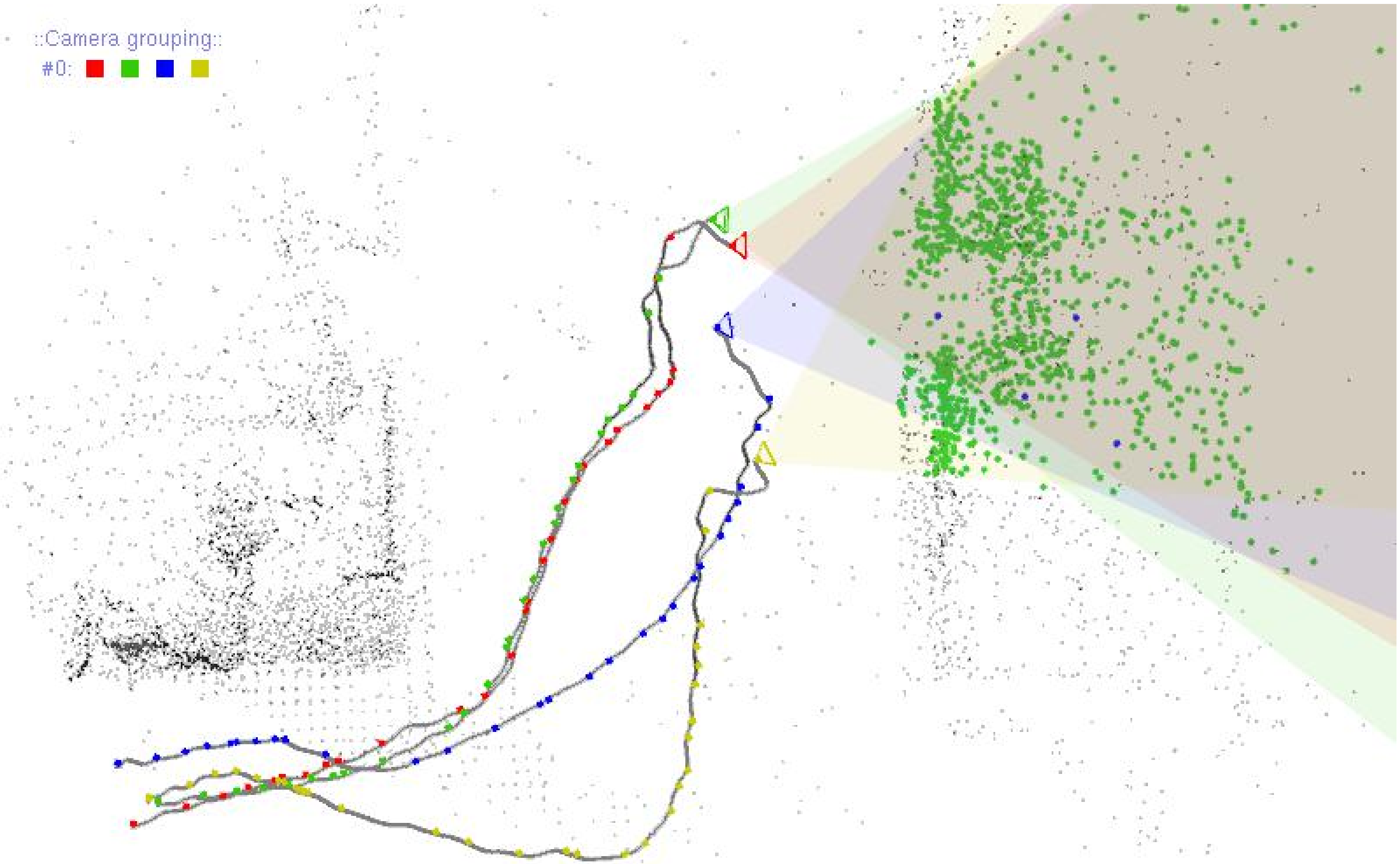
$$\mathbf{T}_m - \mathbf{T}_{mn}^{old}\mathbf{T}_n = \mathbf{0}_{4\times 4}$$

$$s.t.\ \mathbf{T}_q^F - \mathbf{T}_{pq}^F\mathbf{T}_p^F = \mathbf{0}_{4\times 4}$$

$$\Rightarrow \quad \mathbf{T}_1\mathbf{T}_2\cdots\mathbf{T}_m$$



Camera trajectories and map points before inconsistency removal



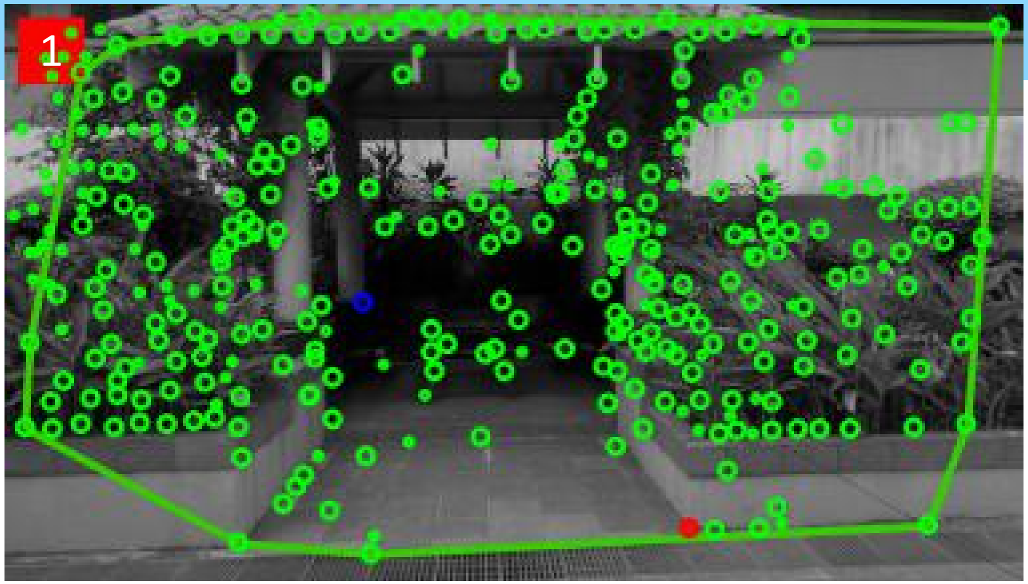
Camera trajectories and map points after inconsistency removal

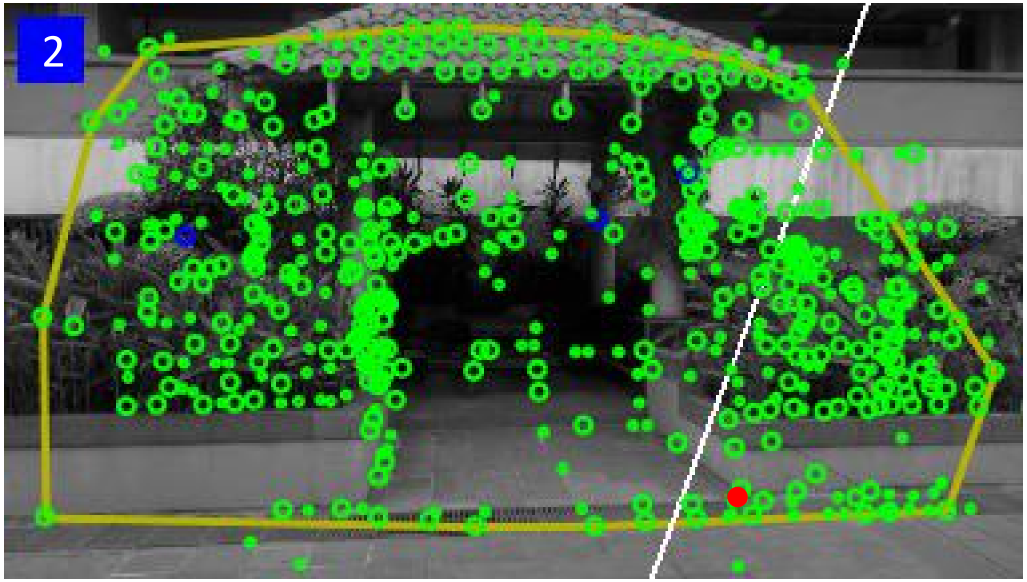
Group #0:

1 3

Group #1:

2 4

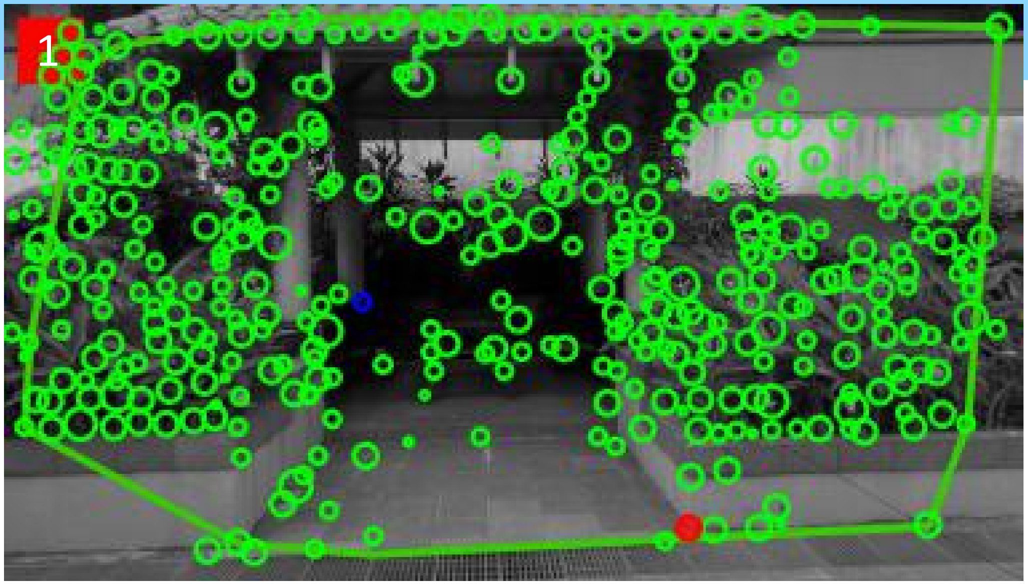


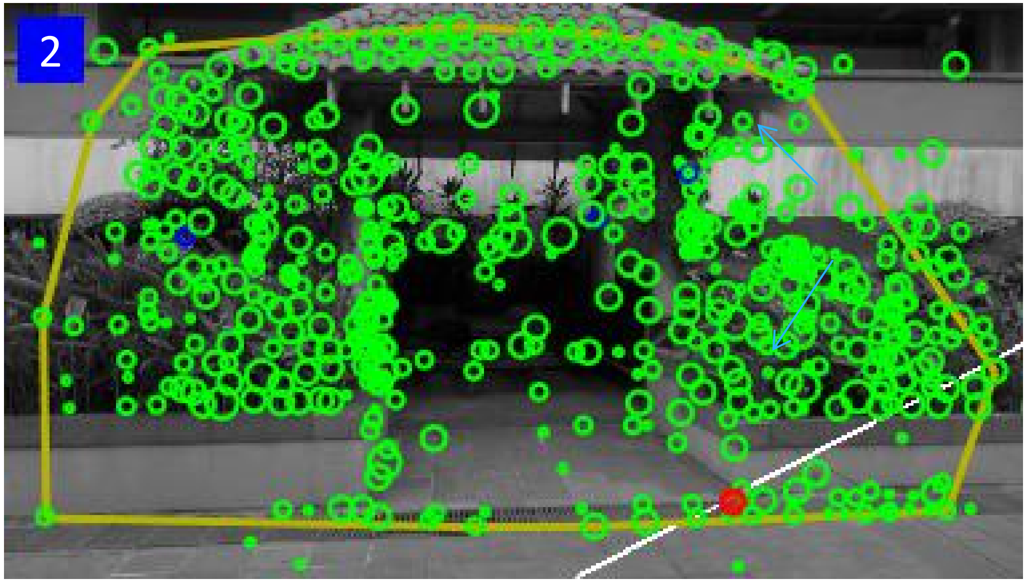


Before group merging and inconsistency removal

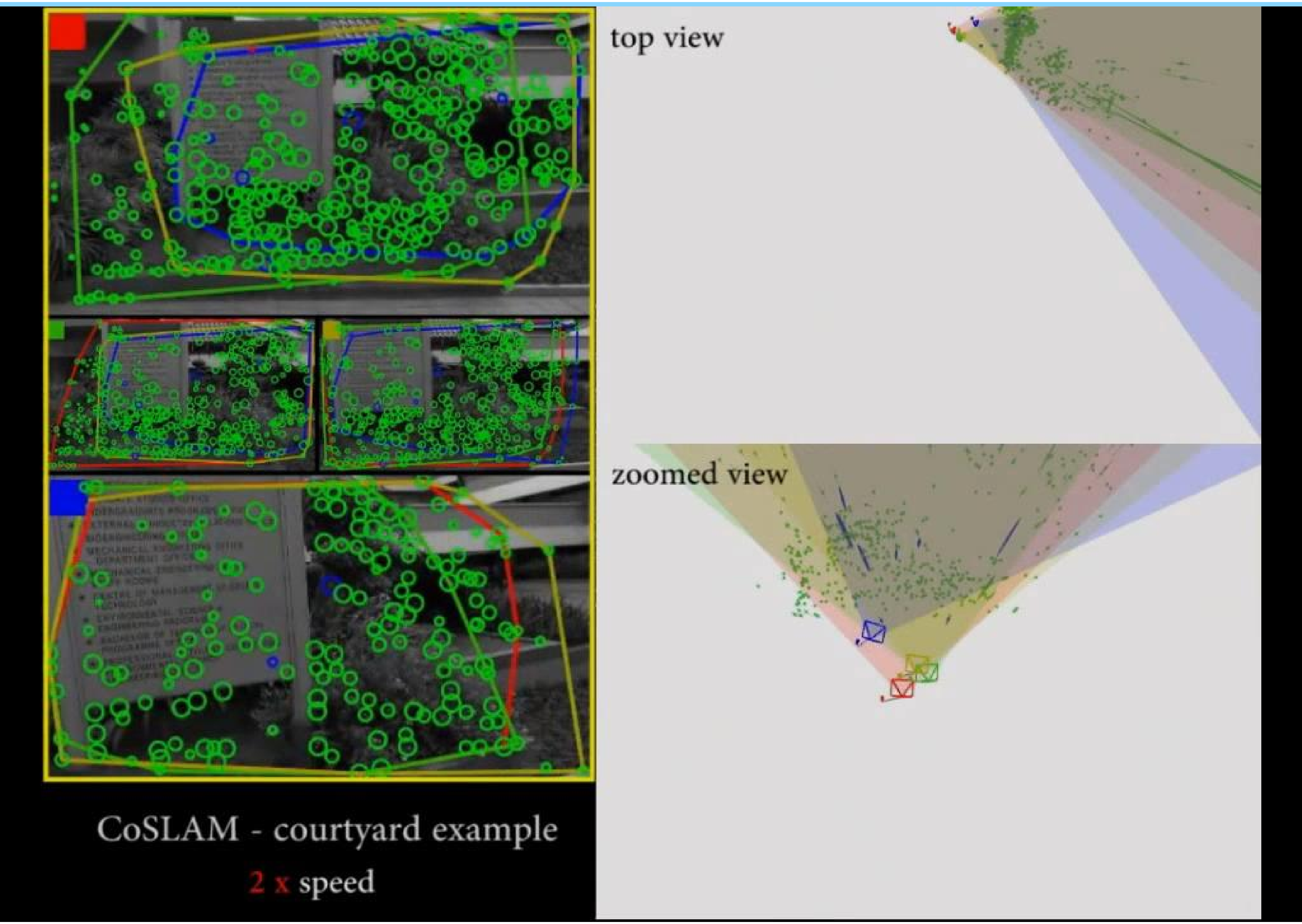
Group #0:

1 3 2 4

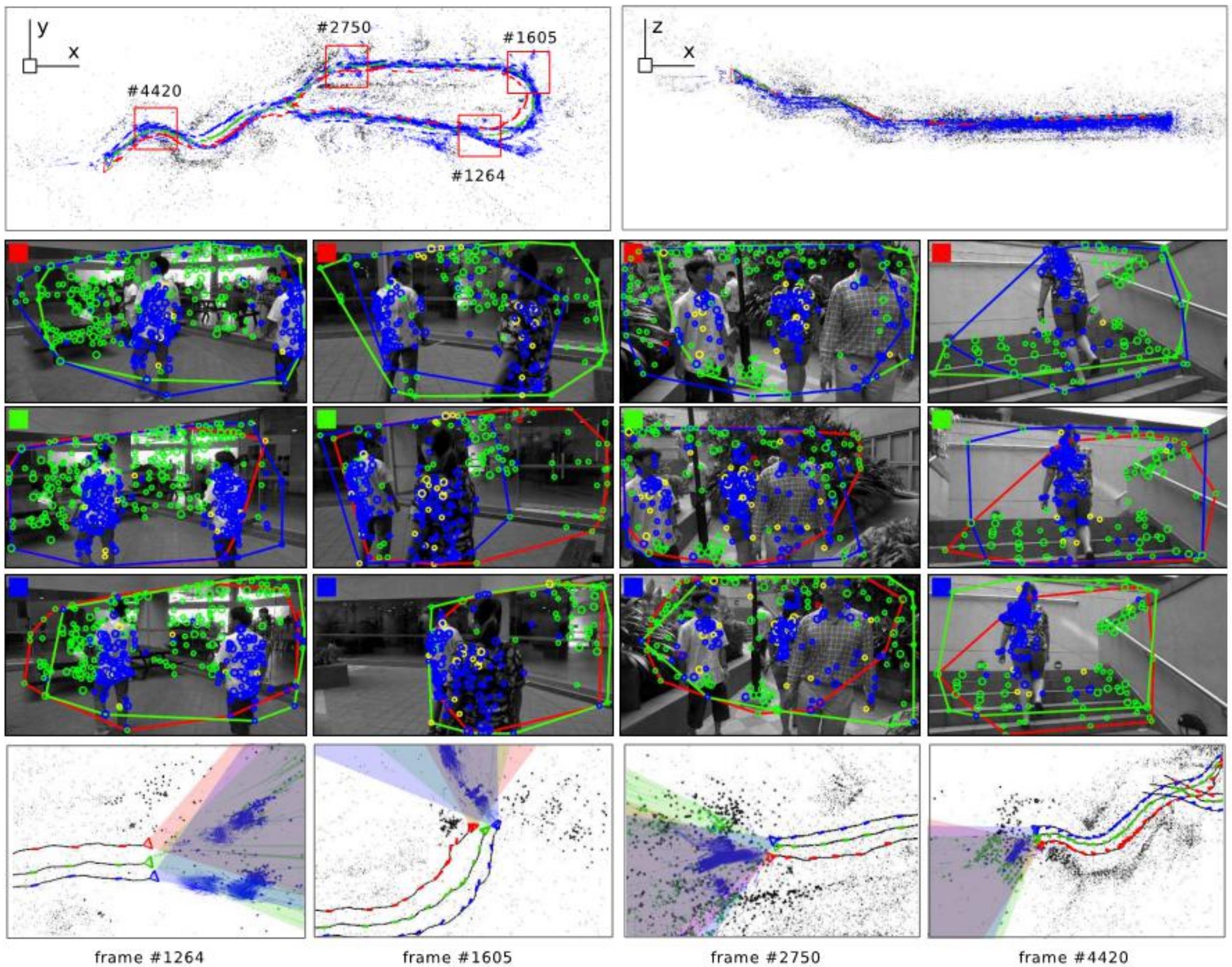


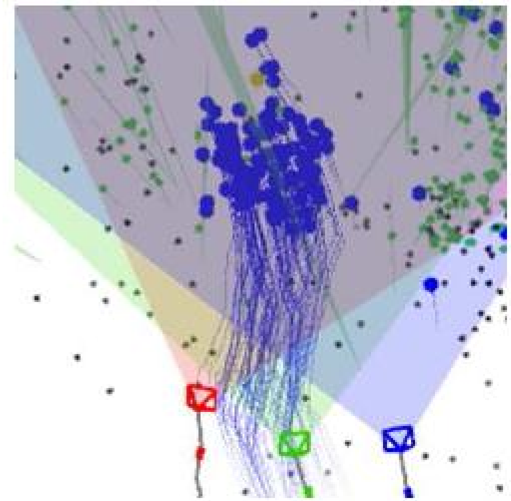


After group merging and inconsistency removal


top view
zoomed view
CoSLAM - courtyard example
2 x speed

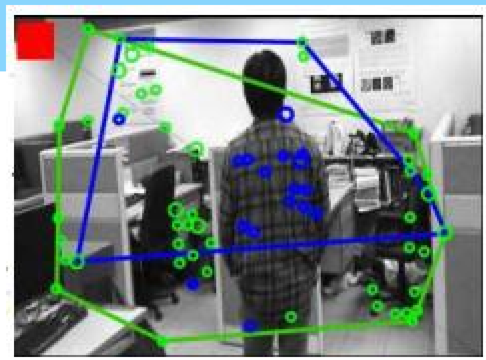
Dynamic scenes



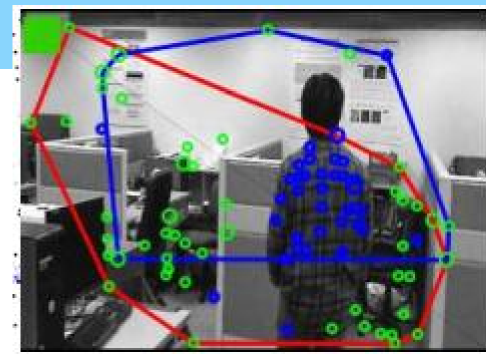


Blue curves are the 3D trajectory of moving points.

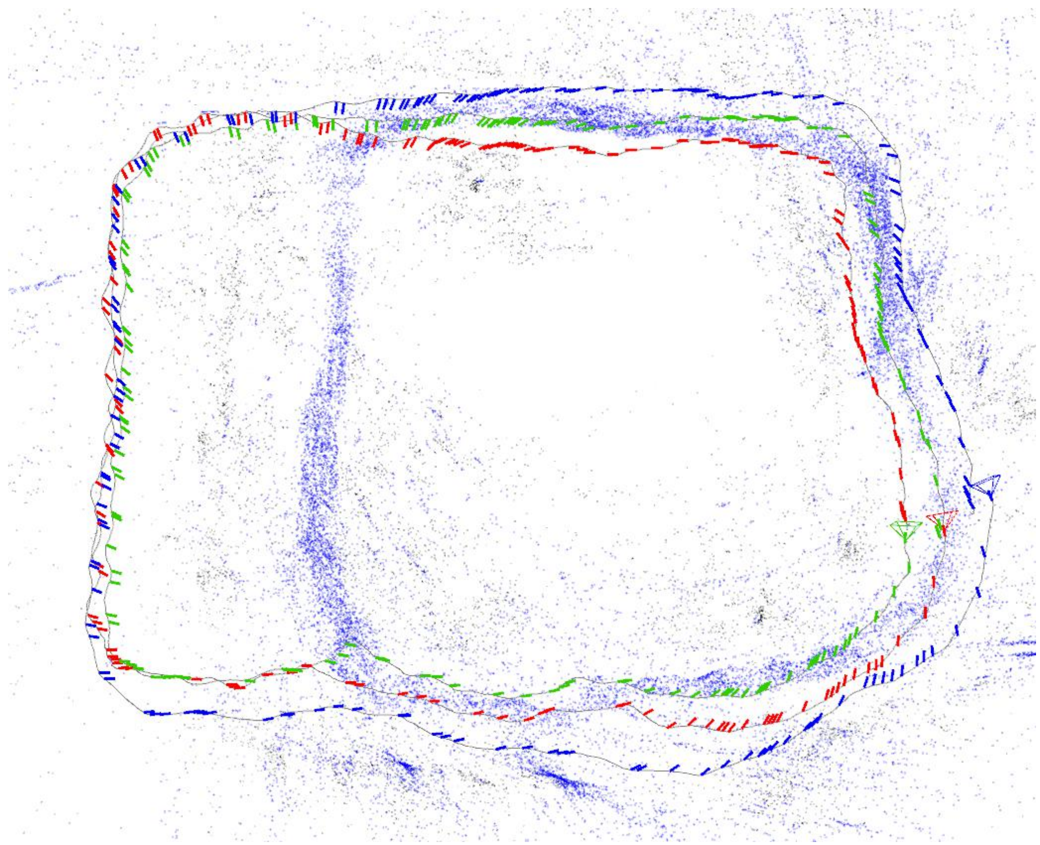
Camera #1

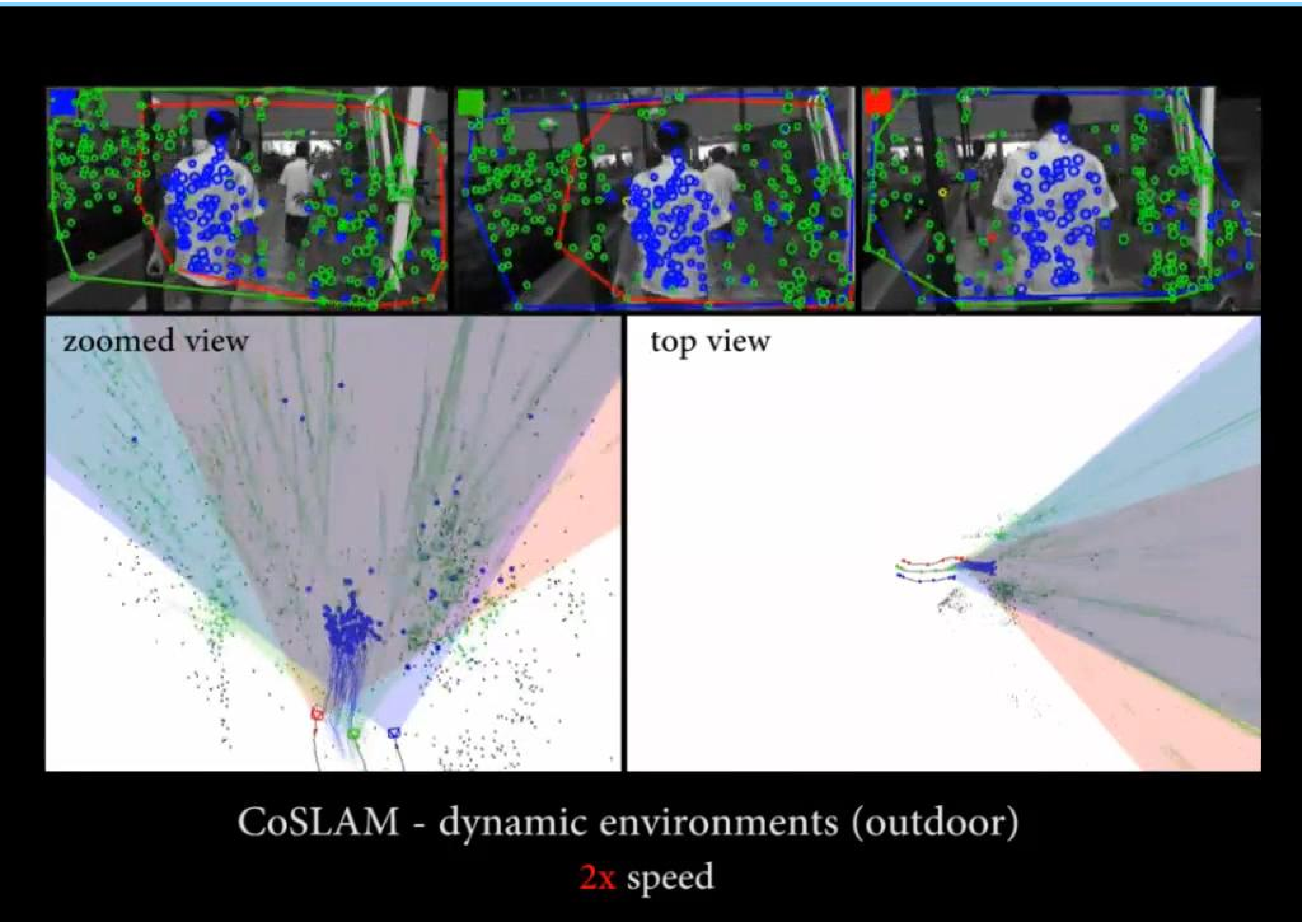


Camera #2



Camera #3




zoomed view
top view
CoSLAM - dynamic environments (outdoor)
2x speed

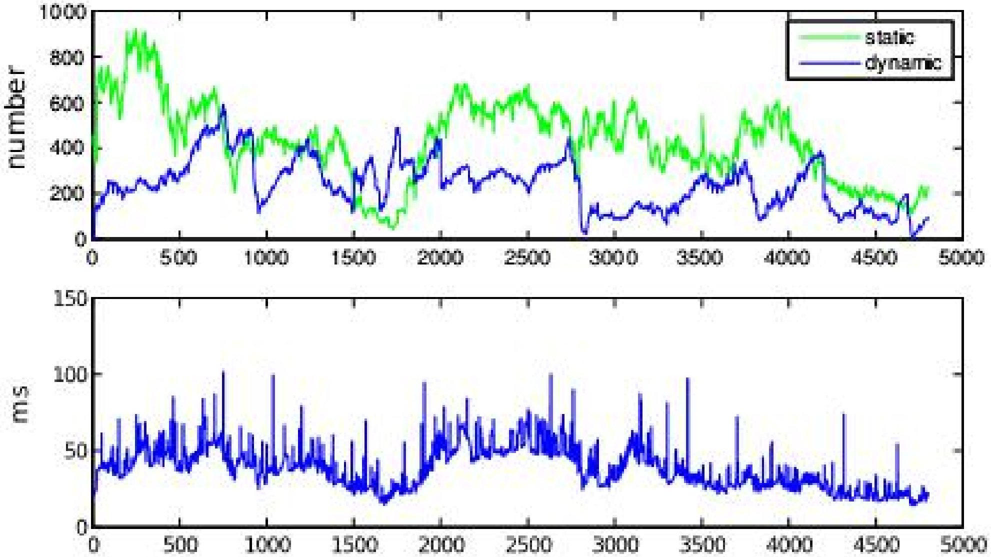
CoSLAM performance

Average timings

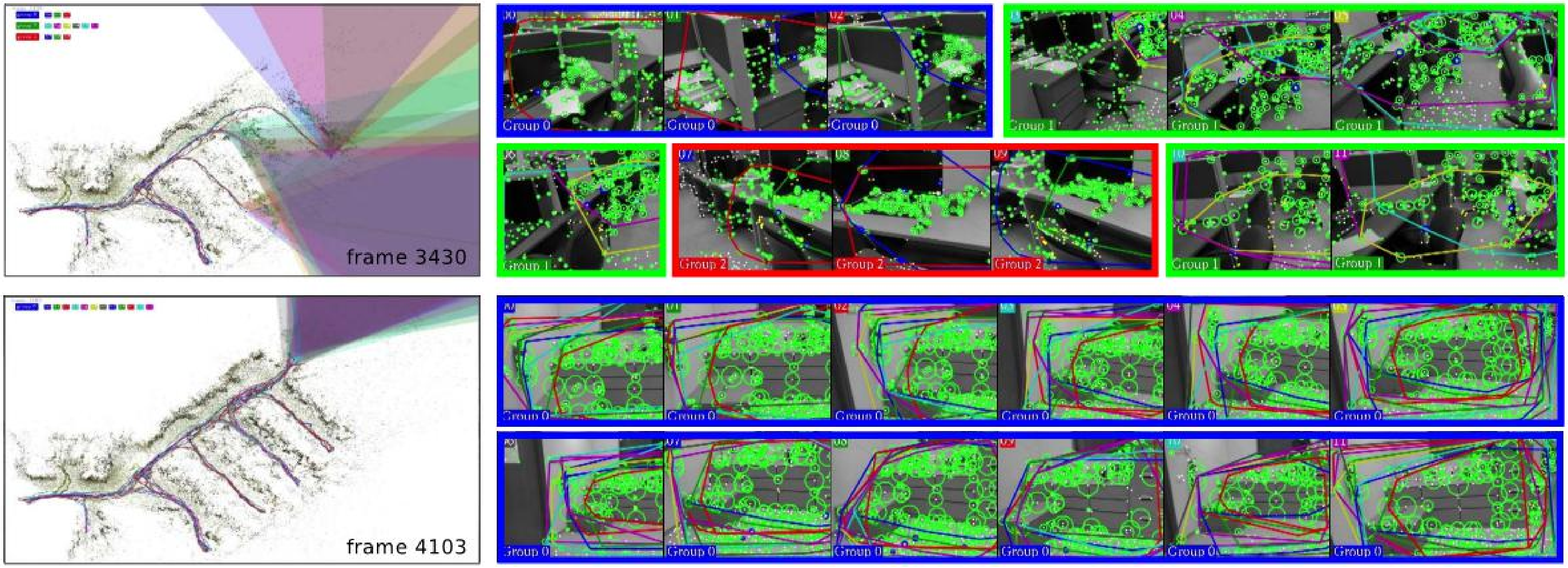
components	ms	calling conditions
feature tracking	8.7	every frame (by GPU)
intra-camera pose estimation	10.7	every frame
inter-camera pose estimation	57.2	see Section 4
map point classification	4.9	every frame
map point registration	14.47	every frame
intra-camera mapping	2.3	see Section 5.2
inter-camera mapping	48.3	see Section 5.2

Nvidia GTX 480

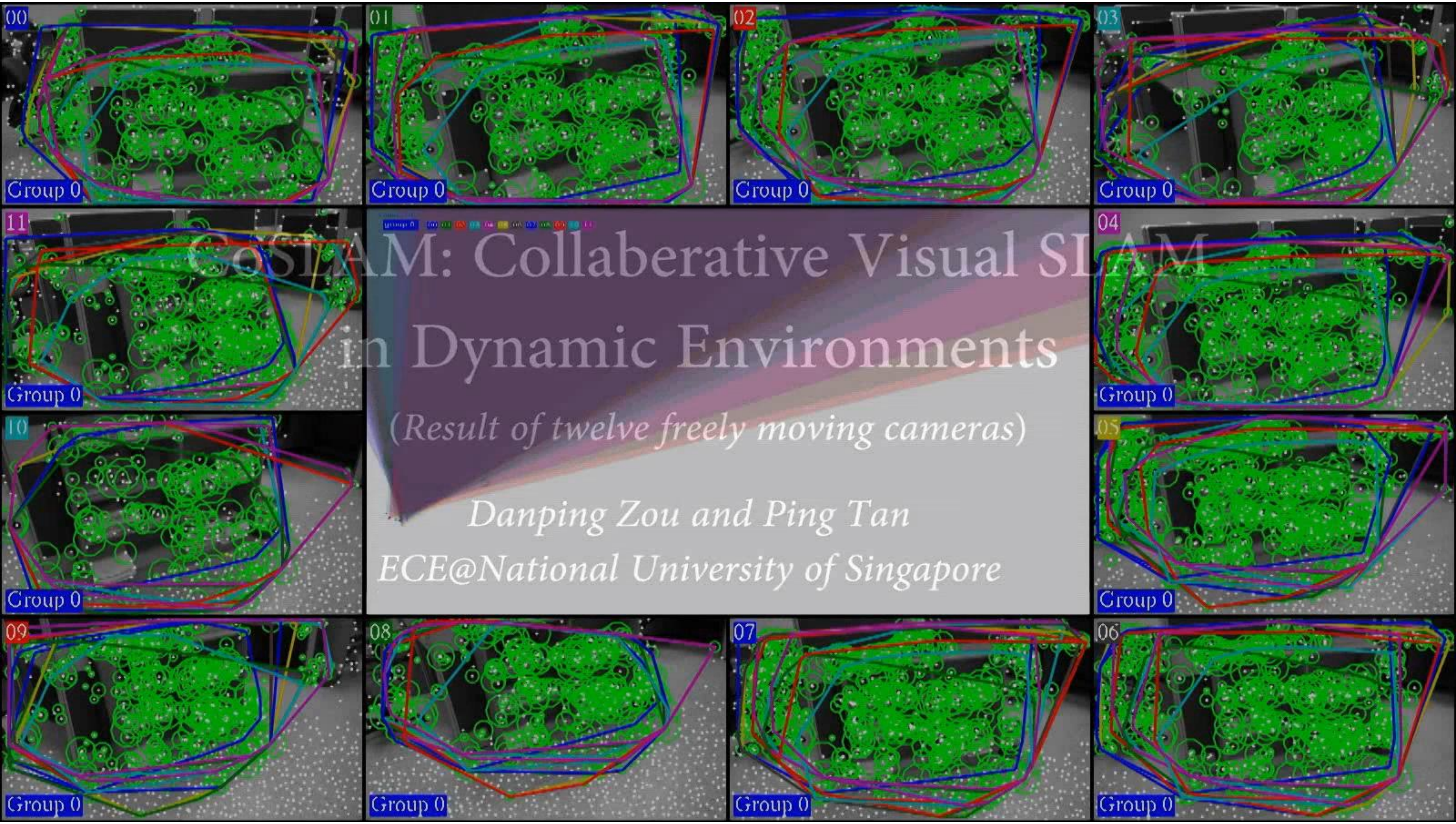
Average run time:
38 ms (**26**fps)



System scalability test



We tested our system scalability with 12 cameras moving independently in a static scene. The cameras form small troops to navigate in the indoor environment through different routes.


CoSLAM: Collaberative Visual SLAM
in Dynamic Environments
(Result of twelve freely moving cameras)
Danping Zou and Ping Tan
ECE@National University of Singapore
00
01
02
03
04
05
06
07
08
09
10
11
Group 0

Real system – Multi-Robot vSLAM

Rui Huang. J.M.Perron, etc. **Orbiting a Moving Target with Multi-Robot Collaborative Visual SLAM,** RSS-MVIGRO, 2015

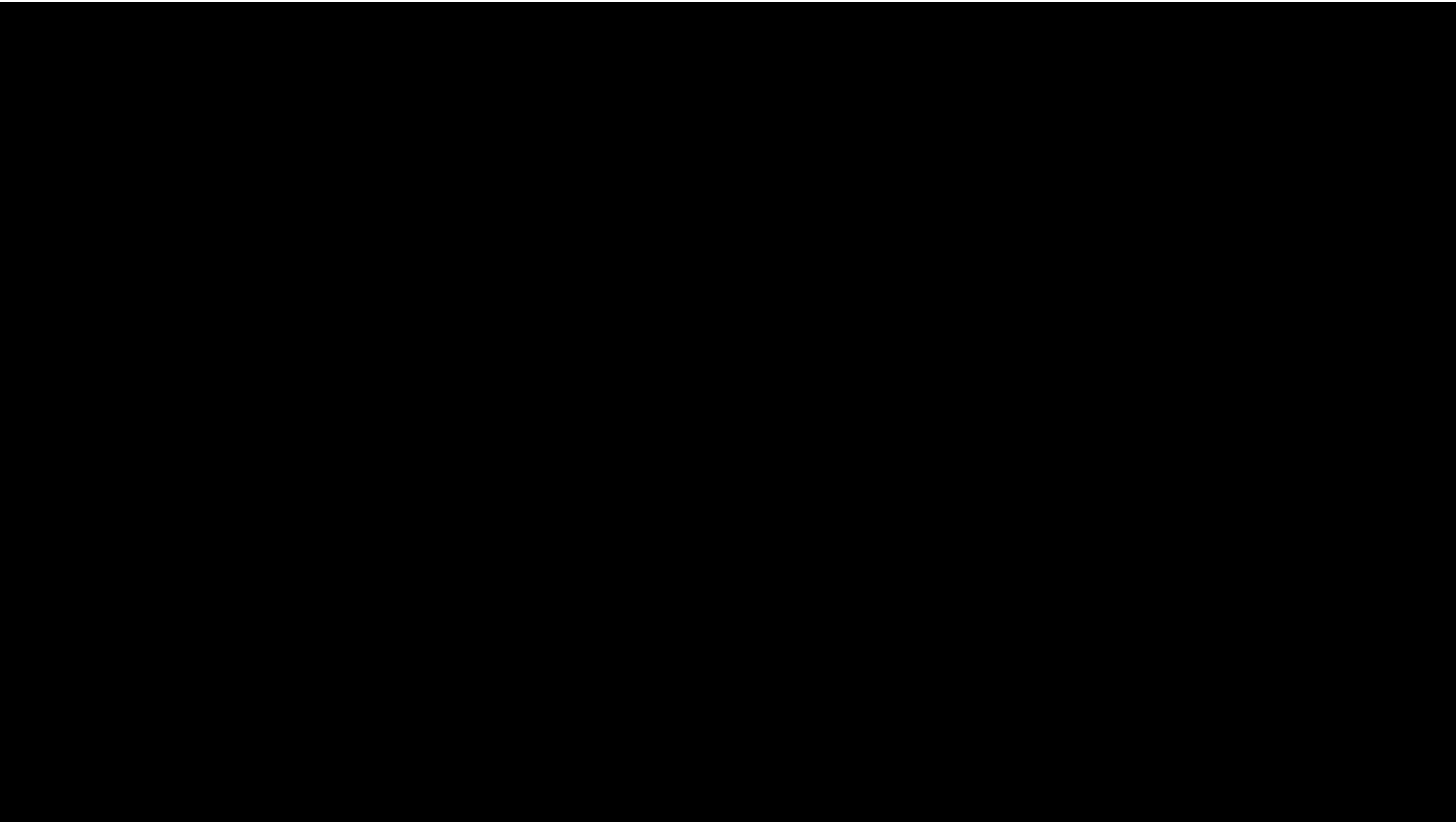
Code

- **CoSLAM (origin)**
  - https://github.com/danping/CoSLAM
- **CoSLAM for multiple robots by Rui Huang(黄睿)**
  - http://huangrui815.github.io/CoSLAM_for_Target_Following/

Rui Huang(黄睿）

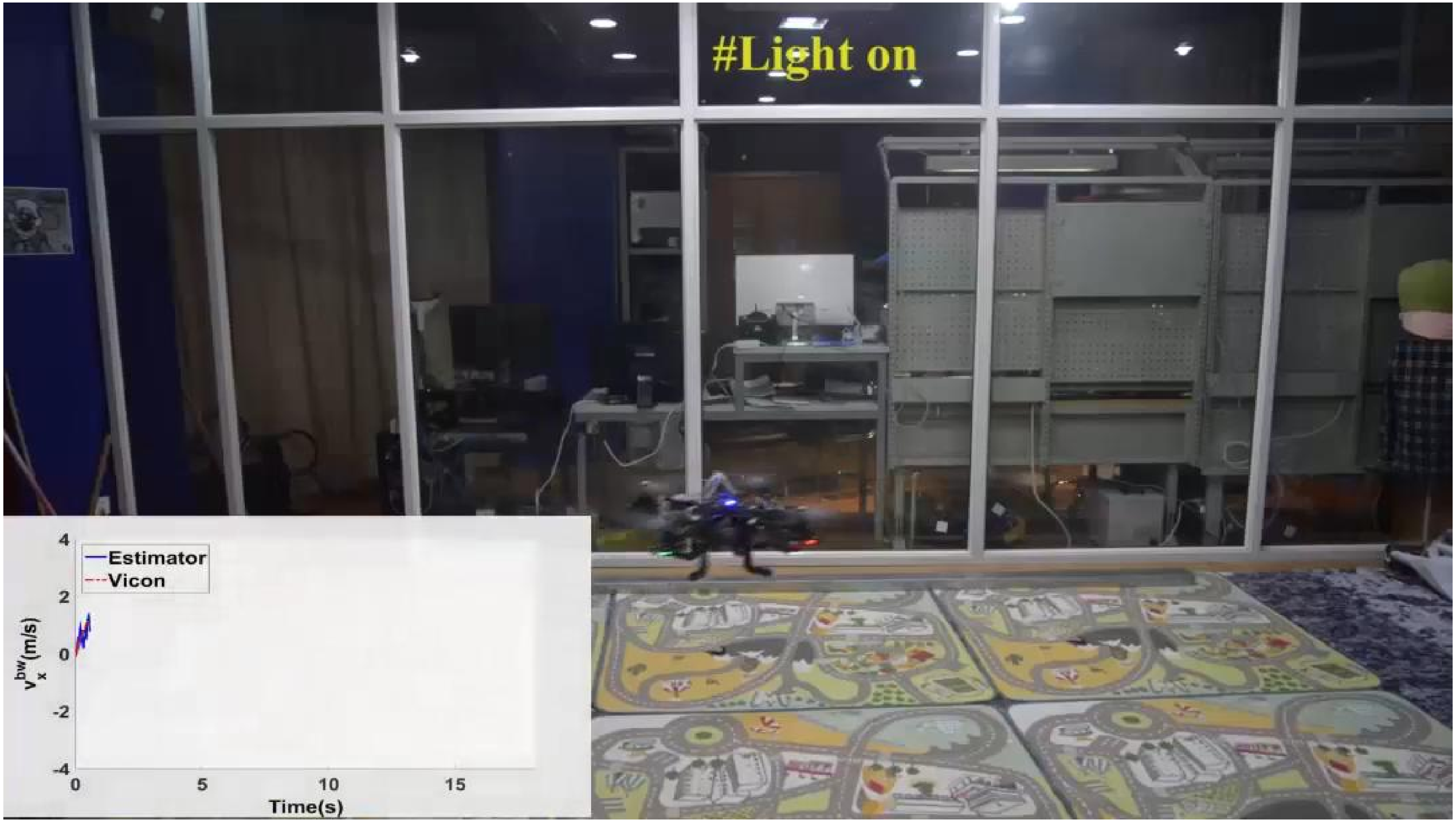
Onboard Visual SLAM /UAV

Active Image Modeling



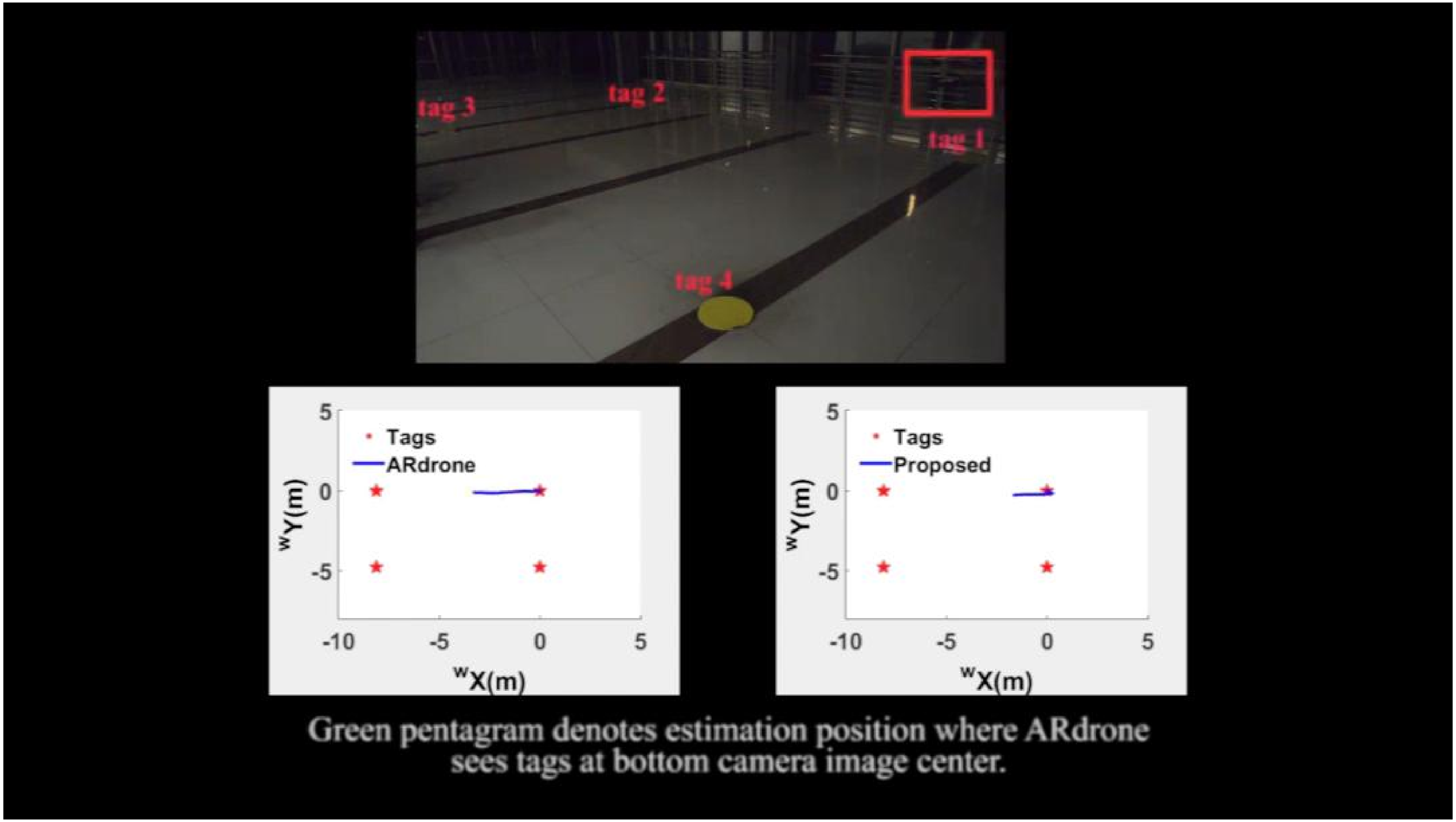
Active Image Modeling

Dark scenes



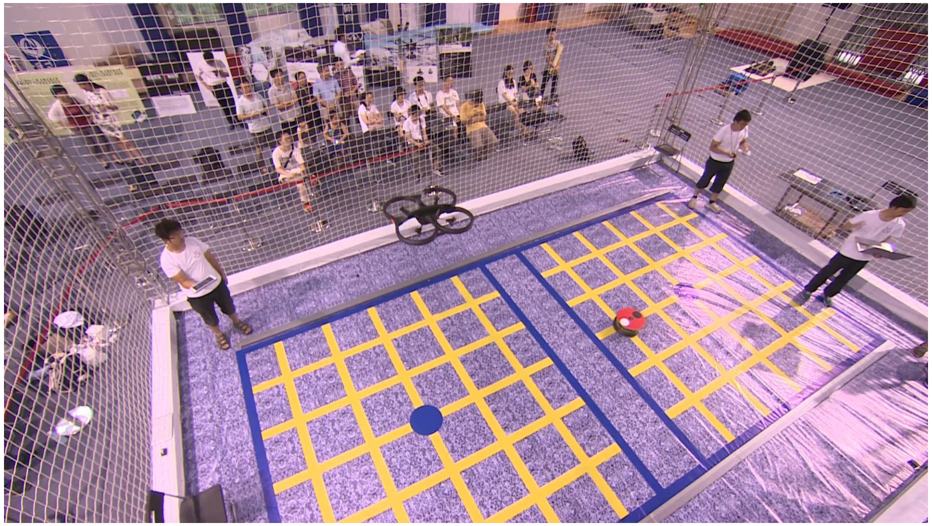
R. Wang, D. Zou etc, **A Robust Aerodynamics-Aided State estimator for Multi-rotor UAVs**, IROS, 2017

Low texture + dark scenes



R. Wang, D. Zou etc, **An Aerodynamics Model-Aided State estimator for Multi-rotor UAVs**, IROS, 2017

SJTU UAV competition



http://drone.sjtu.edu.cn/contest/

招生信息

http://aitee.sjtu.edu.cn/

感知与导航研究所

- 外校考研：学科-信息与通信工程（04电子工程系）
- 博士：
  - 上海交通大学电子系夏令营
- 本校保送：
  - 申请电子工程系“雏鹰”计划，接受跨院系保送
- Abroad: “The Belt and Road” Satellite Navigation and Remote Sensing Program (Master with A-grade National Scholarship)