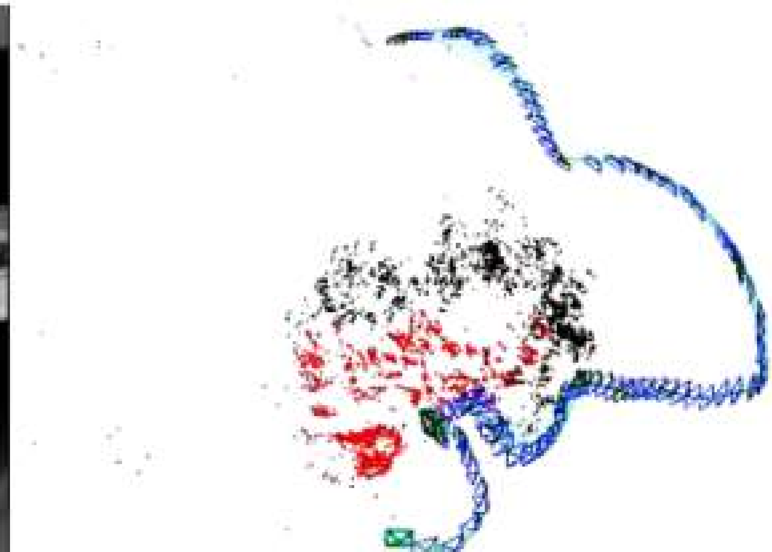
Feature-Based Visual SLAM

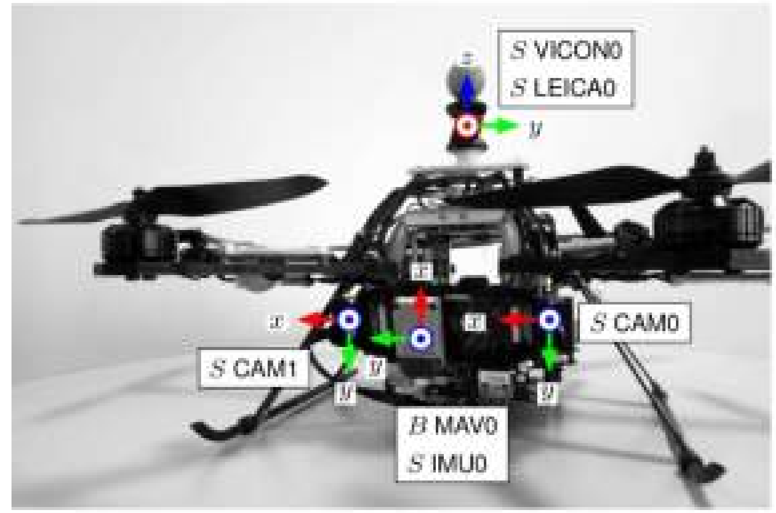
Juan D. Tardós
Universidad de Zaragoza, Spain
robots.unizar.es/SLAMLAB



Outline

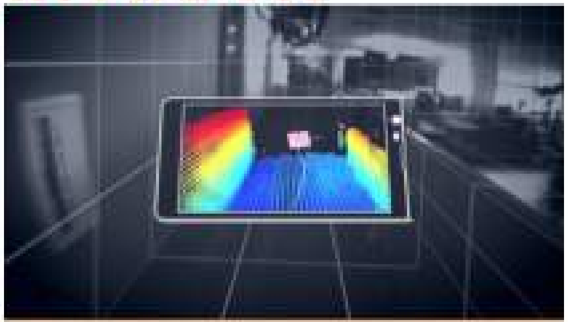
1. Basics: Visual SLAM
2. Features
3. Feature Matching
4. Relocation and Loop Closing
5. Putting all Together
   - Example: ORB-SLAM

1. Basics: Visual SLAM



Visual SLAM for user tracking in AR/VR

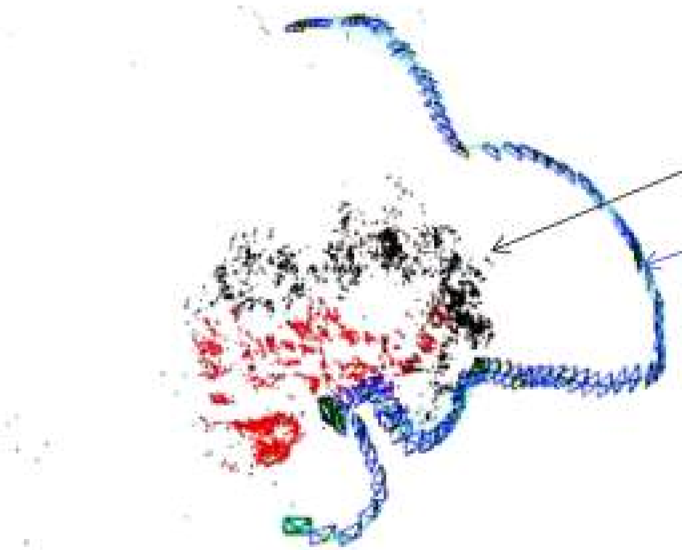
- Project Tango (Google)
  - Area learning (SLAM)
  - Cámara RGB-D



- AR/VR glasses
  - Hololens (Microsoft)
  - Oculus Rift (Facebook)
  - Magic Leap
  - Meta

- Vuforia (Qualcomm)
  - Mobile applications
- Apple (metaio), Oculus (surrealVision),…

Feature-Based Visual SLAM



States

$\mathbf{x}_{wj} \in \mathbb{R}^3$	Coordinates of point *j*
$\mathbf{T}_{iw} \in \mathrm{SE}(3)$	Pose of camera *i*

Measurements

$\mathbf{u}_{ij} = \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix}$	Observation of point *j* from camera *i*

Reprojection error

Projection Function

$$\mathbf{e}_{ij} = \mathbf{u}_{ij} - \pi_i(\mathbf{T}_{iw}, \mathbf{x}_{wj})$$

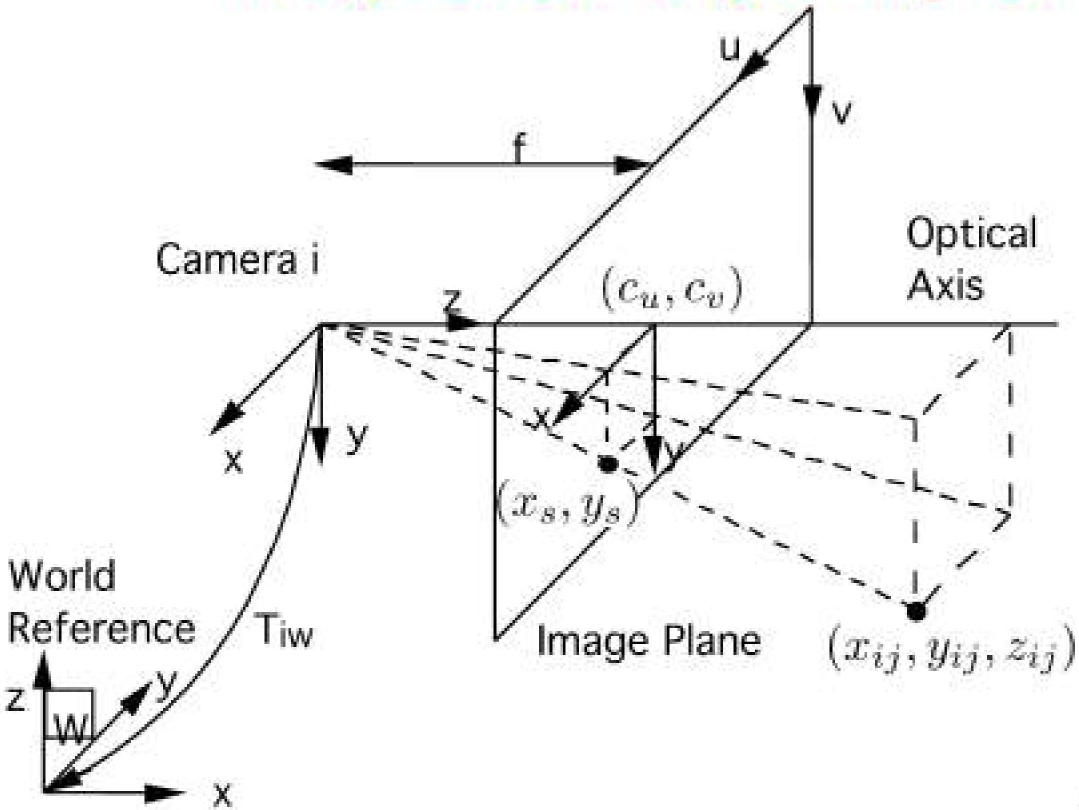
Projection of point j on camera i (1)

$$\mathbf{T}_{iw} \in \mathrm{SE}(3) \begin{cases} \mathbf{R}_{iw} \in \mathrm{SO}(3) & \text{Rotation matrix} \\ \mathbf{t}_{iw} \in \mathbb{R}^3 & \text{Translation vector} \end{cases}$$

$$\mathbf{x}_{ij} = \mathbf{R}_{iw}\mathbf{x}_{wj} + \mathbf{t}_{iw}$$

Coordinates of point *j* w.r.t. camera *i*

Projection of point j on camera i (2)



focal length (mm)

$$x_s = f_i \frac{x_{ij}}{z_{ij}}$$

$$u = (s_u f_i)\frac{x_{ij}}{z_{ij}} + c_{i,u}$$

$$= f_{i,u}\frac{x_{ij}}{z_{ij}} + c_{i,u}$$

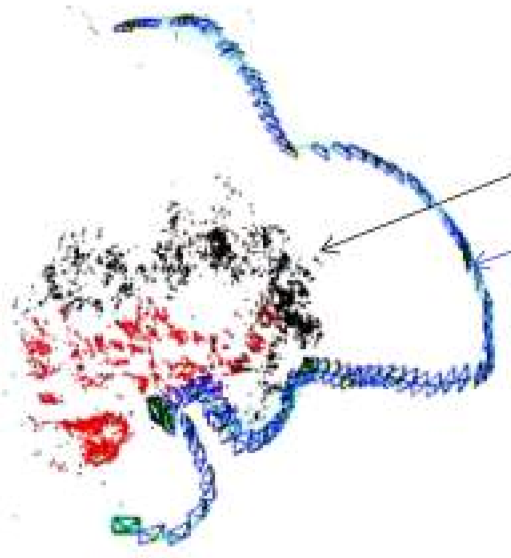
horizontal focal length (pixels)

principal point

- In summary:

$$\pi_i(\mathbf{T}_{iw}, \mathbf{x}_{wj}) = \begin{bmatrix} f_{i,u}\frac{x_{ij}}{z_{ij}} + c_{i,u} \\ f_{i,v}\frac{y_{ij}}{z_{ij}} + c_{i,v} \end{bmatrix}$$

Feature-Based Visual SLAM



States

$\mathbf{x}_{wj} \in \mathbb{R}^3$ Coordinates of point *j*

$\mathbf{T}_{iw} \in \mathrm{SE}(3)$ Pose of camera *i*

Measurements

$\mathbf{u}_{ij} = \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix}$ Observation of point *j* from camera *i*

- Find the state values minimizing the reprojection errors:

$$\mathbf{e}_{ij} = \mathbf{u}_{ij} - \pi_i(\mathbf{T}_{iw}, \mathbf{x}_{wj})$$

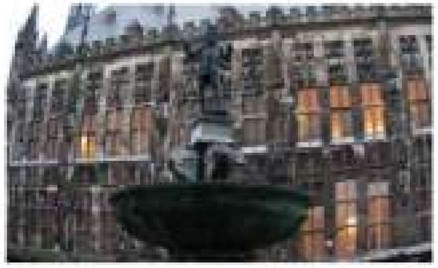
Bundle Adjustment

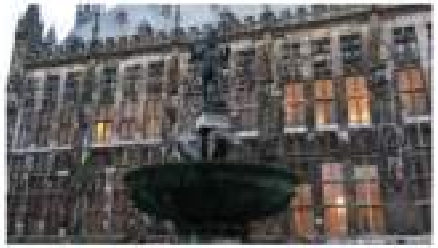
$$\{\mathbf{T}_{1w}..\mathbf{T}_{nw}, \mathbf{x}_{w1}..\mathbf{x}_{wm}\}^\star = \arg\min_{\mathbf{T},\mathbf{x}} \sum_{i,j} \rho_h(\mathbf{e}_{ij}^T \mathbf{\Sigma}_{ij}^{-1} \mathbf{e}_{ij})$$

Some details

$$\mathbf{e}_{ij} = \mathbf{u}_{ij} - \pi_i(\mathbf{T}_{iw}, \mathbf{x}_{wj})$$

$$\{\mathbf{T}_{1w}..\mathbf{T}_{nw}, \mathbf{x}_{w1}..\mathbf{x}_{wm}\}^\star = \arg\min_{\mathbf{T},\mathbf{x}} \sum_{i,j} \rho_h(\mathbf{e}_{ij}^T \mathbf{\Sigma}_{ij}^{-1} \mathbf{e}_{ij})$$





- Assumption: the camera has been calibrated
  - Focal lengths and principal point are known
  - Distortion can be corrected
- $\rho_h(\,)$ robust cost function (i.e. Huber cost) to downweight wrong matchings
- $\Sigma_{ij} = \sigma_{ij}^2 \mathbf{I}_{2\times 2}$ std. dev. typically = 1 pixel * scale

Huber cost function

- L2 cost (quadratic)

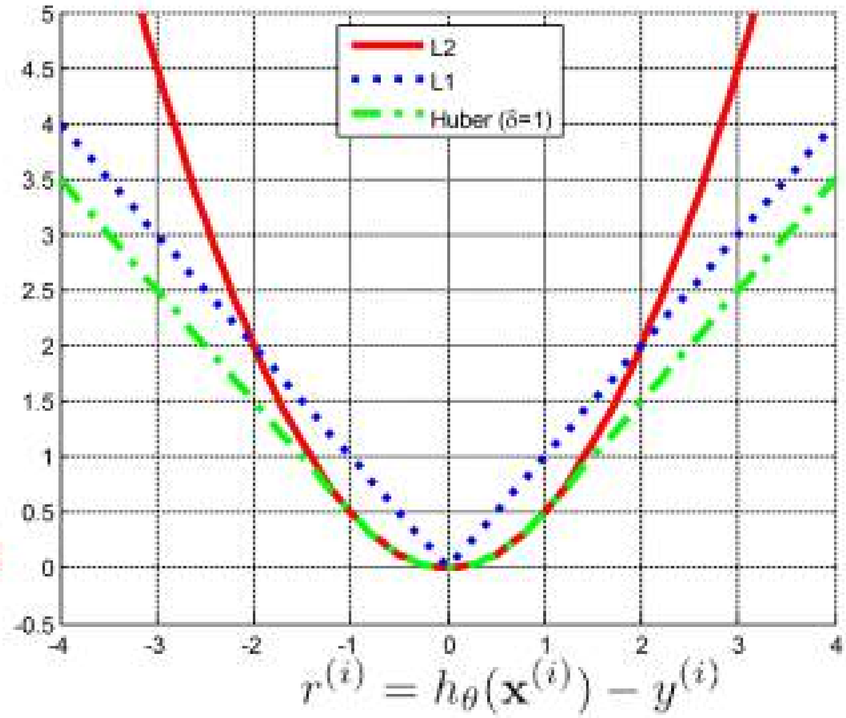
$$J_{L2}(\theta) = \frac{1}{2}\sum_{i=1}^{N}\left(h_\theta(\mathbf{x}^{(i)}) - y^{(i)}\right)^2$$

Differentiable ☺

- L1 cost (absolute value)

$$J_{L1}(\theta) = \sum_{i=1}^{N}\left|h_\theta(\mathbf{x}^{(i)}) - y^{(i)}\right|$$

Non differentiable ☹



- Huber cost:

$$L_H(r,\delta) = \begin{cases} r^2/2 & \text{if } |r| \le \delta \\ \delta|r| - \delta^2/2 & \text{if } |r| > \delta \end{cases}$$

Differentiable ☺

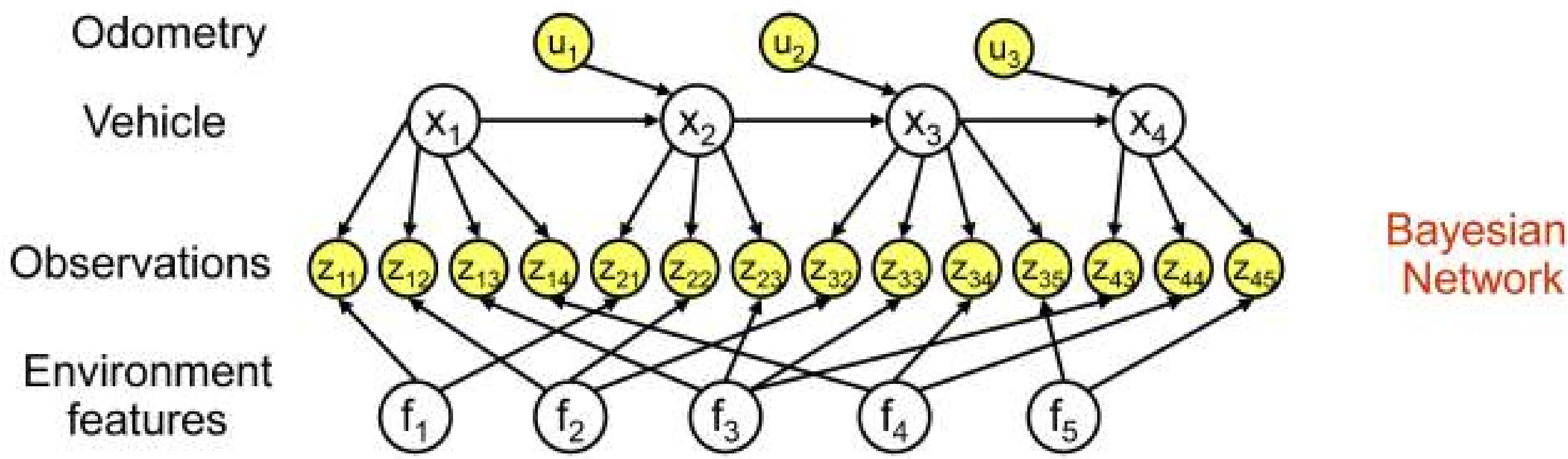
$$J_H(\theta) = \sum_{i=1}^{N} L_H(r^{(i)},\delta) = \sum_{\left|r^{(i)}\right|\le\delta} {r^{(i)}}^2/2 + \sum_{\left|r^{(i)}\right|>\delta} \delta\left|r^{(i)}\right| - \delta^2/2$$

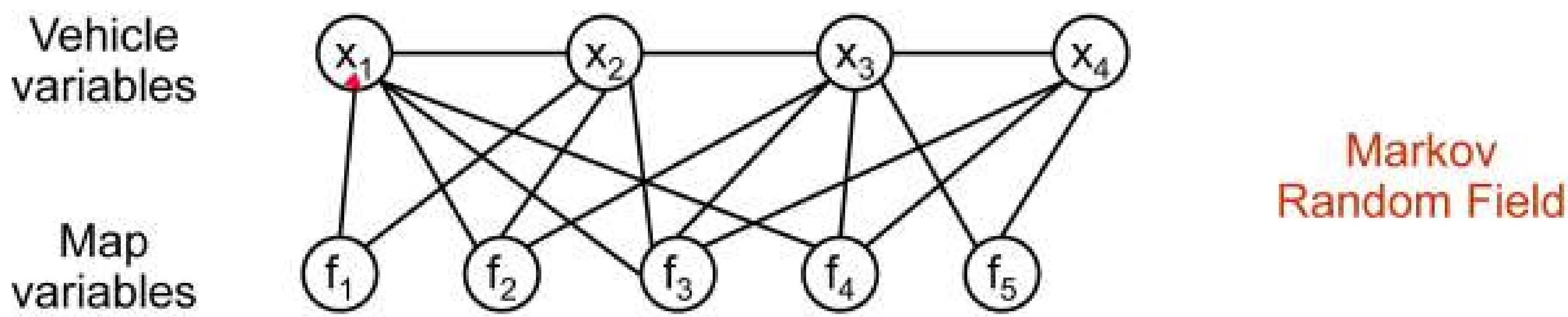
Full Bundle Adjustment in Real Time?

$$\{\mathbf{T}_{1w}..\mathbf{T}_{nw}, \mathbf{x}_{w1}..\mathbf{x}_{wm}\}^{\star} = \arg\min_{\mathbf{T},\mathbf{x}} \sum_{i,j} \rho_h(\mathbf{e}_{ij}^T \mathbf{\Sigma}_{ij}^{-1} \mathbf{e}_{ij})$$

- The problem is sparse
  - Not all cameras see all points!
- But still not feasible in real time
  - example: 1k images and 100k points → 1s per LM iteration
- Local BA or sliding-window BA
- BA requires very good initial solutions

Structure of the SLAM problem



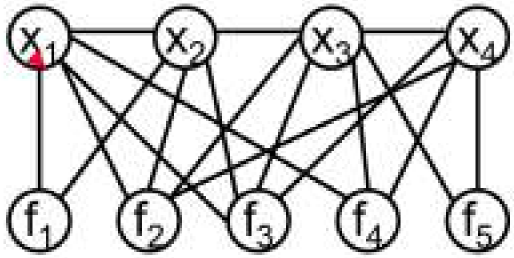


SLAM Problem

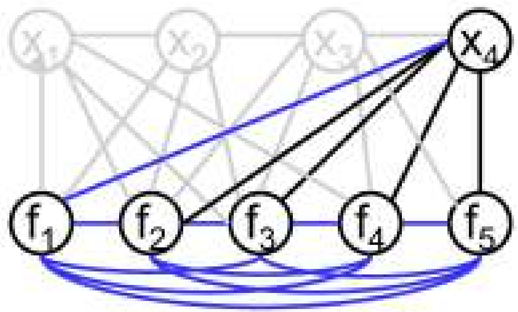
$$p(x_{1:k}, f_{1:n} \mid z_{1:k}, u_{1:k})$$

- The problem size grows with time
- The set of relationships is sparse

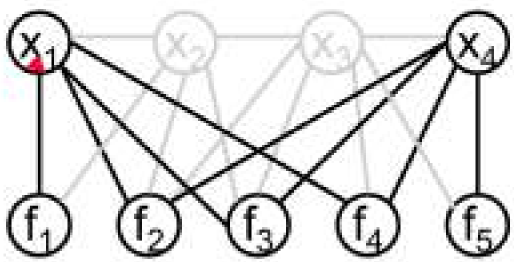
Maps with Thousands of Features?



- Original SLAM problem



- EKF approach
  - Only keeps the last pose
  - $O(n^2)$ with the number of features
  - Limited to 200-300 features in real-time



- Keyframe approach (PTAM)
  - Uses only a few keyframes for map estimation with non-linear optimization
  - Can handle thousands of points
  - Given the same computational effort is more precise than EKF-SLAM

Hauke Strasdat, J. M. M. Montiel, Andrew J. Davison, **Real-time Monocular SLAM: Why Filter?.** IEEE Int. Conf. Robotics and Automation, ICRA 2010.

BA + Keyframes, what else do I need?

- Which features will I use?
- How to match them?
- How to start when the map is empty?
- How to track the camera pose?
- How to add new points to the map?
- How to make it run in real time?
  - Which information to keep, what to throw away?
- What if objects or people move?
- What if I get lost?
- How to detect a loop?
- How to correct drift after a loop?

2. Features

Local Features, Interest points, Keypoints

- Detector: find local maxima of a certain operator

original Image

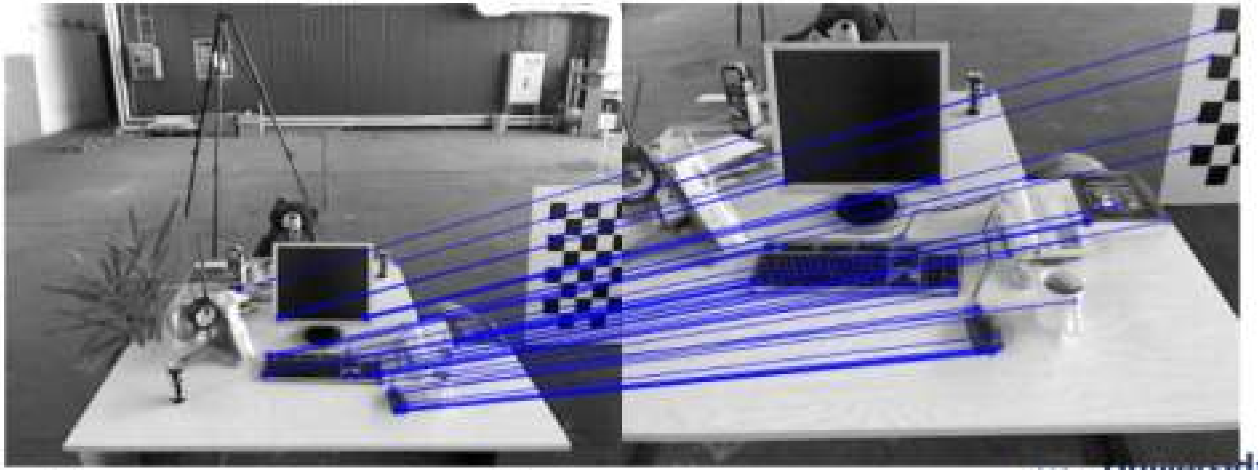
Harris detector
(corner-like)

DoG detector
(blob-like)

- Descriptor: to recognize the feature in new images

Feature Requirements

- Repeatability
- Accuracy
- Invariance
  - Illumination
  - Position
  - In-plane rotation
  - Viewpoint
  - Scale
- Efficiency



Corner detectors

- Harris Matrix or Moments Matrix:

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

  - I_x I_y: Image gradients
  - w: circular weights (uniform or Gaussian)
  - < >: sum over the image patch (*u*,*v*), weighted with *w*

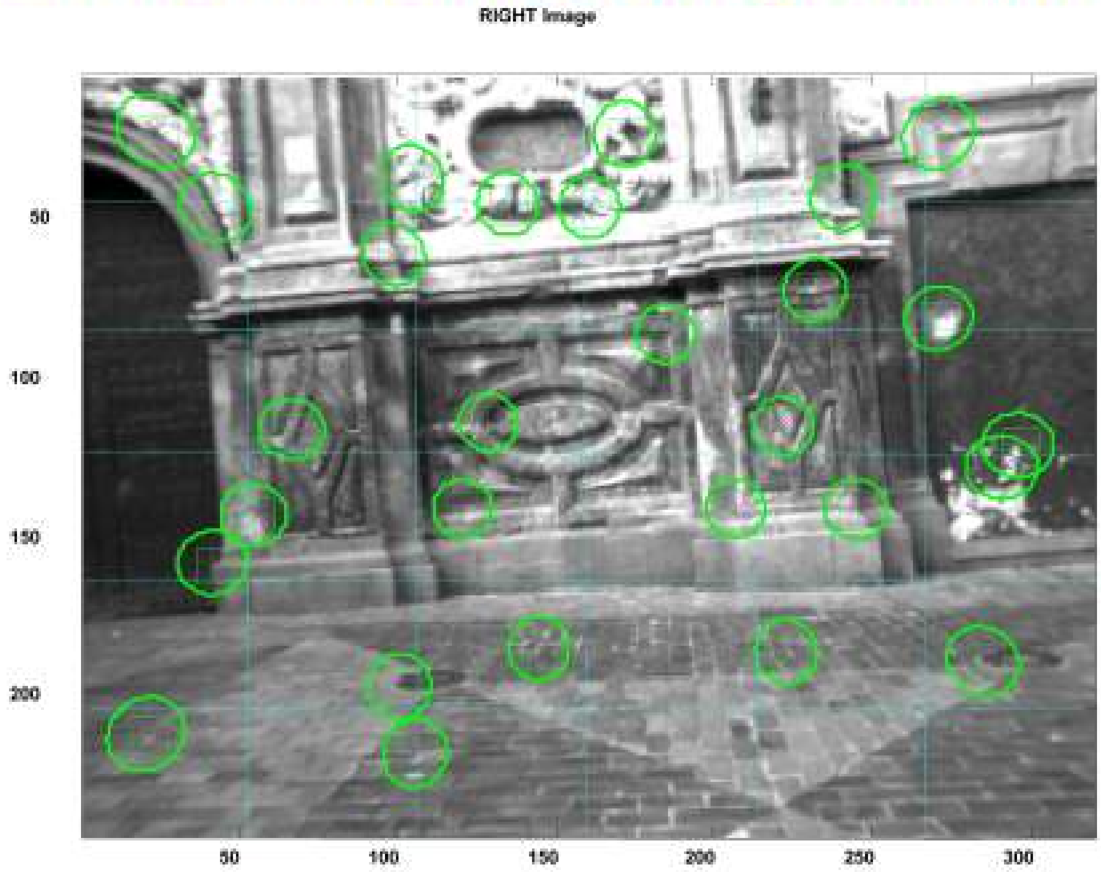
- Harris detector:

$$M_c = det\mathbf{A} - \alpha tr^2\mathbf{A} = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \qquad \alpha = 0.04 .. 0.15$$

- Shi-Tomasi detector:

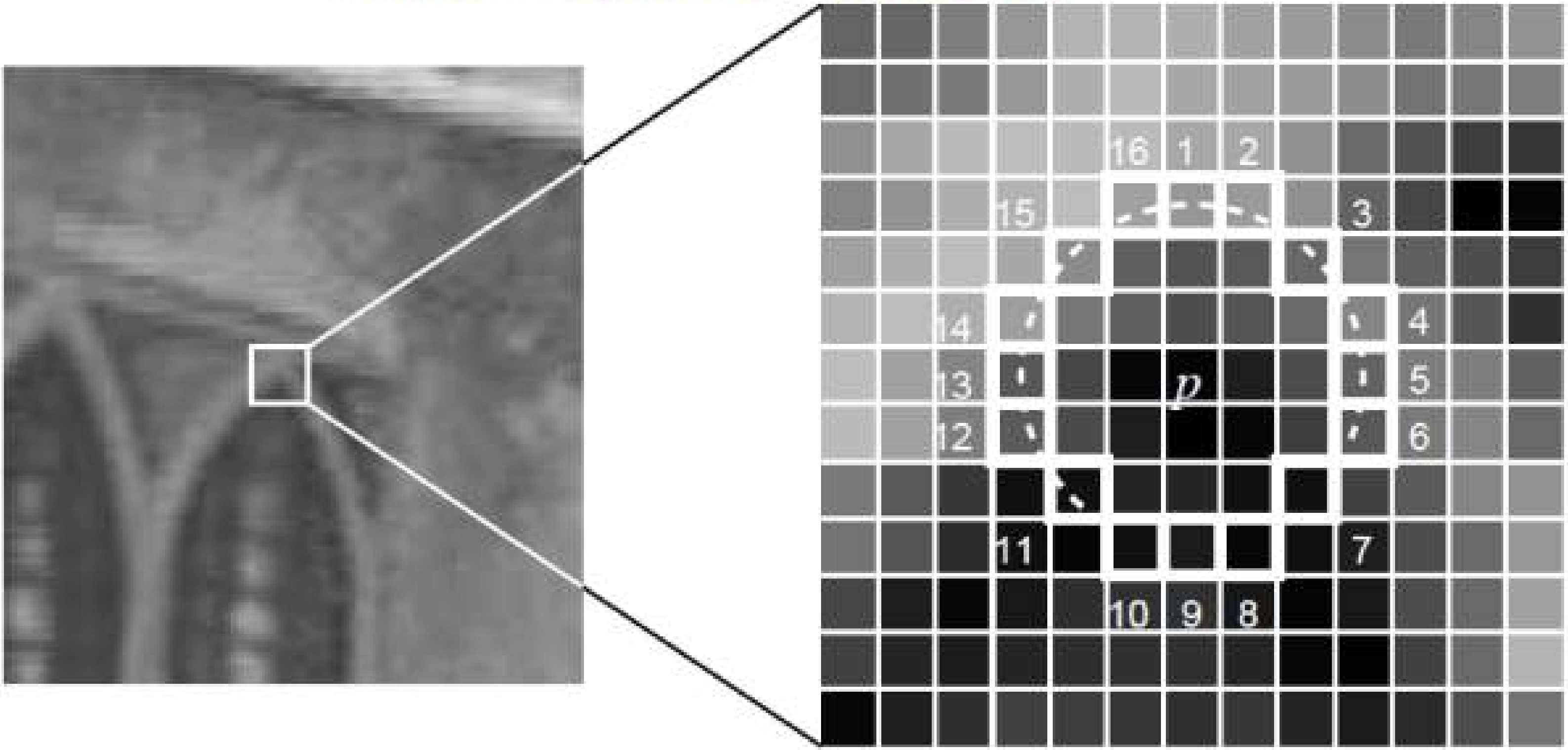
$$M_c = \min(\lambda_1, \lambda_2) \qquad (\lambda_1, \lambda_2) = eig(A)$$

Good for Tracking using Correlation



- Shi-Tomasi points
- Predict position in next image (@15-30 Hz)
- Search by normalized correlation with a 11x11 patch

FAST corner detector

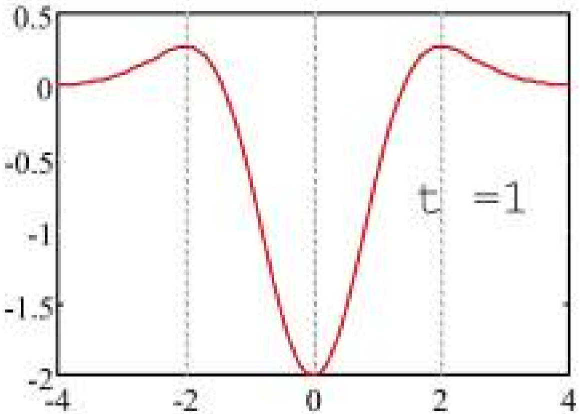


- Pixel p surrounded by n consecutive pixels all brighter (or darker) than p
- Much faster than other detectors

E Rosten, T Drummond , Machine learning for high-speed corner detection, European Conf. on Computer Vision 2006

Blob detector using LoG

- Gaussian Filter (scale t) $L(x,y,t) = g(x,y,t) * f(x,y)$
- Laplacian of Gaussian (LoG) $\nabla^2 L = L_{xx} + L_{yy}$
- Normalized LoG $\nabla^2_{norm} L(x,y;t) = t(L_{xx} + L_{yy})$



- Feature detector:

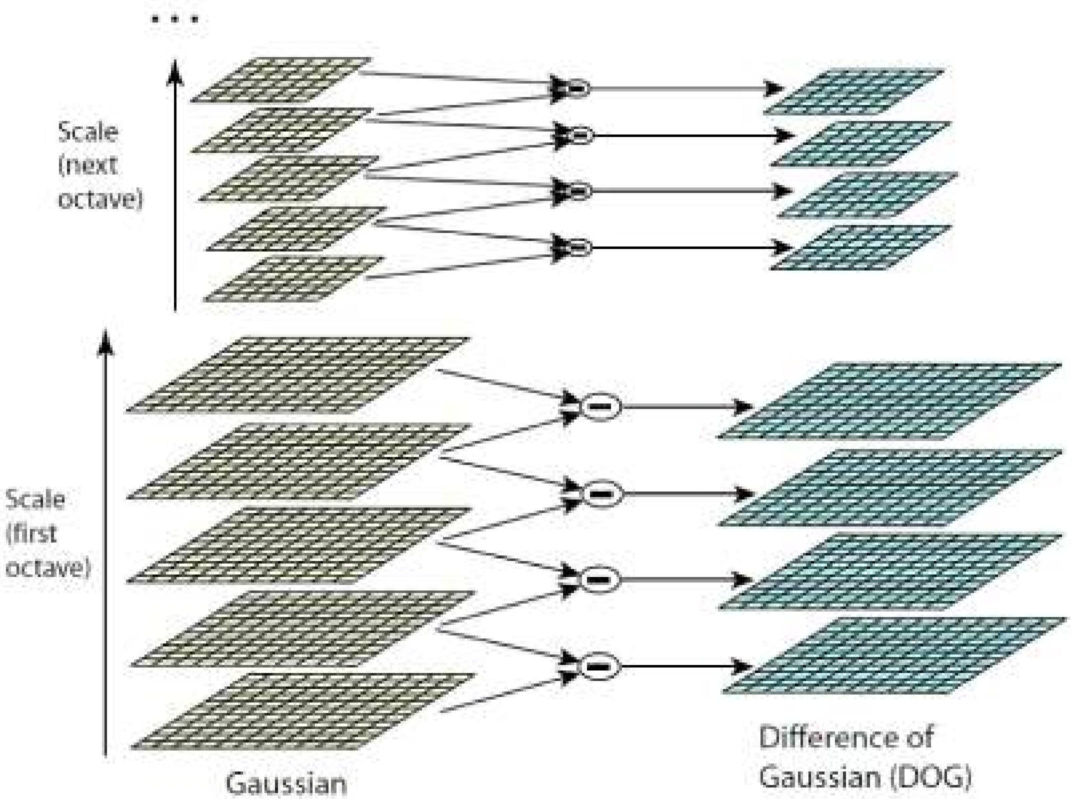
$$(\hat{x},\hat{y};\hat{t}) = \mathrm{argmaxminlocal}_{(x,y;t)}(\nabla^2_{norm} L(x,y;t))$$

  – Strong response for blobs of size $\sqrt{t}$

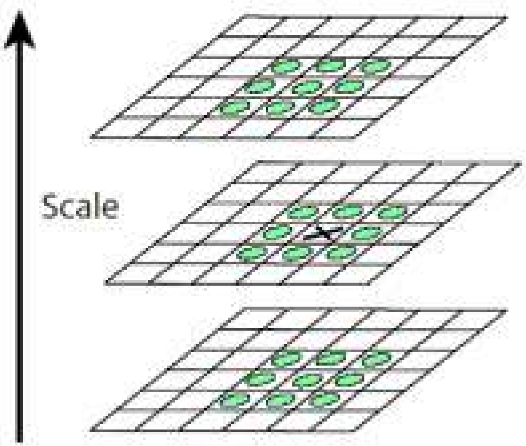
SIFT detector: Difference of Gaussians

- LoG ≈ Difference of Gaussians DoG:

$$\nabla^2 L(x,y;t) = \frac{1}{2\Delta t}\left(L(x,y;t+\Delta t) - L(x,y;t-\Delta t)\right)$$



Search for maxima in space and scale



Automatic scale selection

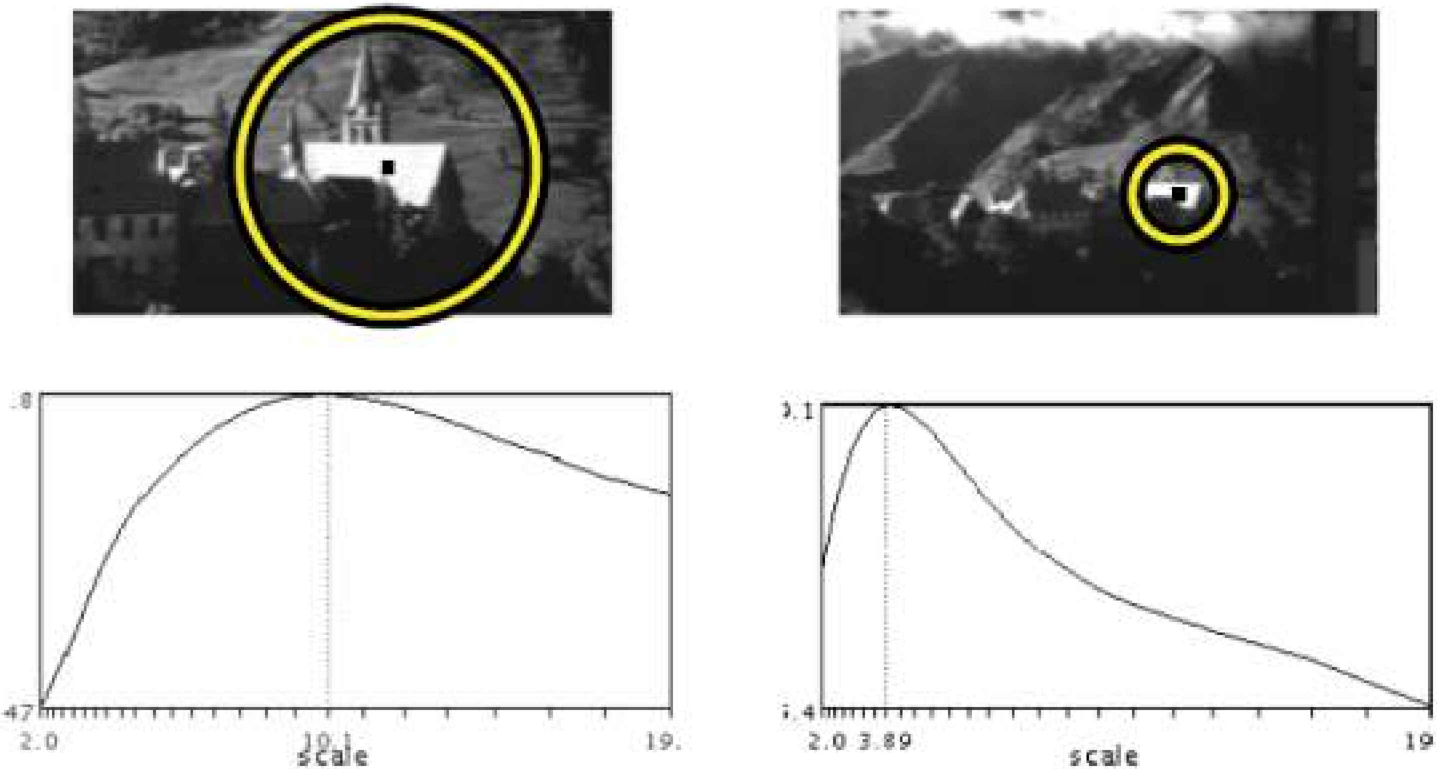
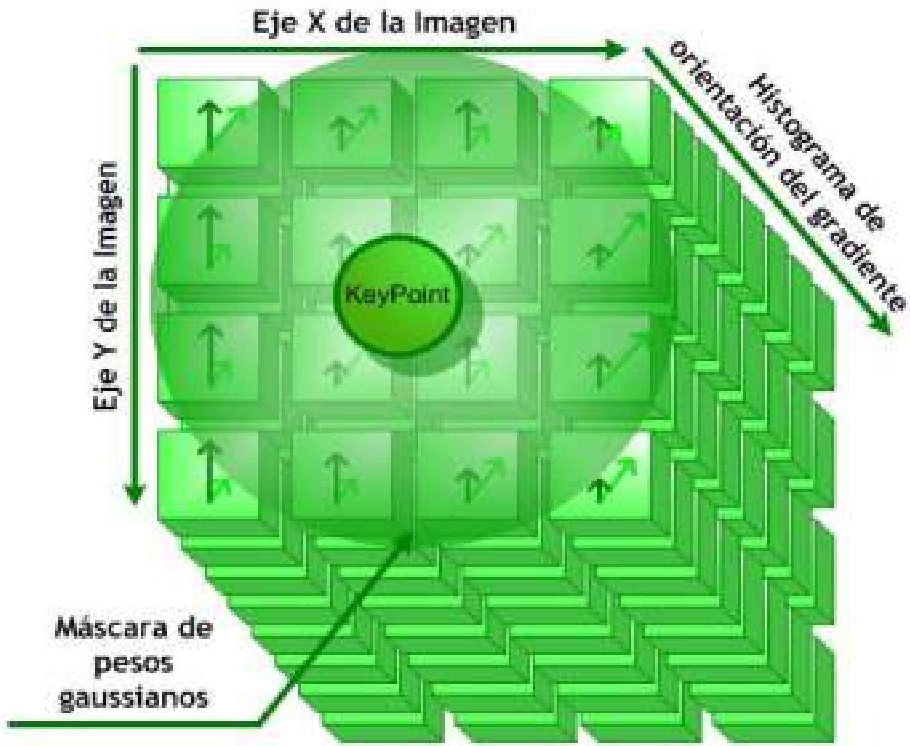


Fig. 3.5 Example of characteristic scales. The top row shows images taken with different zoom. The bottom row shows the responses of the Laplacian over scales for two corresponding points. The characteristic scales are 10.1 and 3.9 for the left and right images, respectively. The ratio of scales corresponds to the scale factor (2.5) between the two images. The radius of displayed regions in the top row is equal to 3 times the selected scales.

SIFT Descriptor

- Histogram of 8 gradient orientations in 16 areas of 4x4 pixels around the detected keypoint

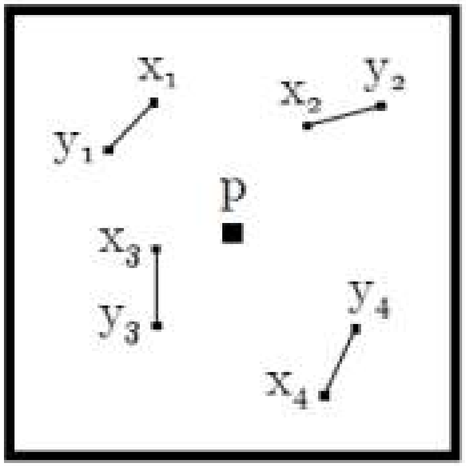


  - 128 bytes (floats): 16 areas x 8 histogram bins

Binary descriptors: BRIEF

- Computed around a FAST corner

BRIEF descriptor:



$$D_i(\mathbf{p}) = \begin{cases} 1 & \text{if } I(\mathbf{p}+\mathbf{x}_i) < I(\mathbf{p}+\mathbf{y}_i) \\ 0 & \text{otherwise} \end{cases}$$

$$\hookrightarrow D(\mathbf{p}) = [\,1\,0\,0\,1\,1\,0\,0\,0\,1 \ldots\,]$$

- Binary string, 256 bits in length.
- It is not invariant to scale or rotation.

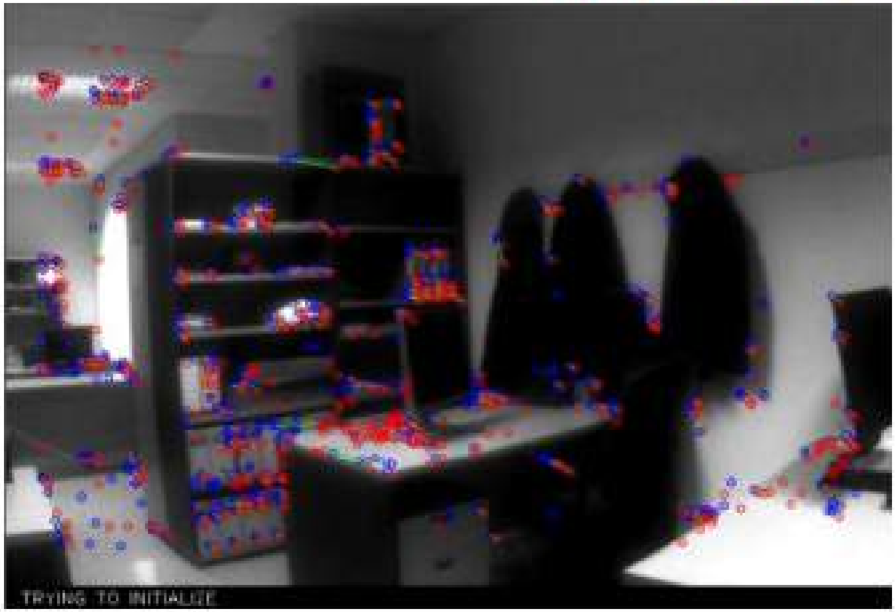
Popular Features for Visual SLAM

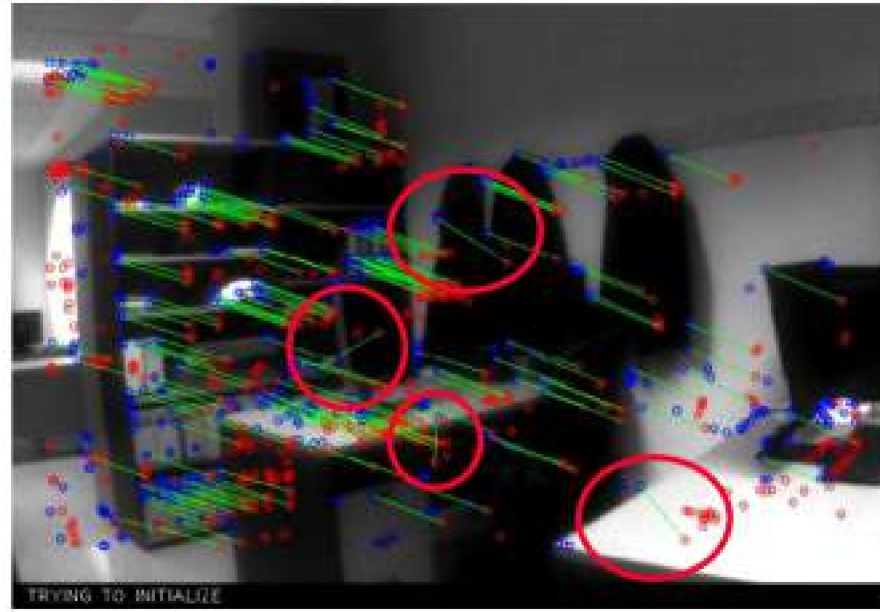
Detector	Descriptor	Rotation Invariant	Automatic Scale	Accuracy	Relocation & Loops	Efficiency
Harris	Patch	No	No	++++	-	++++
Shi-Tomasi	Patch	No	No	++++	-	++++
SIFT	SIFT	Yes	Yes	++	++++	+
SURF	SURF	Yes	Yes	++	++++	++
FAST	BRIEF	No	No	+++	+++	++++
ORB	ORB	Yes	No	+++	+++	++++

- ORB: Oriented FAST and Rotated Brief
  - 256-bit binary descriptor
  - Fast to extract and match (Hamming distance)
  - Good for tracking, relocation and Loop detection
  - Multi-scale detection → same point appears on several scales

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G.
ORB: an efficient alternative to SIFT or SURF, ICCV 2011

3. Feature Matching

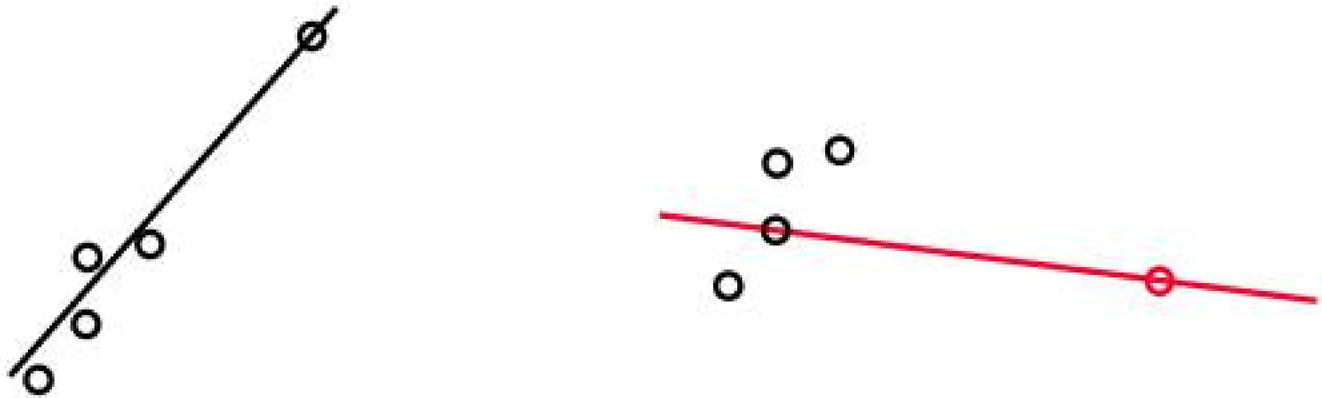




- Compare descriptors
- Spurious matchings
- Search for consensus with a robust technique: RANSAC

The problem of spurious matchings

- Least-squares is very sensitive to spurious data
- A single spurious match may to ruin the estimation
- Leverage point:

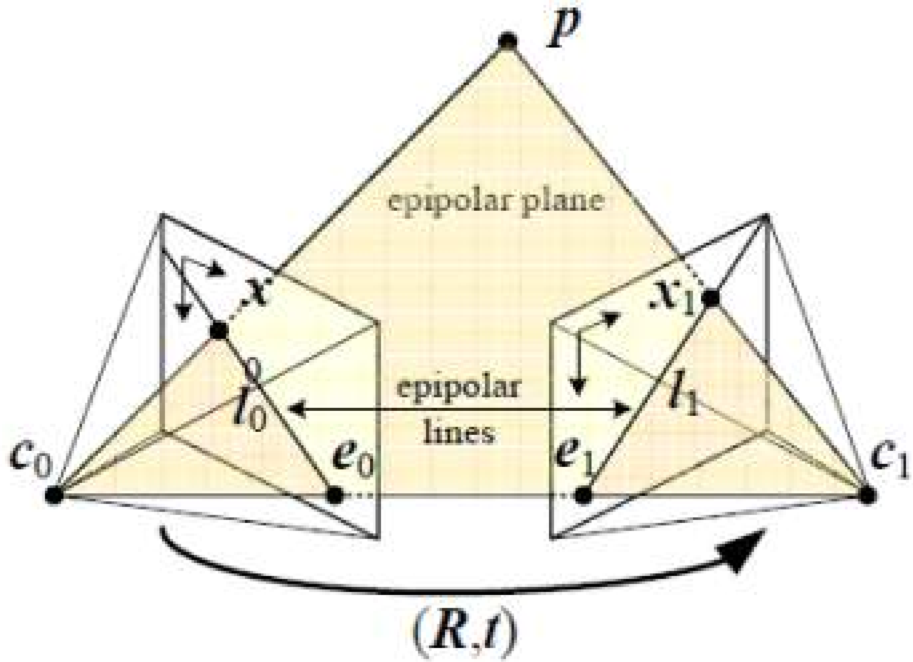


  - Removing the points with higher residuals DOES NOT SOLVE THE PROBLEM

RANSAC: RANdom SAmpling Consensus

```
RANSAC (P) return M and S
-- P: set of potential matches
-- M: alignment model found (requires at least k matchings)
-- S: set of supporting matches
for i = 1..max_attempts
      Si ← choose randomly k matchings from P
      Mi ← compute alignment model from Si
      Si* ← matchings in P that agree with Mi  (with tolerance ε )
      if  #(Si*) > consensus_threshold
         Mi* ← compute alignment model from Si*  (using least squares)
         return Mi* and Si*
      end if
endfor
return failure
```

Two View Model: Epipolar Constraint



- Vectors **$\mathbf{t} = \mathbf{c}_1 - \mathbf{c}_0$, $\mathbf{p} - \mathbf{c}_0$, $\mathbf{p} - \mathbf{c}_1$** must be coplanar

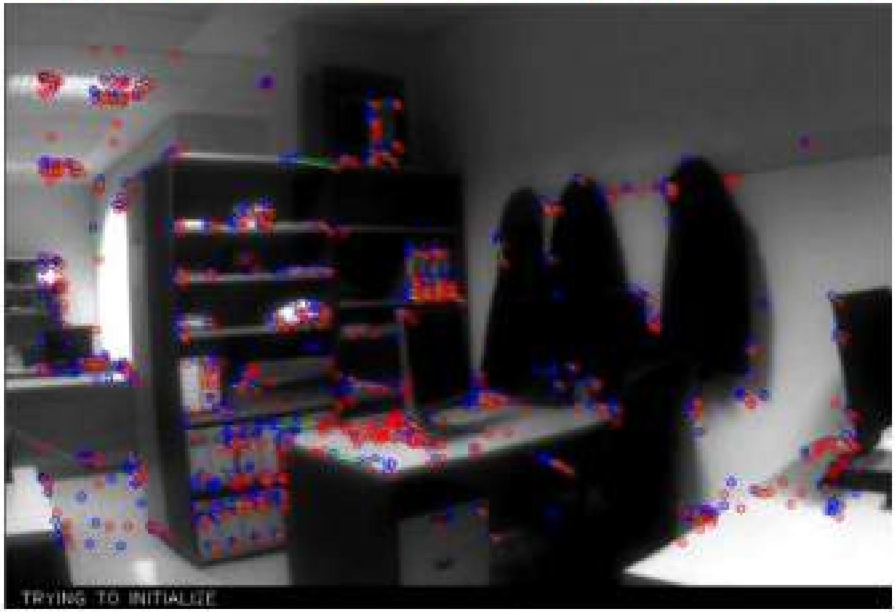
- Epipolar constraint: $$\mathbf{x}_{c1}^T \mathbf{E}\, \mathbf{x}_{c0} = 0$$

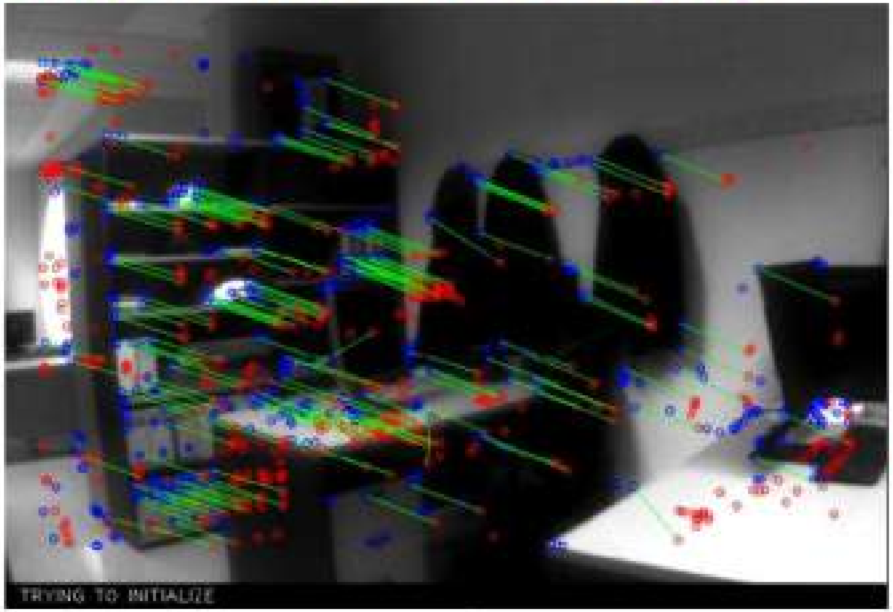
- Essential Matrix: $$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \mathbf{R}$$

Matching Problems

Problem	Inputs	Model to find	Basic Equation	d.o.f.	Min. # of matches	Minimal solution
Camera Location	$\mathbf{u}_{ij}, \mathbf{x}_{wj}$	Pose $\mathbf{T}_{iw}$	$\pi_i(\mathbf{T}_{iw}, \mathbf{x}_{wj})$	6	3	p3p
Initialize 3D scene	$\mathbf{u}_{1j}, \mathbf{u}_{2j}$	Essential Matrix $\mathbf{E}_{12} = [\mathbf{t}]_\times \mathbf{R}$	$\mathbf{u}_{1j}^T \mathbf{E}_{12} \mathbf{u}_{2j} = 0$	5	5	5-point 8-point
Initialize 2D scene	$\mathbf{u}_{1j}, \mathbf{u}_{2j}$	Homography $\mathbf{H}_{12}$	$\mathbf{u}_{1j} = \mathbf{H}_{12} \mathbf{u}_{2j}$	8	4	

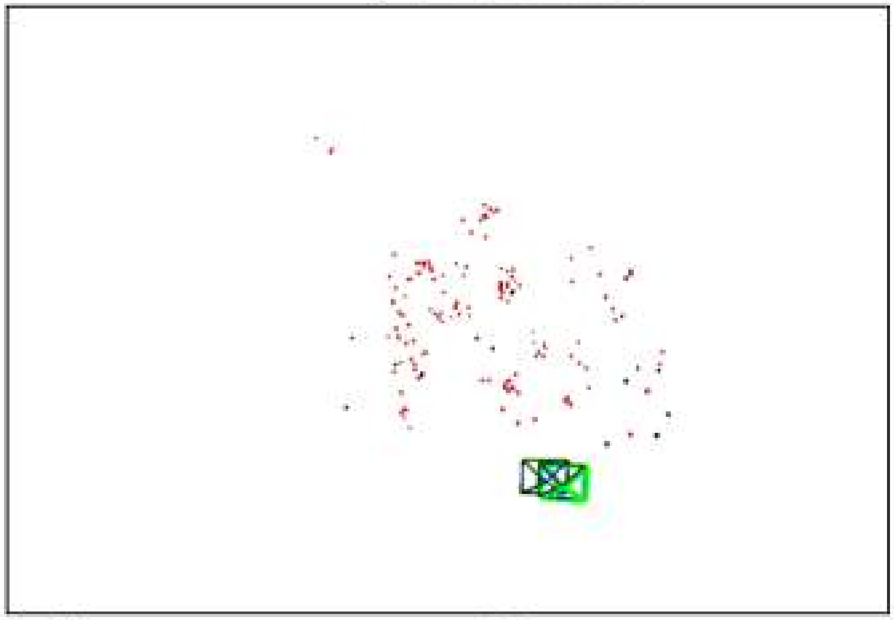
Matchings in 2 Frames → 3D Points and Motion





SFM:

- 5pt algorithm
- 8pt algorithm



Unknown Scale!

4. Relocation and Loop closing

- Relocation problem:

  During SLAM tracking can be lost: occlusions, low tecture, quick motions,…

  - Re-acquire camera pose and continue

- Loop closing problem

  SLAM is working, and you come back to a previously mapped area

  - Loop detection: to avoid map duplication
  - Loop correction: to compensate the accumulated drift

- In both cases you need a place recognition technique

Why is Loop Detection Difficult?

- Is this a loop closure?

Likely algorithm answer:

YES **YES** **TRUE POSITIVE**

Why is Loop Detection Difficult?

- Is this a loop closure?

Scene 1430

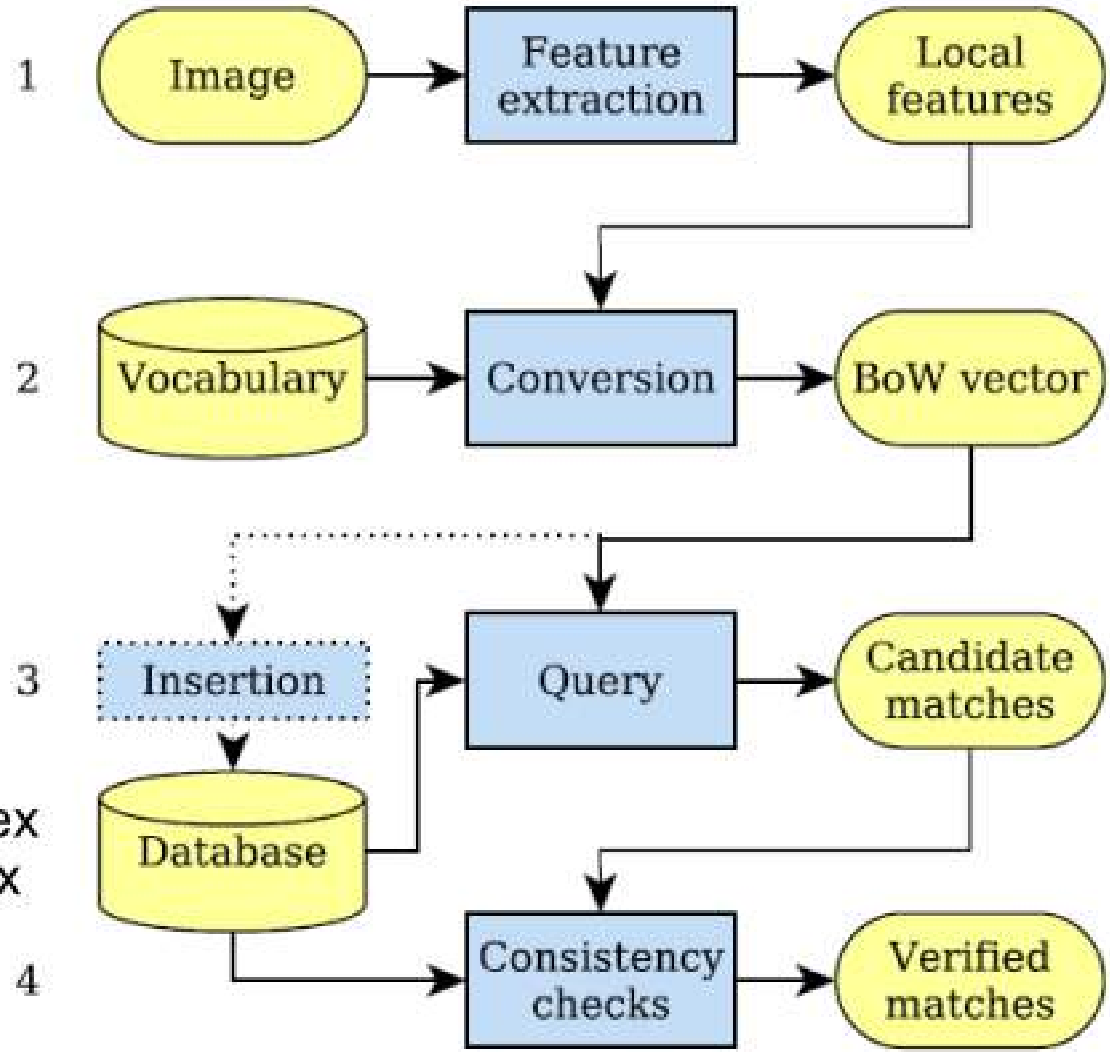
Scene 1244

Likely algorithm answer:

NO **YES** **FALSE POSITIVE**

Perceptual aliasing is common in indoor scenarios

Bag of Words Approach

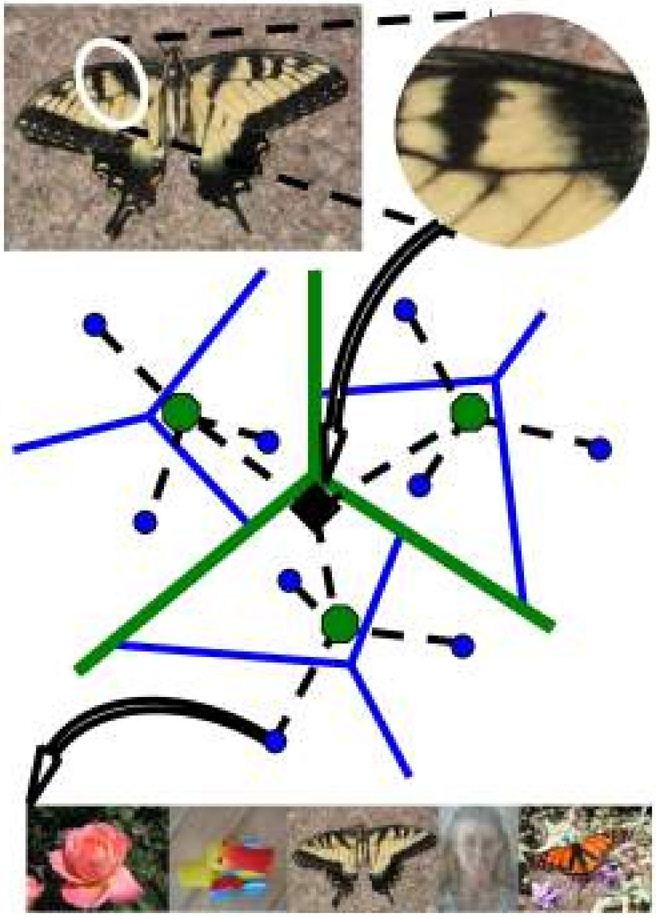


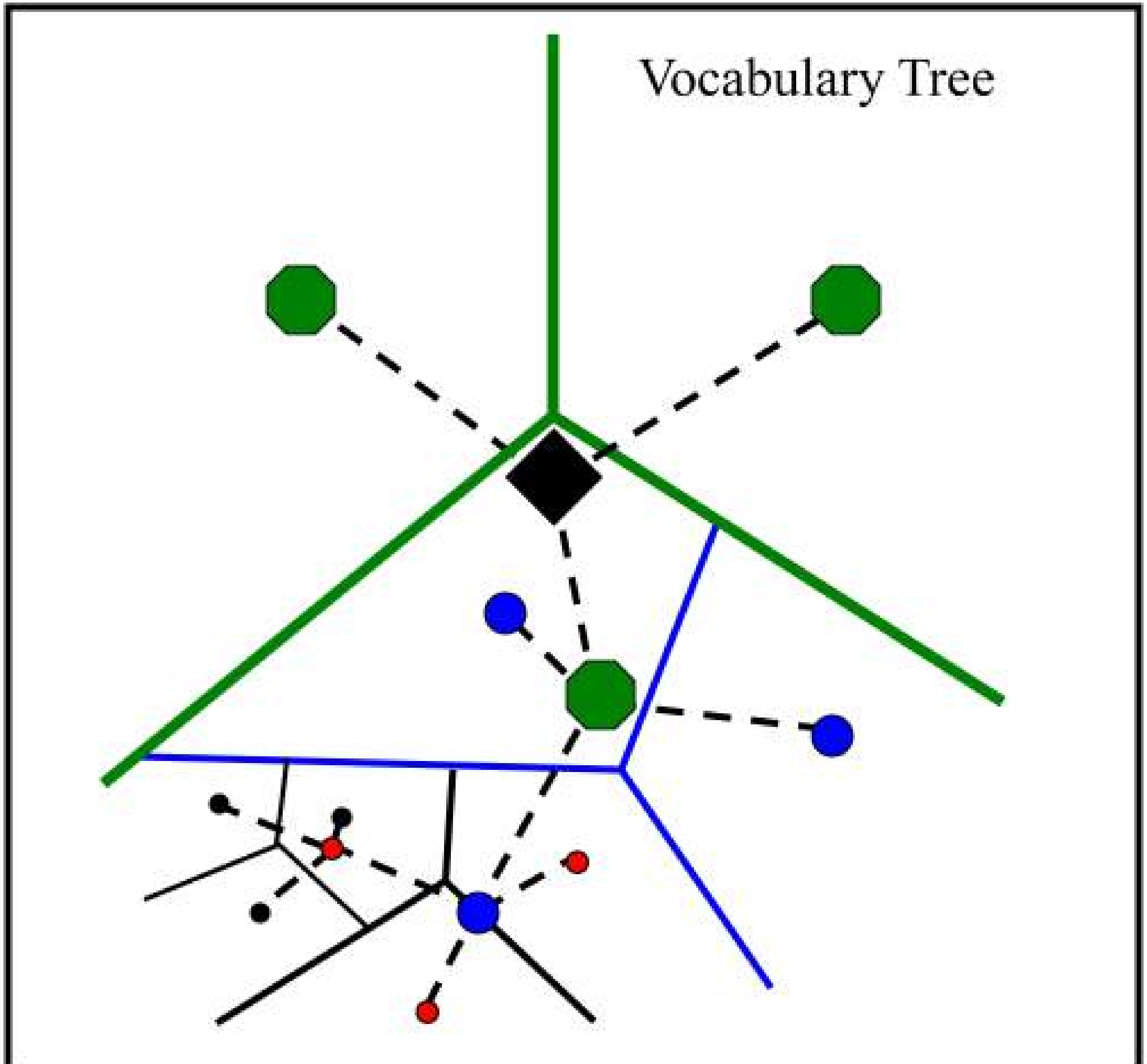
Binary Features
BRIEF, ORB

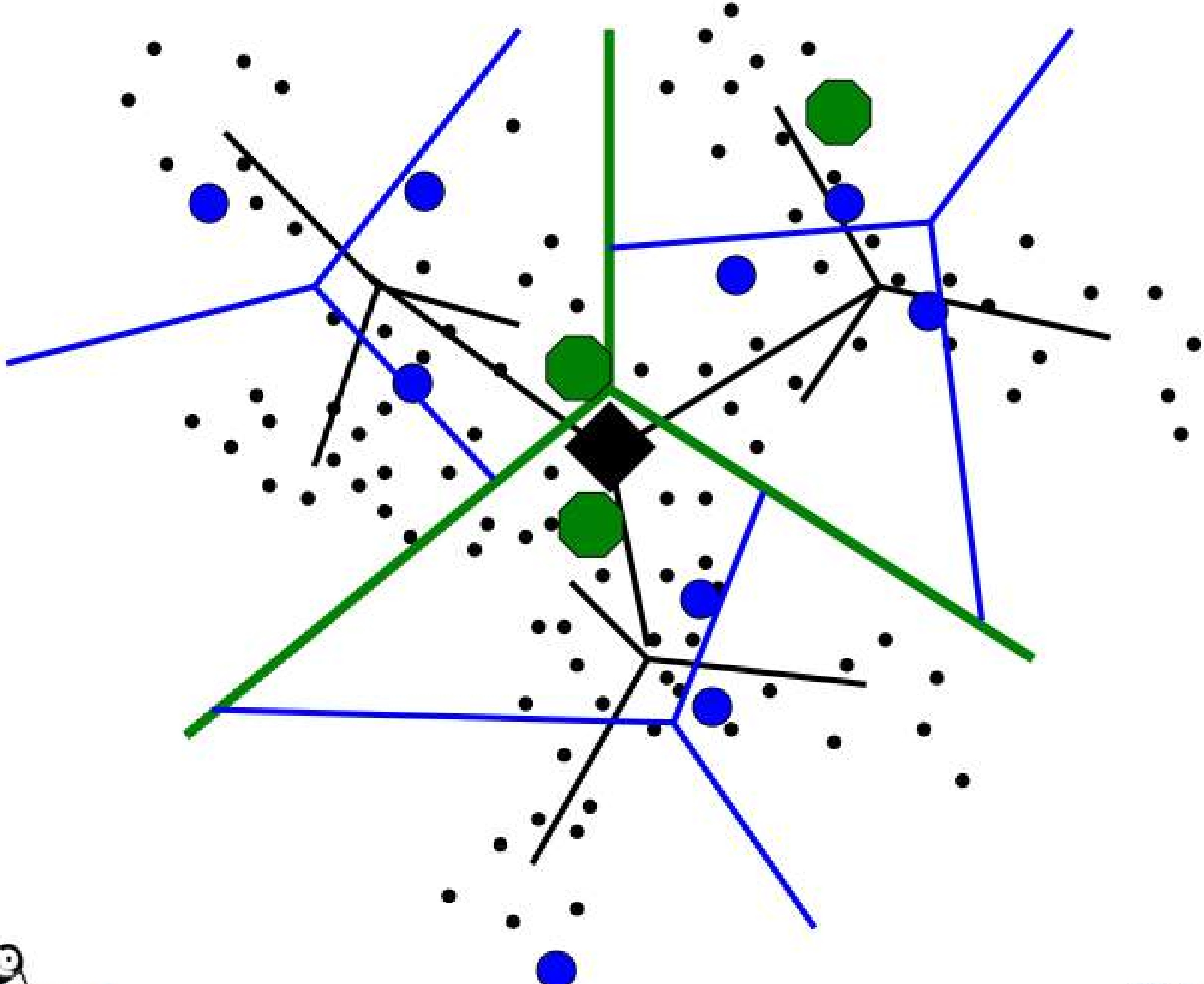
Inverse Index
Direct Index

Scalable Recognition with a Vocabulary Tree

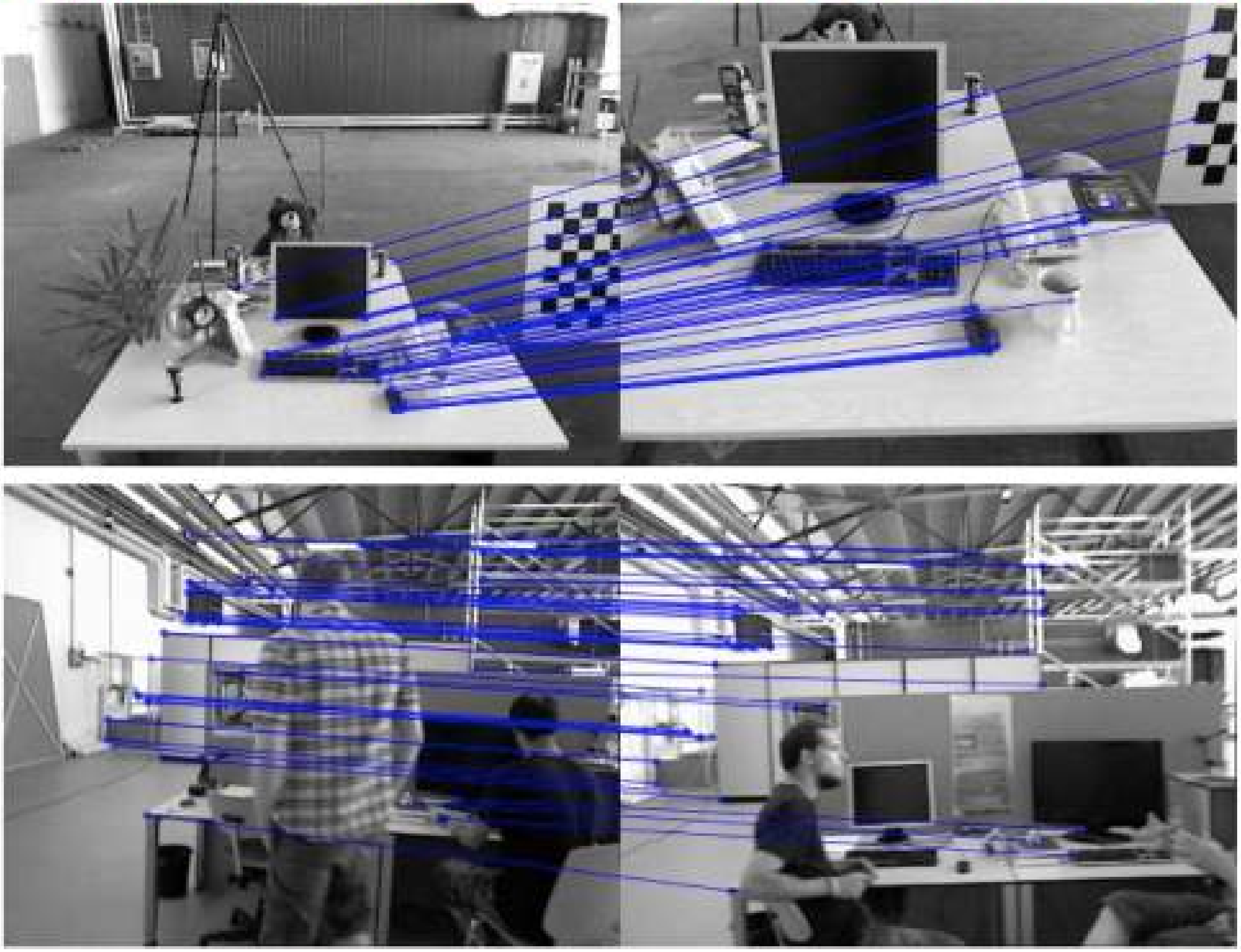
David Nistér, Henrik Stewénius
CVPR 2006




Vocabulary Tree

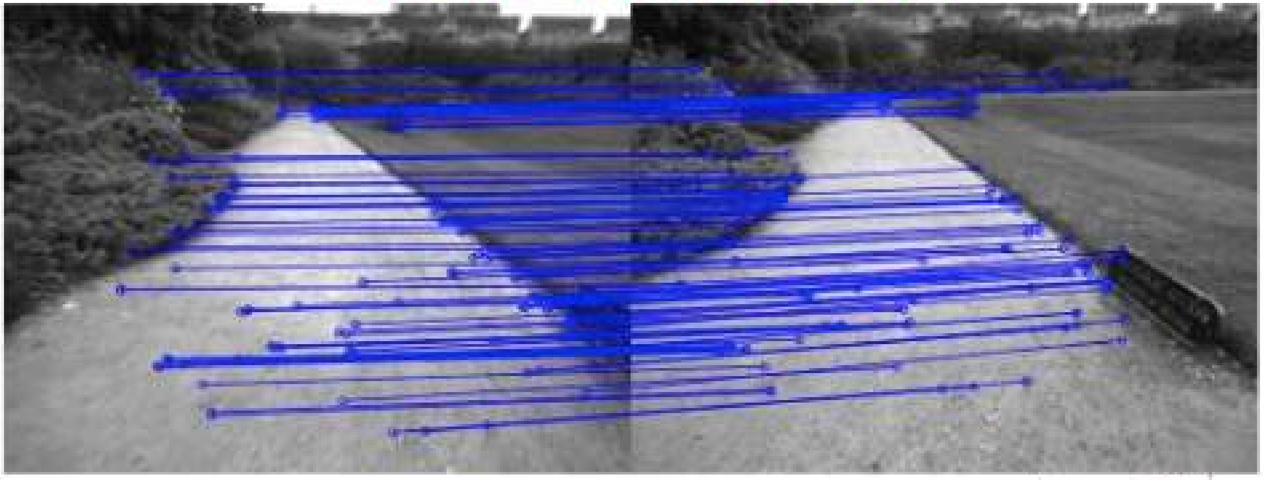


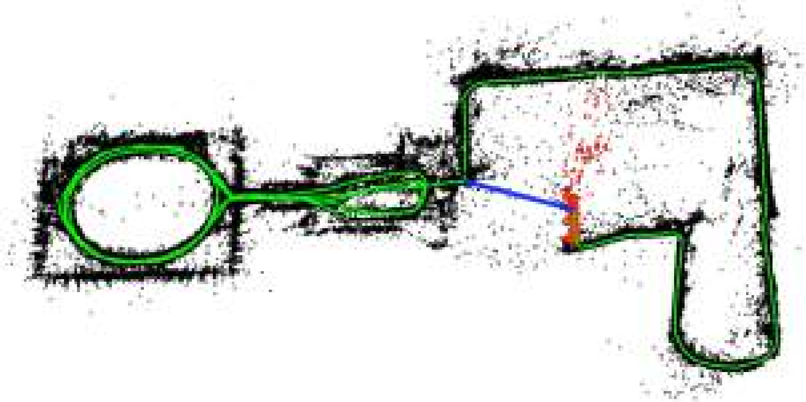
Examples with DBoW2 using ORB features

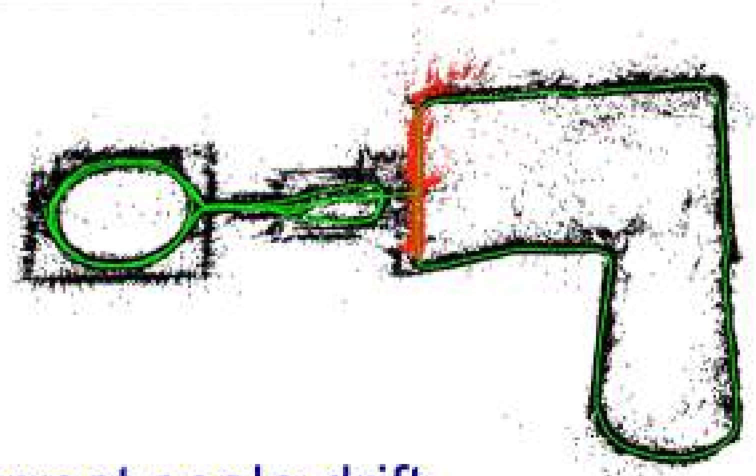


D. Gálvez-López, J.D. Tardós: *Bags of Binary Words for Fast Place Recognition in Image Sequences*, IEEE Trans. Robotics 28(5):1188-1197, 2012 (DBow2 software)

Loop Correction







- 7 Dof graph optimization, to correct scale drift
- And optionally Full BA (little improvement, much slower)

Outline

1. Basics: BA and Visual SLAM
2. Features
3. Feature Matching
4. Relocation and Loop Closing
5. Putting all Together
   - Example: ORB-SLAM

ORB-SLAM: Feature-Based SLAM, 2015

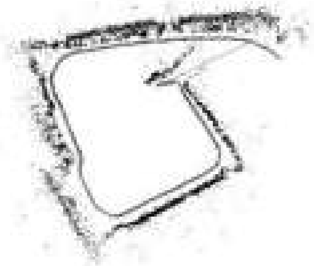
- Use the same features for:
  - Tracking
  - Mapping
  - Loop closing
  - Relocation
- ORB: FAST corner + Oriented Rotated Brief descriptor
  - Binary descriptor
  - Very fast to compute and compare
- Real-time, large scale operation
- Survival of the fittest for points and keyframes

Raúl Mur-Artal, José M. M. Montiel and Juan D. Tardós , **ORB-SLAM: A Versatile and Accurate Monocular SLAM System**, IEEE Trans. on Robotics 31(5): 1147-1163, Oct 2015 (software)

Recent Key Ideas

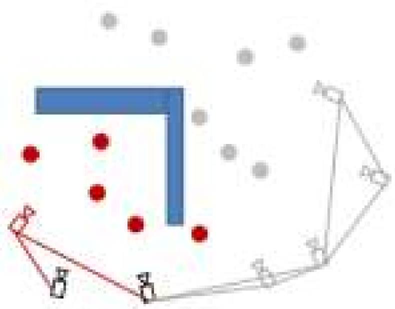
- Scale Drift-Aware Loop Closing

  H. Strasdat, J.M.M. Montiel and A.J. Davison
  Scale Drift-Aware Large Scale Monocular SLAM
  RSS 2010



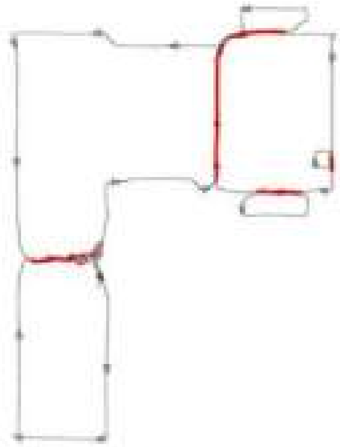
- Covisibility Graph

  H. Strasdat, A. J. Davison, J. M. M. Montiel , K. Konolige
  Double Window Optimization for Constant Time Visual SLAM
  ICCV 2011



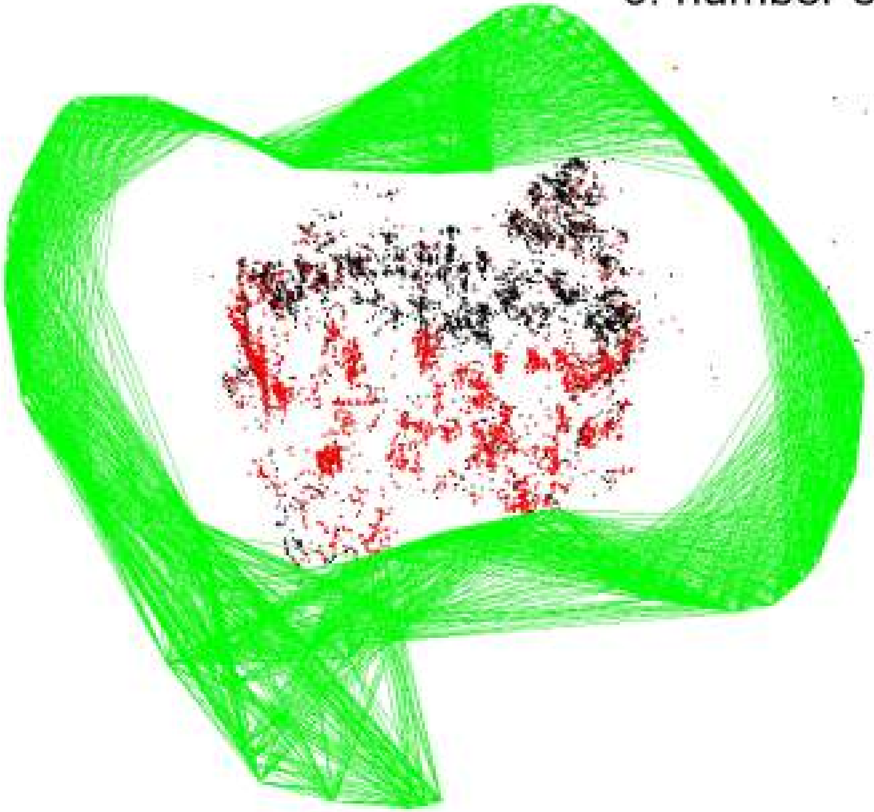
- Bags of Binary Words (DBoW)

  D. Gálvez-López and J. D. Tardós
  Bags of Binary Words for Fast Place Recognition in Image Sequences, IEEE Transactions on Robotics 2012



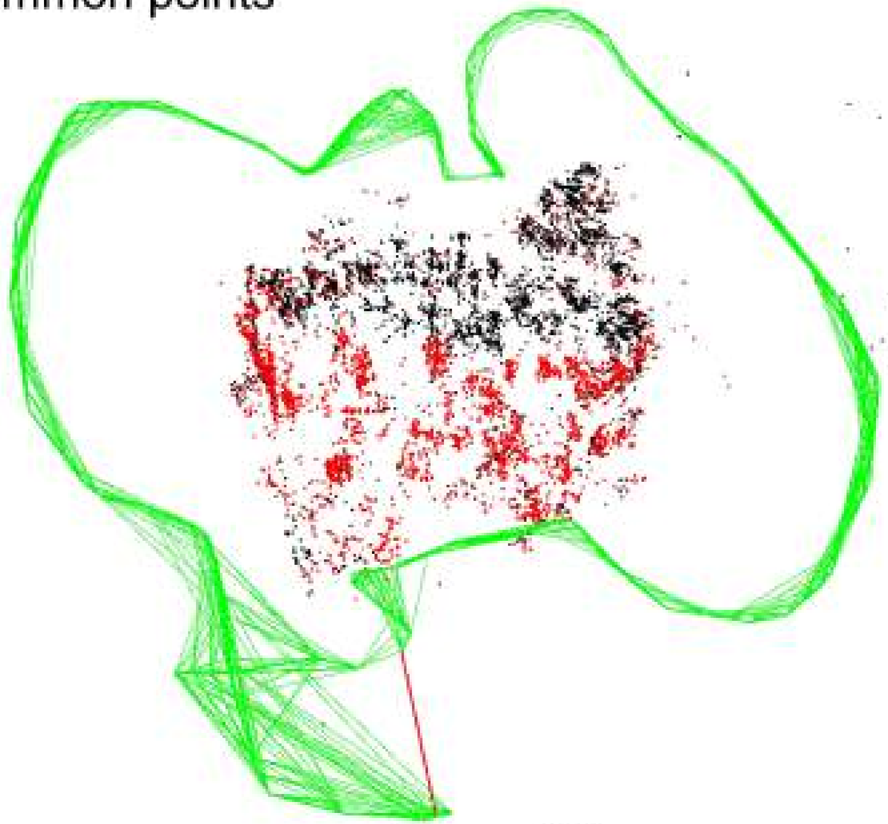
Covisibility Graph and Essential Graph

θ: number of common points



$\theta_{min} = 15$

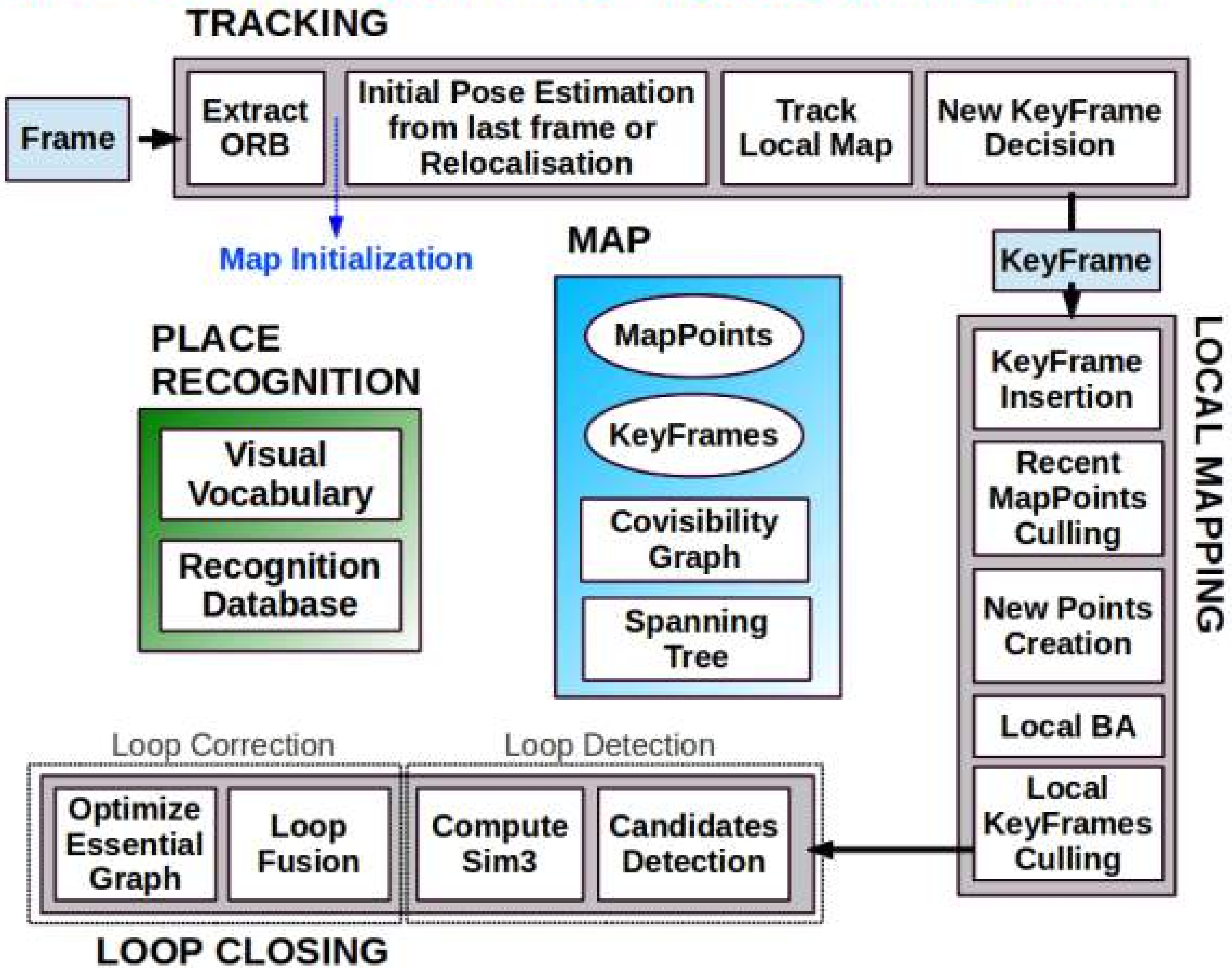
Used for Local BA



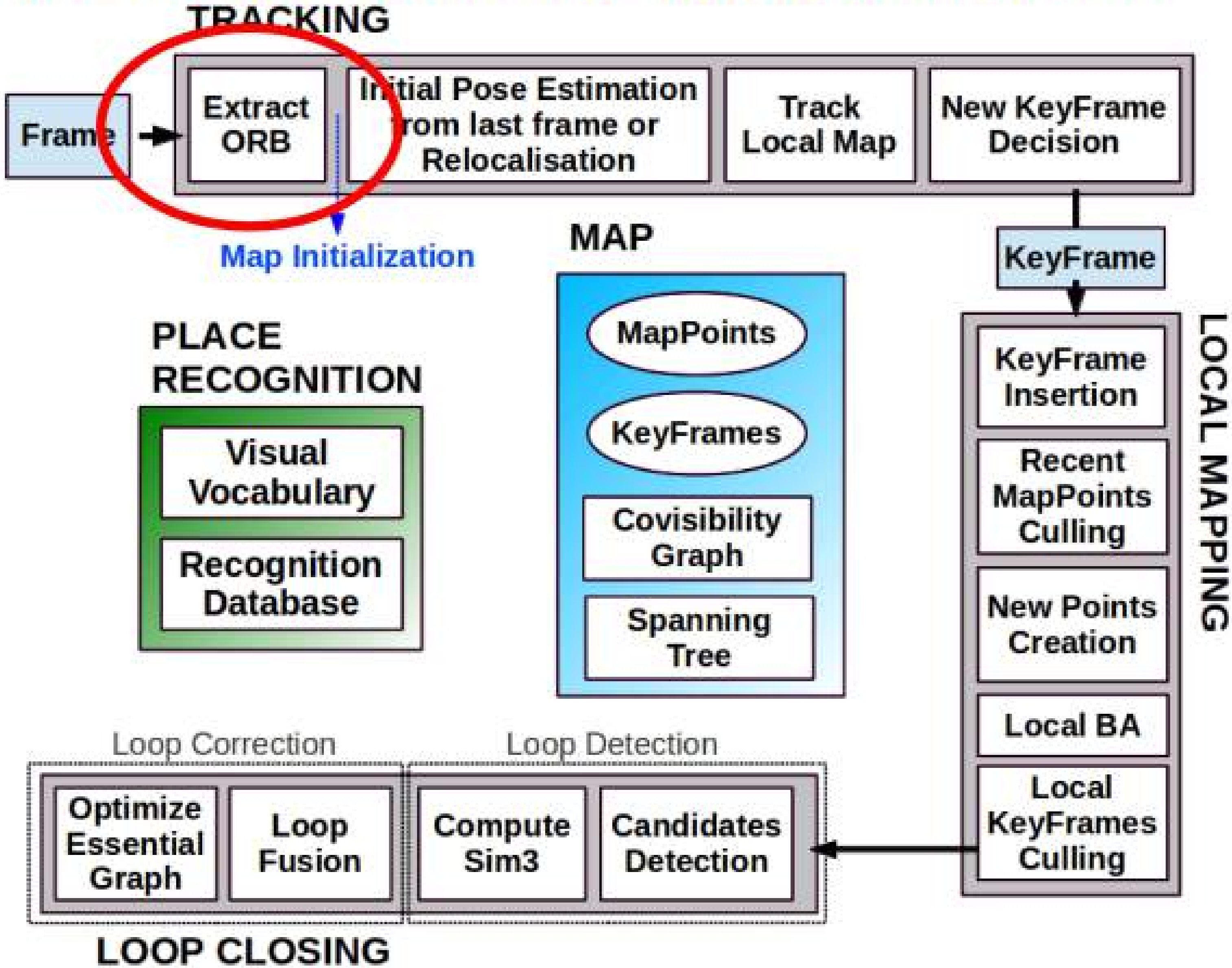
$\theta_{min} = 100$

Used for Loop Correction

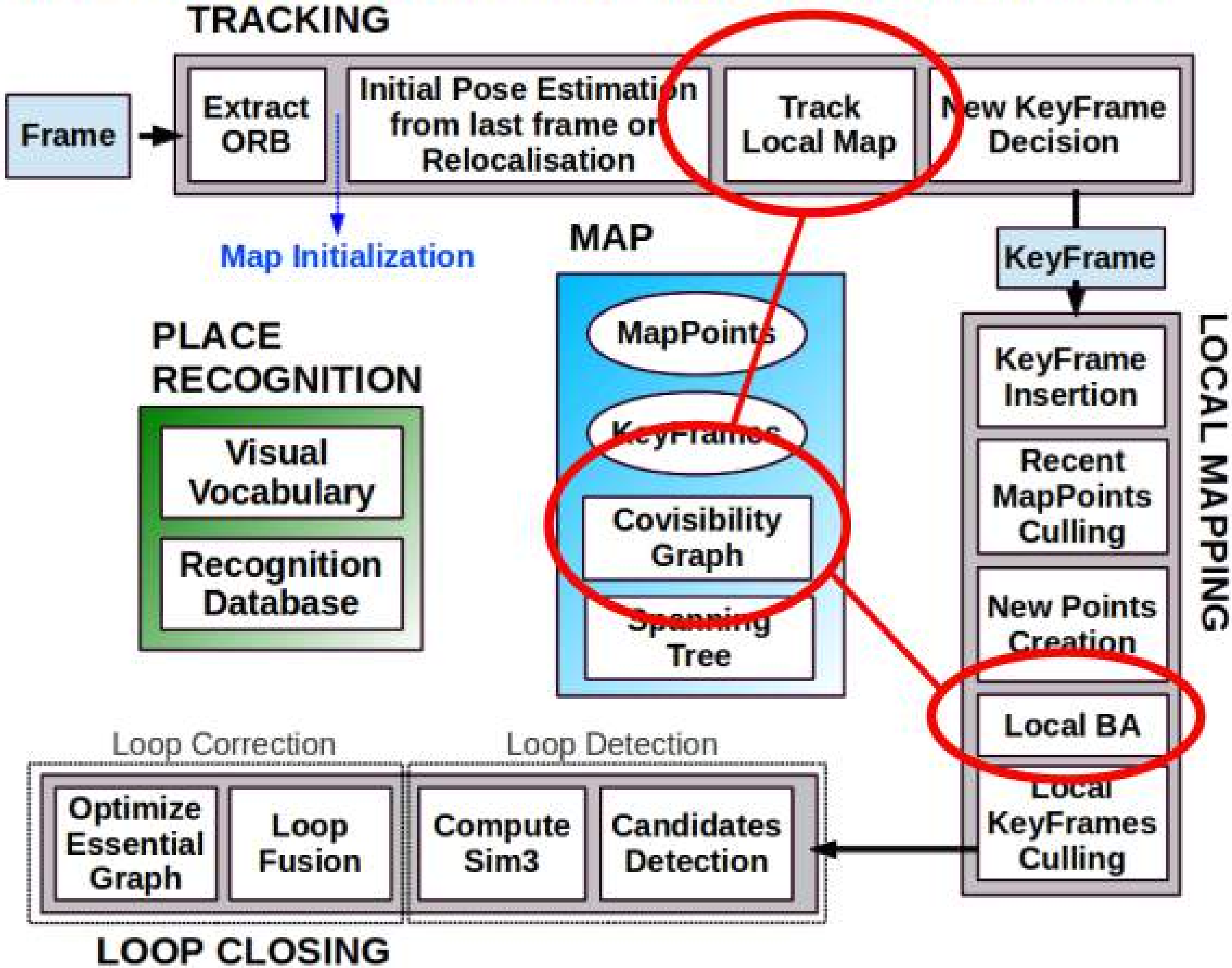
ORB-SLAM: Real-Time Monocular SLAM



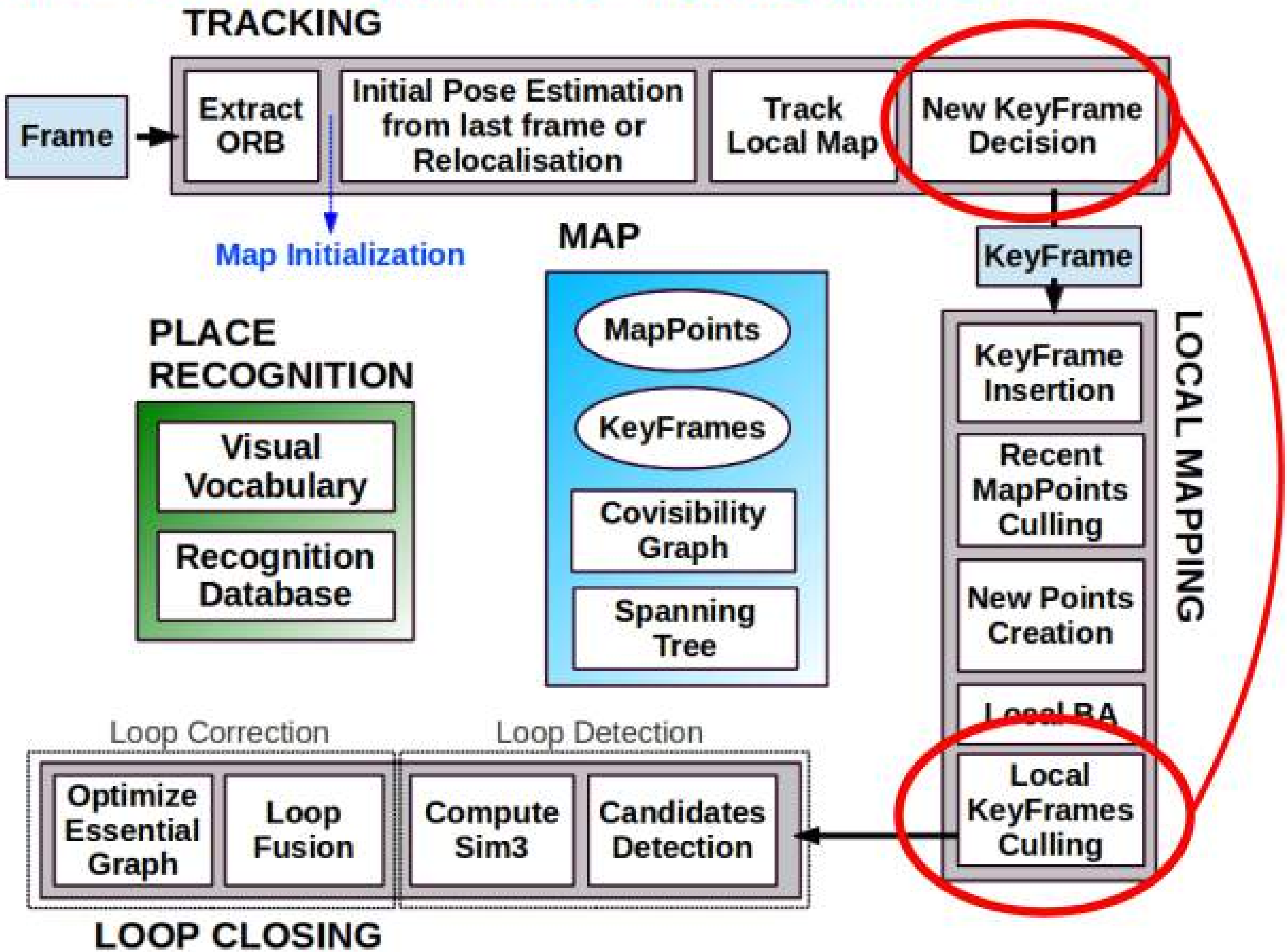
ORB-SLAM: Real-Time Monocular SLAM



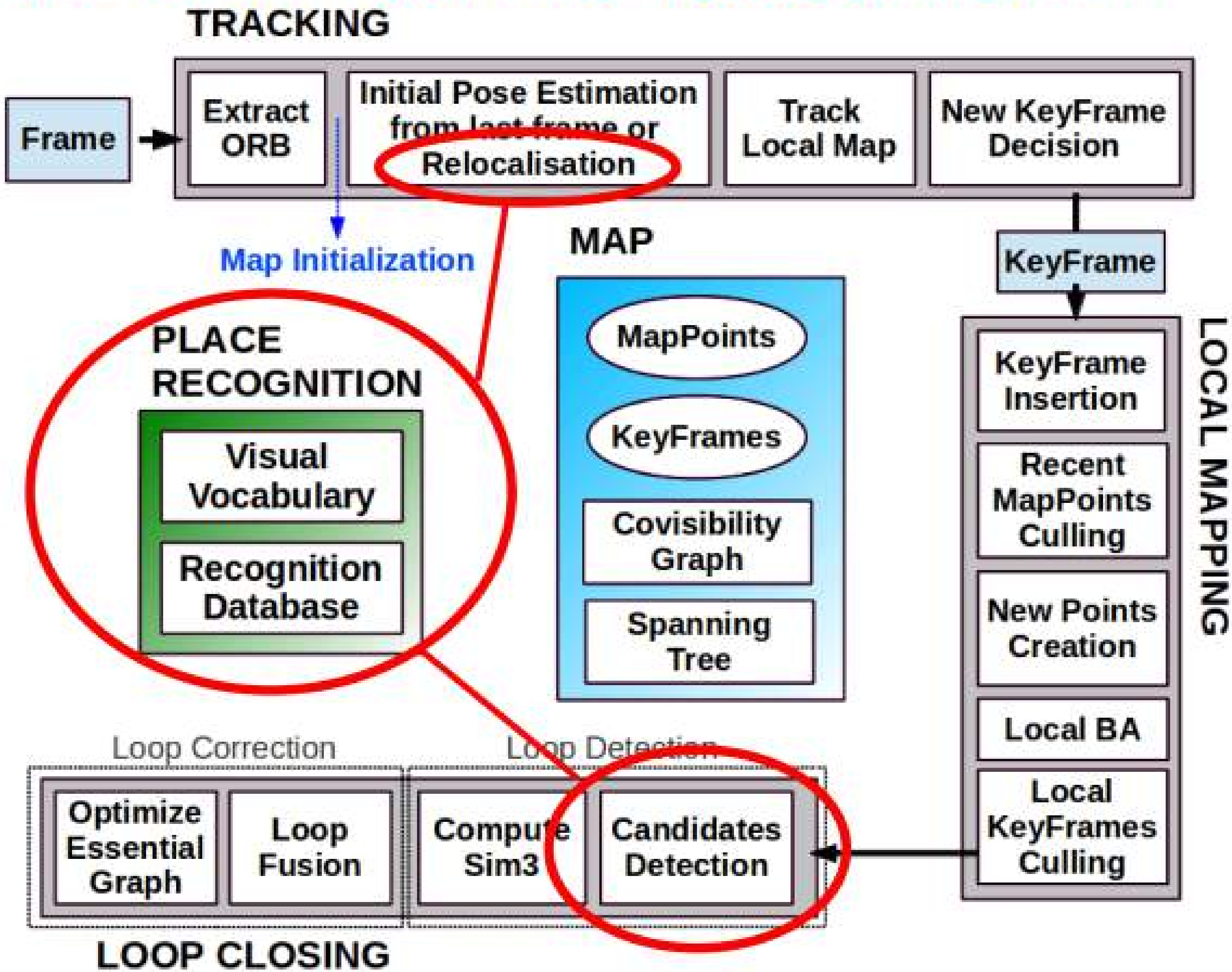
ORB-SLAM: Real-Time Monocular SLAM



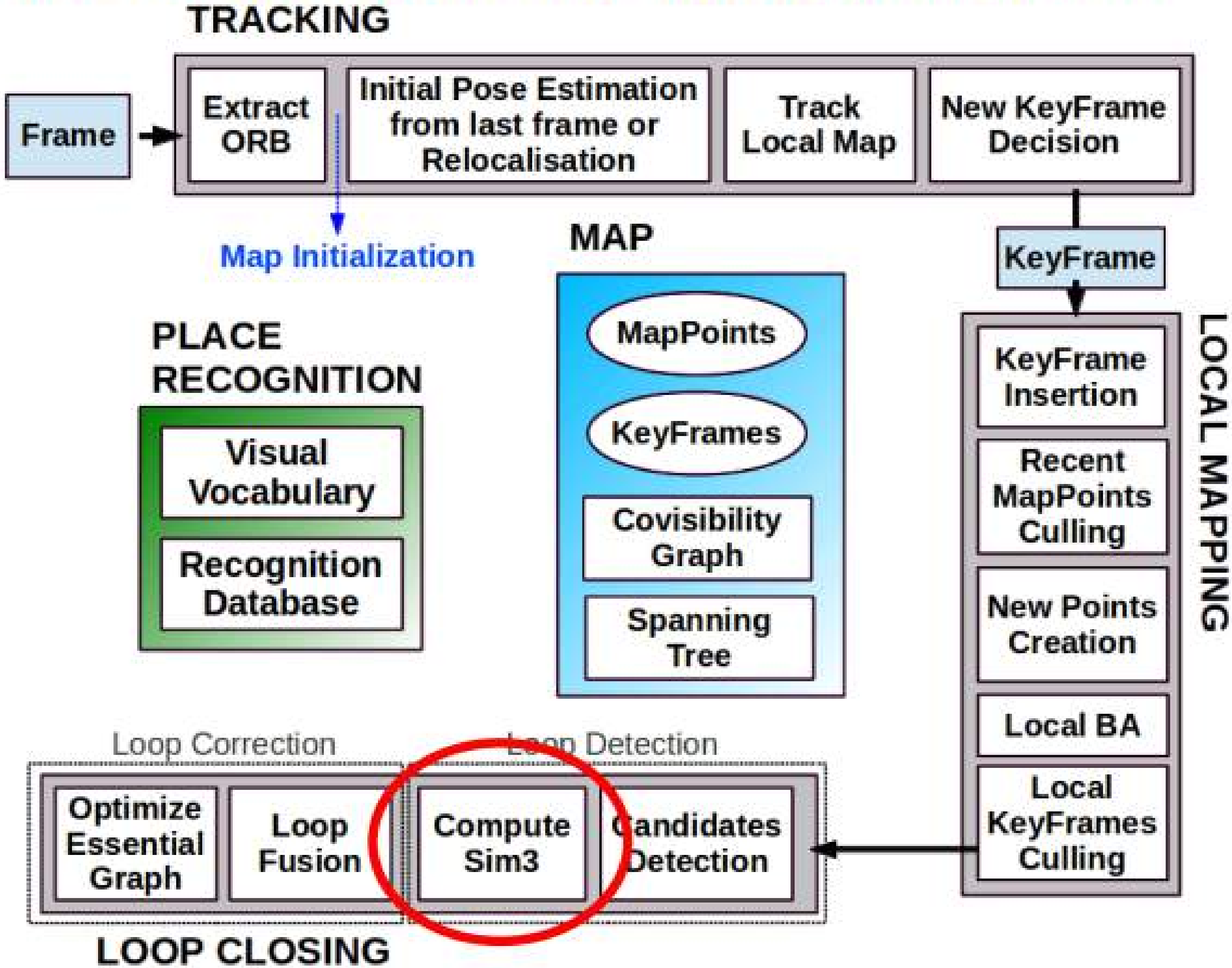
ORB-SLAM: Real-Time Monocular SLAM



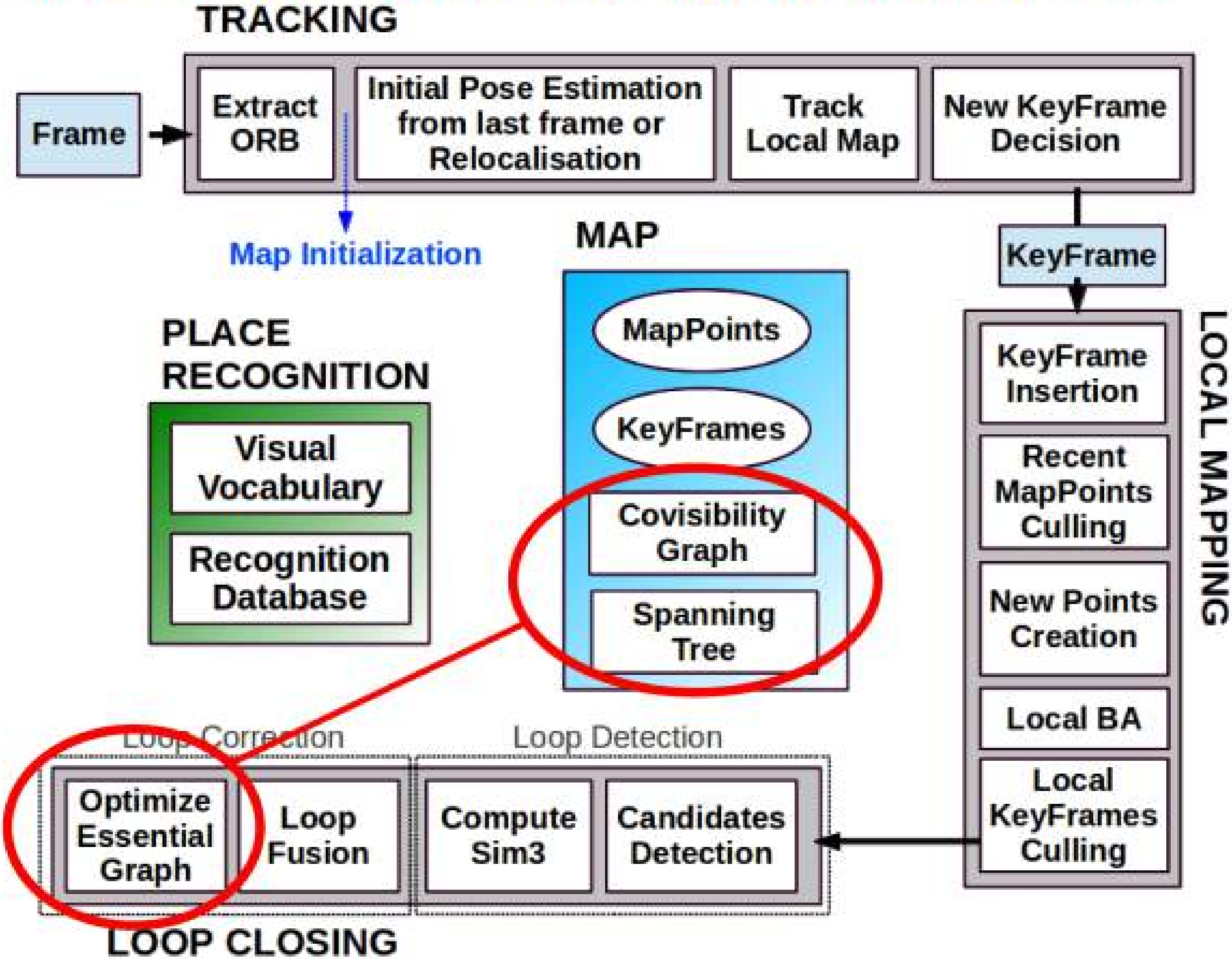
ORB-SLAM: Real-Time Monocular SLAM



ORB-SLAM: Real-Time Monocular SLAM



ORB-SLAM: Real-Time Monocular SLAM



Tracking: Fast KeyFrame Insertion

Survival of the Fittest
KeyFrame Selection

Fast Keyframe Insertion
(no distance threshold)

Culling of redundant
Keyframes

ORB-SLAM

PTAM

Relocation

Bags of Binary Words

Same ORBs used in Tracking and Mapping

Good Viewpoint Invariance (ORB)

ORB-SLAM

PTAM

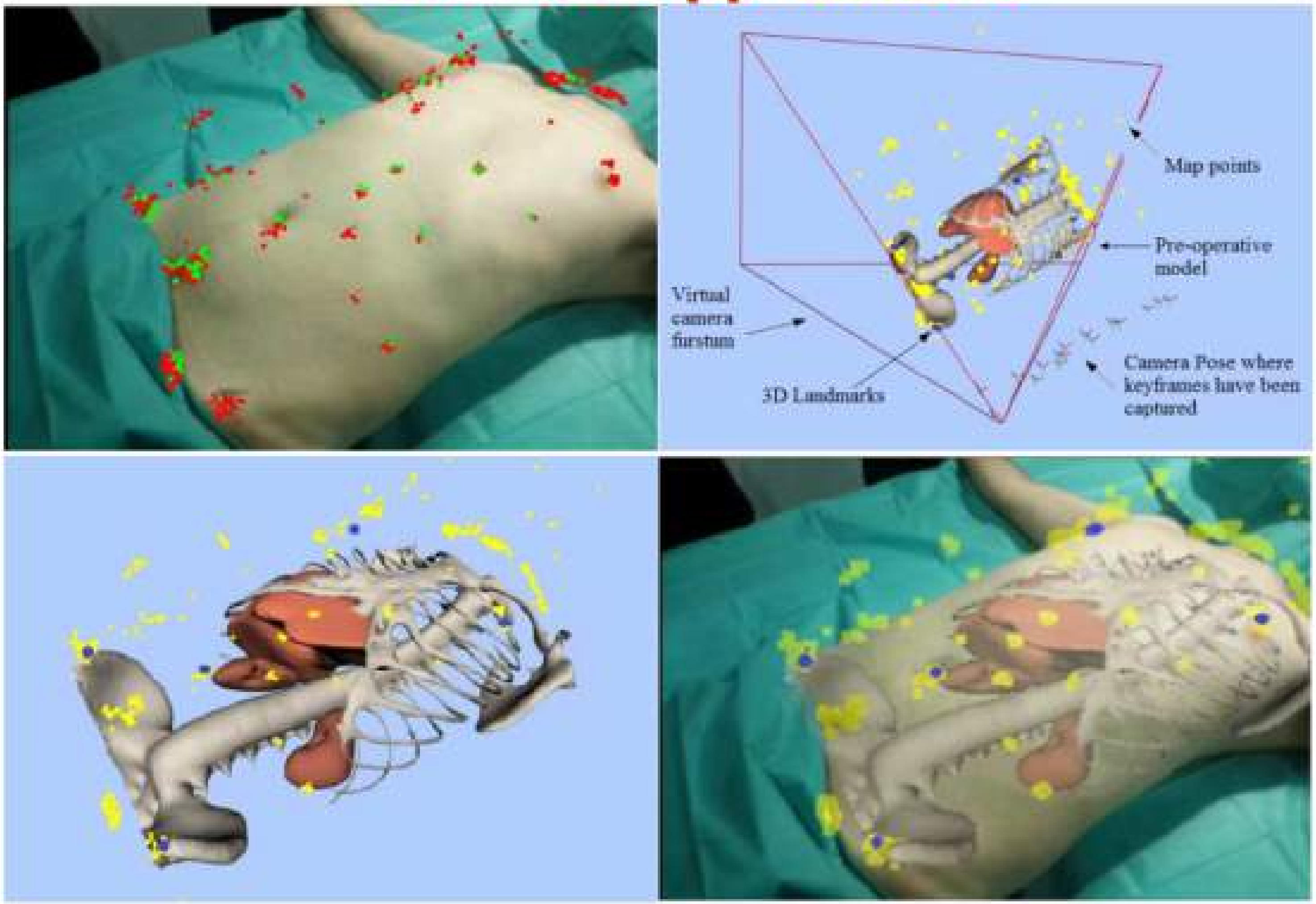
ORB-SLAM outdoors: Kitti Dataset

ORB-SLAM indoors: TUM RGB-D dataset

ORB-SLAM2 Stereo

ORB-SLAM2: Challenging Lighting

Medical Applications



N. Mahmoud et al. On-patient See-through Augmented Reality based on Visual SLAM, CARS 2016. Video

Inside the Body!

I. Cirauqui. Evaluación de ORBSLAM en Secuencias de Endoscopia Médica. TFG Grado Ingeniería Tecnologías Industriales. EINA, Mayo 2016.

Inside the Body!

I. Cirauqui. Evaluación de ORBSLAM en Secuencias de Endoscopia Médica. TFG Grado Ingeniería Tecnologías Industriales. EINA, Mayo 2016.

ORB-SLAM + Semi-Dense Mapping

Feature-Based SLAM

- Limitations
  - Monocular → the absolute scale is unknown
  - Requires a reasonably lit area
  - Needs texture: will fail with large plain walls
  - Map is too sparse for interaction with the environment
- Extensions
  - Improve agility using IMU
  - Stereo: real scale and more robust to quick motions
  - Semi-dense or dense mapping
- Further information about ORBSLAM:
  - http://webdiis.unizar.es/~raulmur/orbslam/
  - Source code available under GPLv3

The ORB-SLAM Team