ProSLAM: Graph SLAM from a Programmer's Perspective

author

January 7, 2019

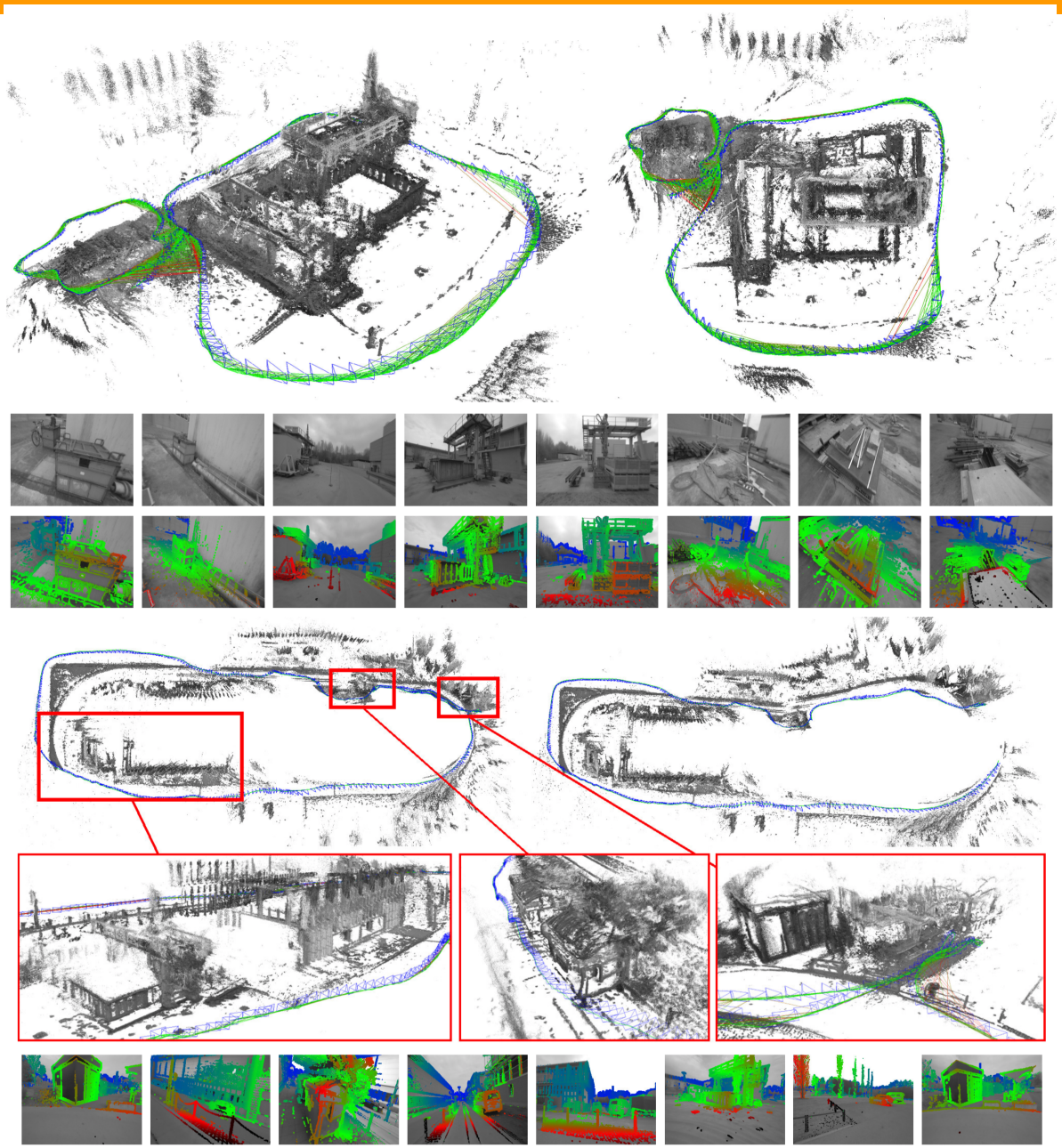
Outline

Introduction

What is SLAM ?

- **S**imultaneous
- **L**ocalization
- **A**nd
- **M**apping

History: LSD SLAM



Large-**S**cale **D**irect **SLAM**:

Age:

2014

Matching:

feature-*less*

Cameras:

monocular

State of the art: ORB SLAM 2

ORB SLAM 2:

Age
June
2017

Matching:
stereo keypoints
bundle **adjustment**

Cameras:
monocular
stereo
RGB-D

Problem: Other SLAM approaches

Issues:

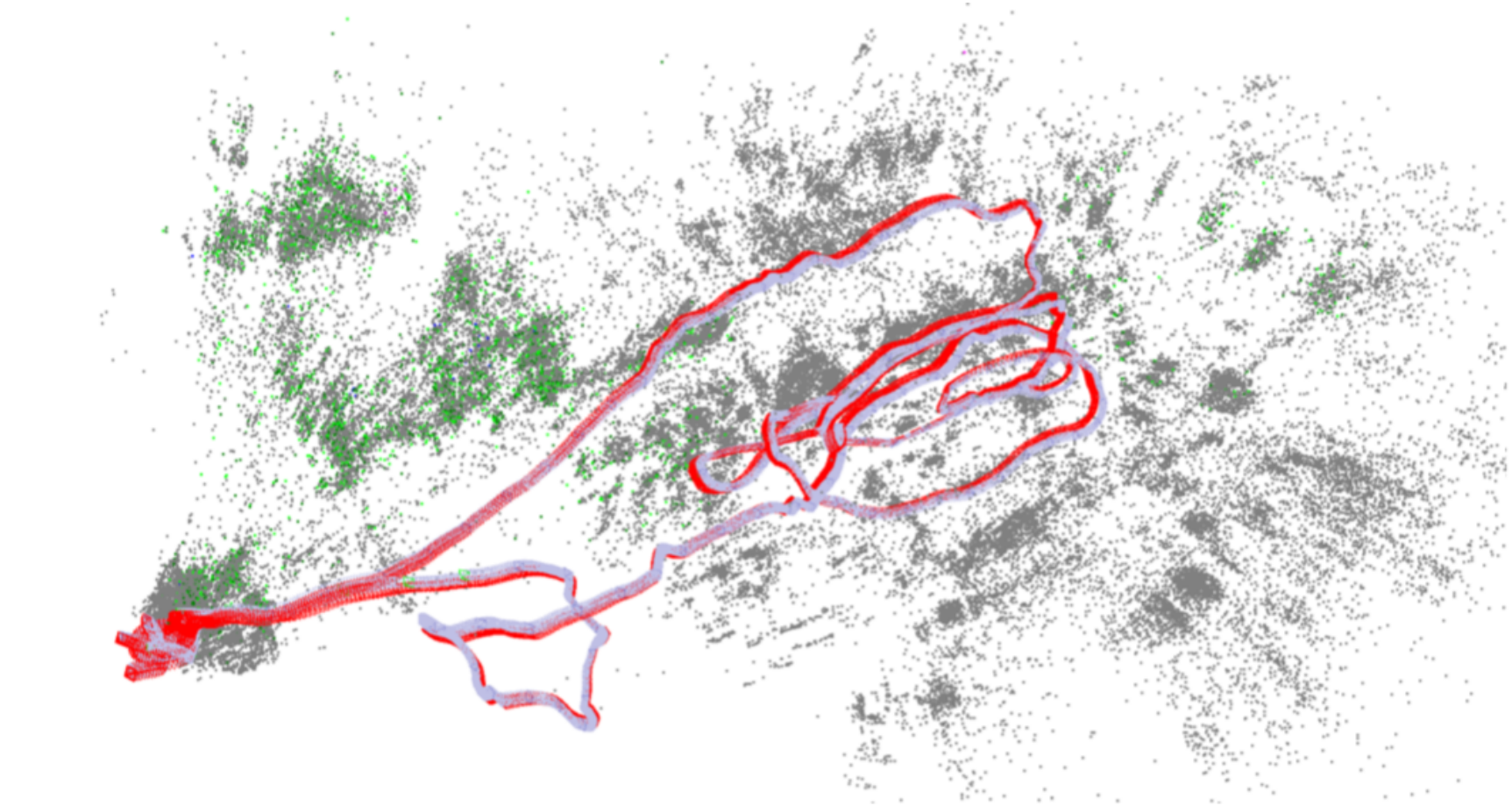
- Computationally **Heavy**
- High Theoretical **Complex**ity
- **How** to implement it in practice ?

$$E_p(\boldsymbol{\xi}_{ji}) = \sum_{\mathbf{p} \in \Omega_{D_i}} \left\| \frac{r_p^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma^2_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}} \right\|_\delta$$

$$r_p(\mathbf{p}, \boldsymbol{\xi}_{ji}) := I_i(\mathbf{p}) - I_j(\omega(\mathbf{p}, D_i(\mathbf{p}), \boldsymbol{\xi}_{ji}))$$

$$\sigma^2_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})} := 2\sigma_I^2 + \left(\frac{\partial r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\partial D_i(\mathbf{p})} \right)^2 V_i(\mathbf{p})$$

ProSLAM: Proposed solution



Age

September *2017*

Matching:

feature-based: **landmarks**

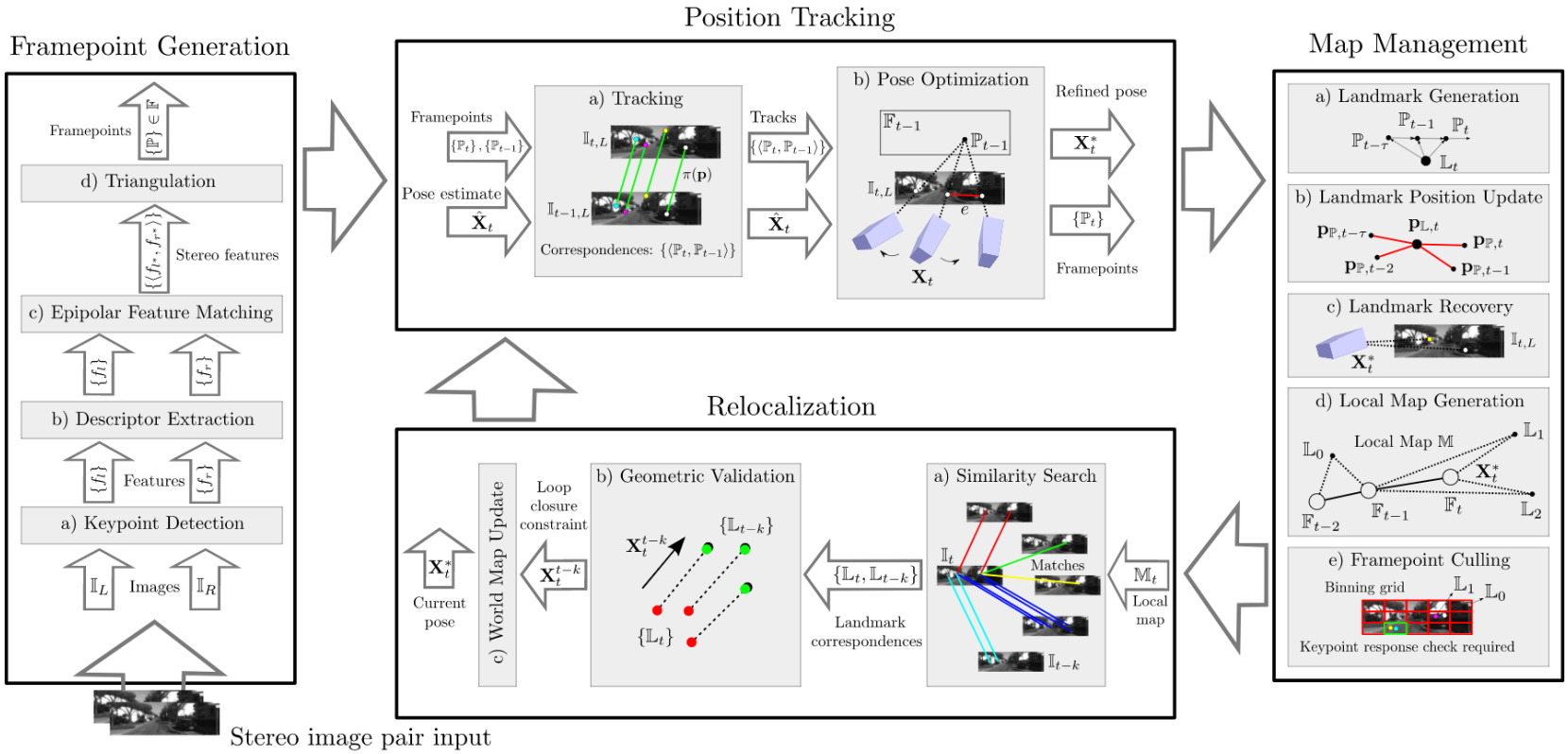
local bundle adjustment

Cameras:

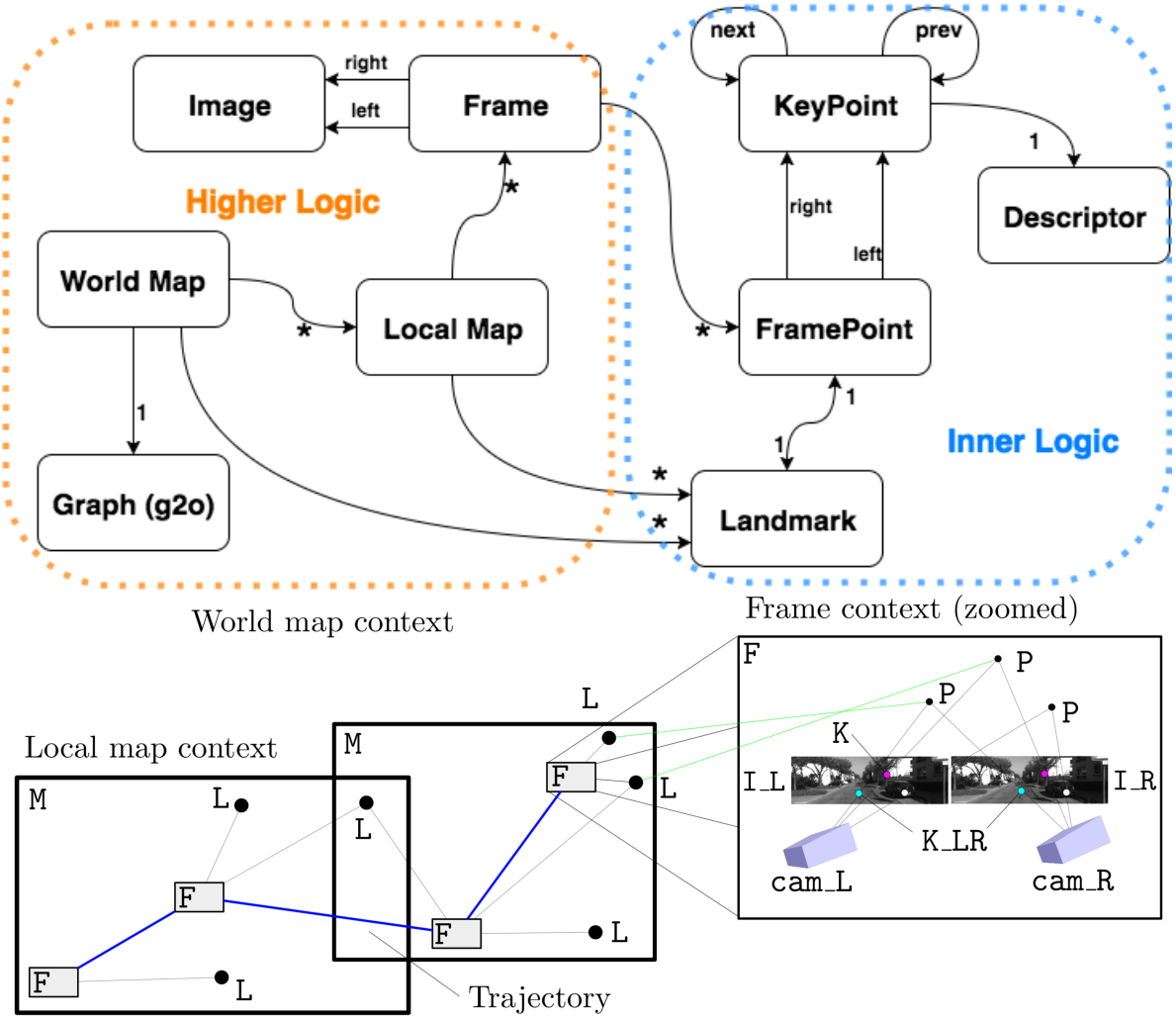
stereo

Algorithm

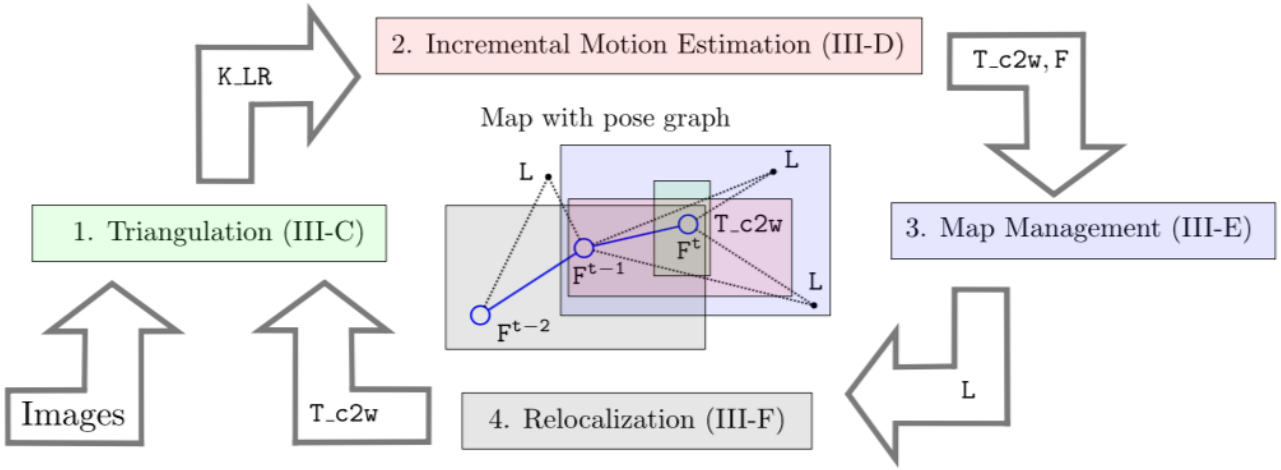
ProSLAM: From above

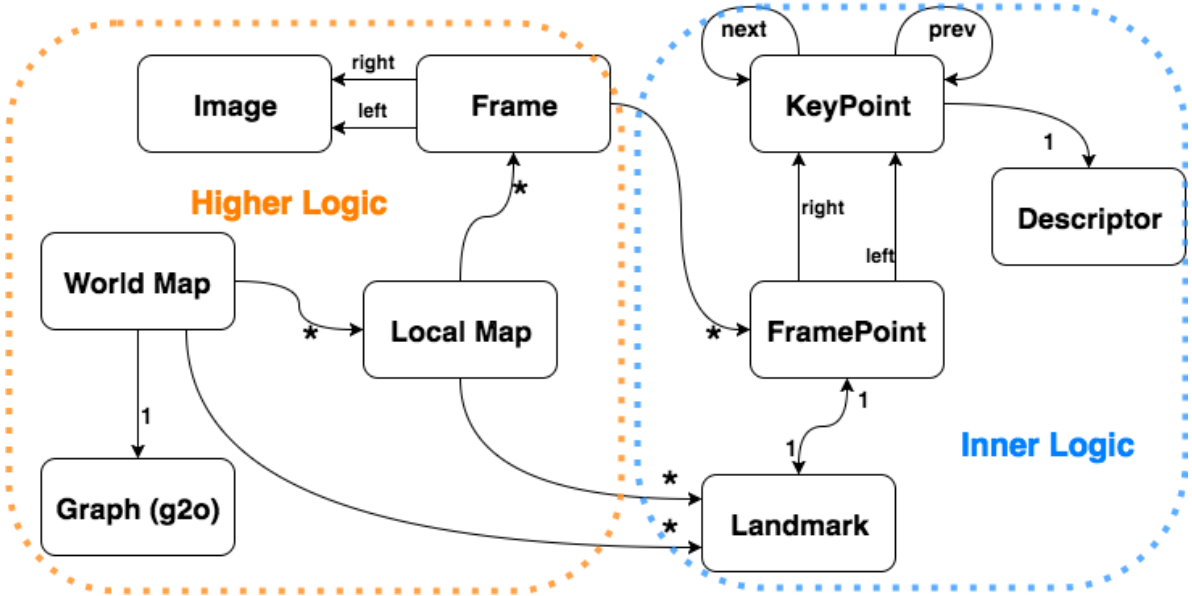


Data Structs: visual representation

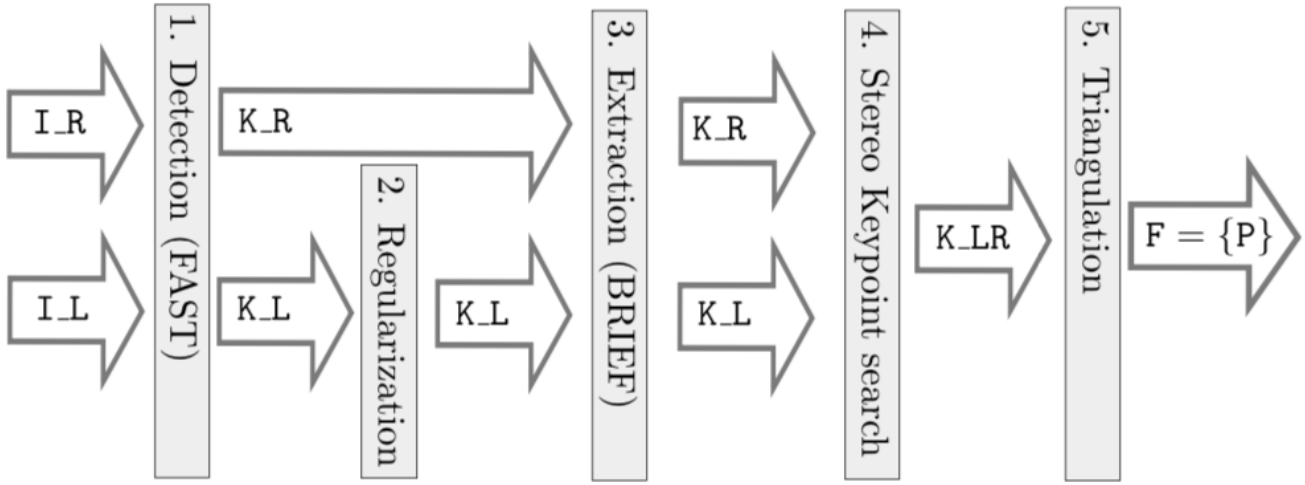


Pipeline: four phases





Triangulation: phase one



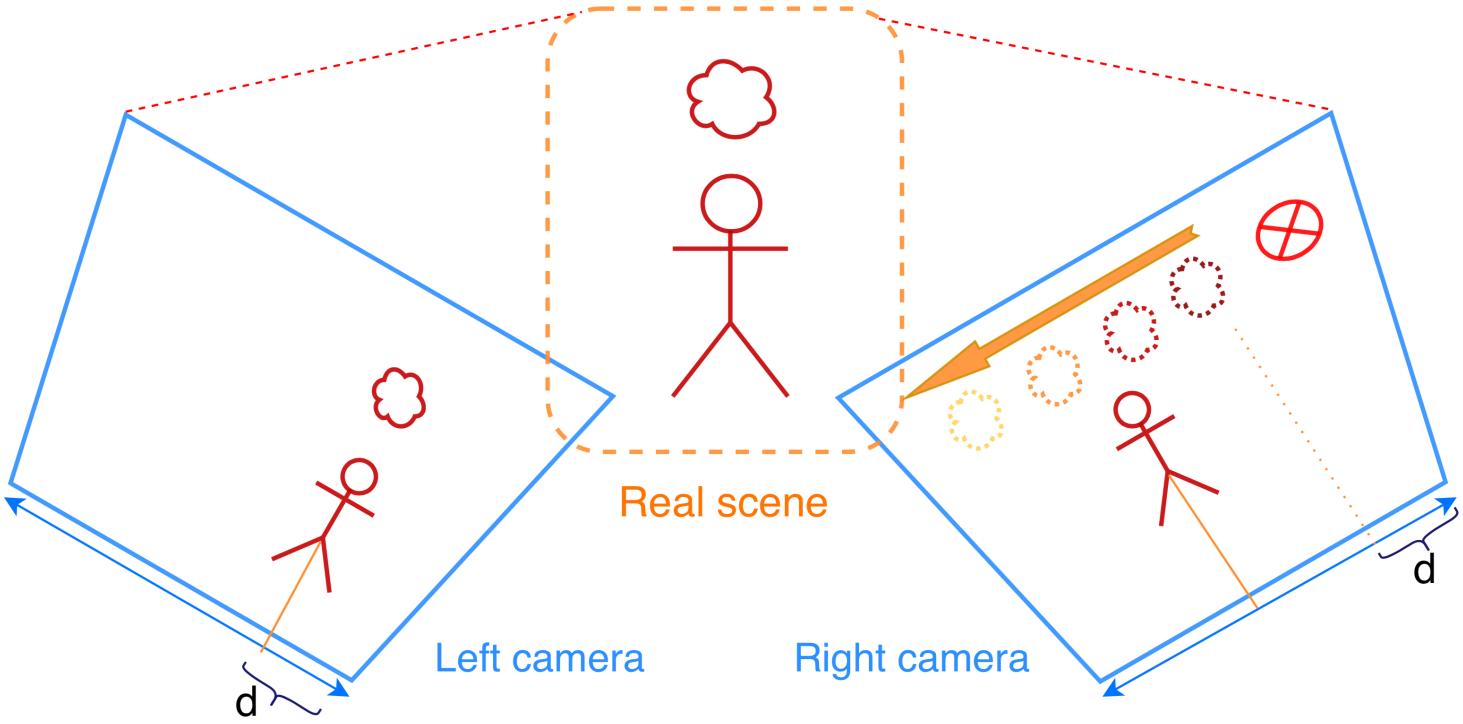
Framepoint coords:

$$p_z = \frac{\overbrace{B}^{\text{stereo camera baseline}}}{K_r.c - K_l.c} \quad (1)$$

$$p_x = \underbrace{\frac{p_z}{F_x}} K_l.c - C_x \quad (2)$$

$$p_y = \underbrace{\frac{p_z}{F_y}}_{\text{focal length}} K_l.r - C_y \quad (3)$$

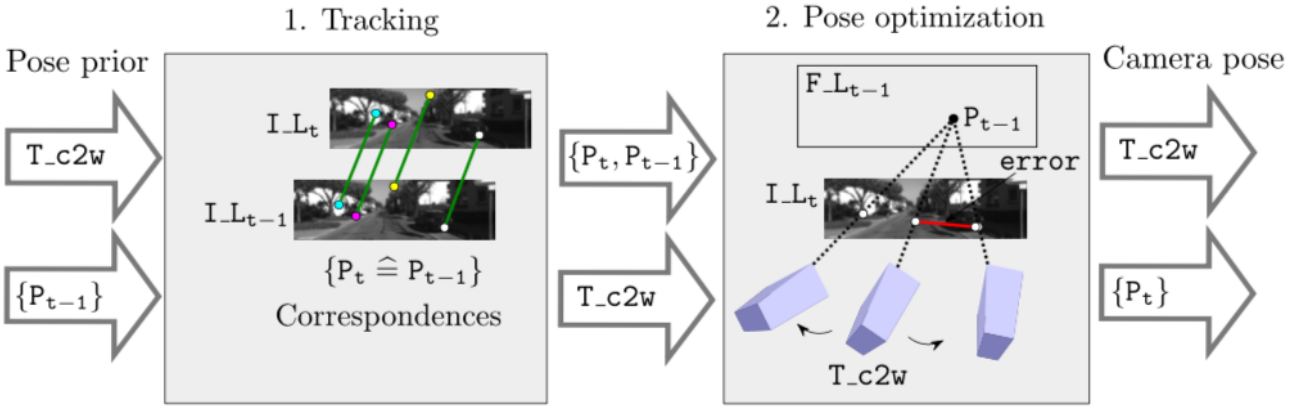
Triangulation : Intuition



Correspondence matching:

- Expected on the **left side** of the other image.
- Candidates **sorted** with euclidean distance metric.

Incremental Motion Estimation: phase two



$$\overbrace{K_P}^{\text{new keypoint}} = \pi(P_L * \underbrace{T_{w2c} * \overbrace{P_{t-1}}^{\text{framepoint}} . p_w}_{P_{t-1}.p_c}) \tag{4}$$

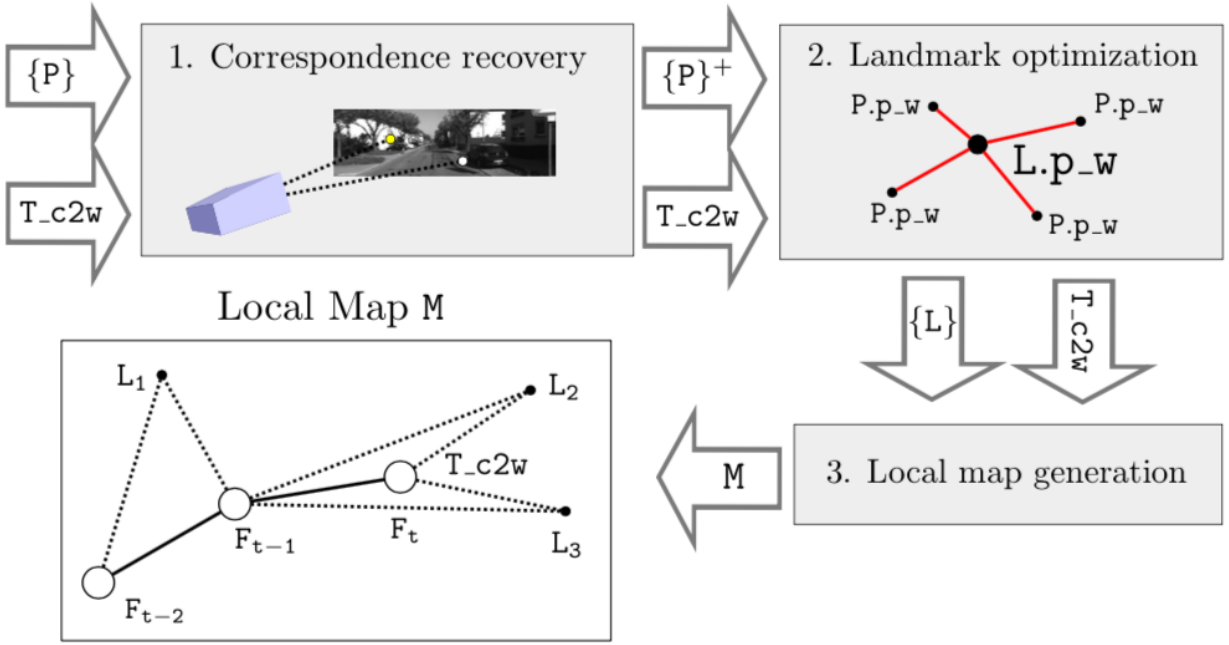
π: cam proj function, P_L: left cam proj matrix, T_m: motion

Constant velocity evolution model:

$$T_m = F_{t-1}.T_{w2c} * F_{t-2}.(T_{w2c})^{-1} \tag{5}$$

$$T_{w2c} = T_m * F_{t-1}.T_{w2c} \tag{6}$$

Map Management: phase three



Correspondence recovery

Bookkeping unmatched framepoints

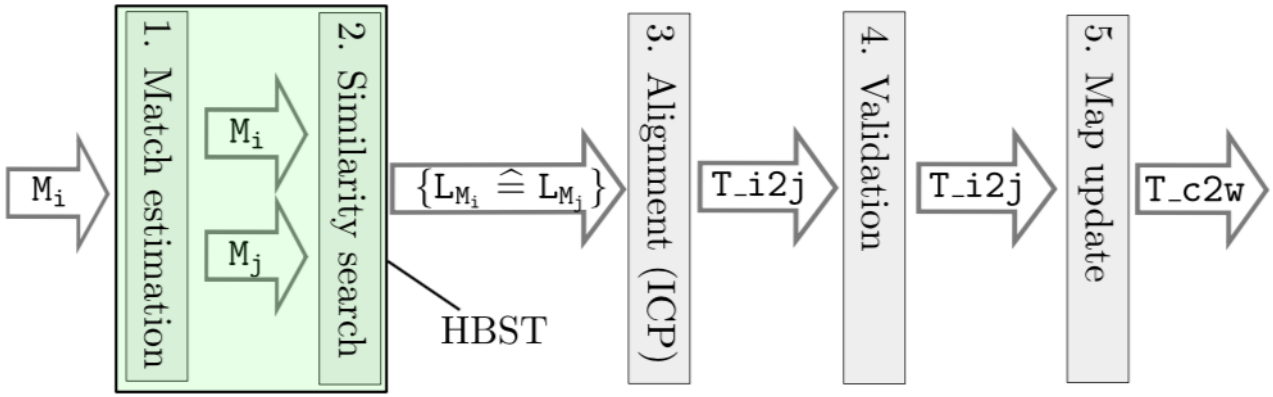
Landmark optimization

Refine extimation with information filter

Local map generation

If translation or rotation wtr previous one exceeds threshold

Relocation: phase four

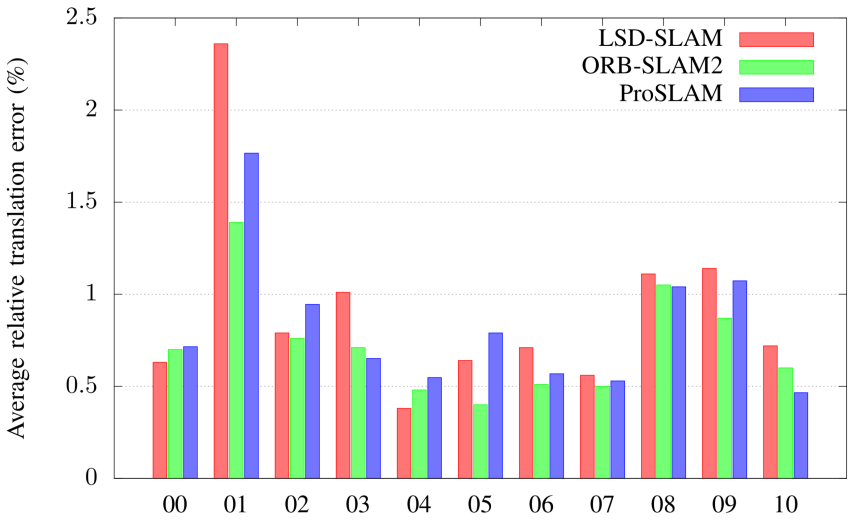


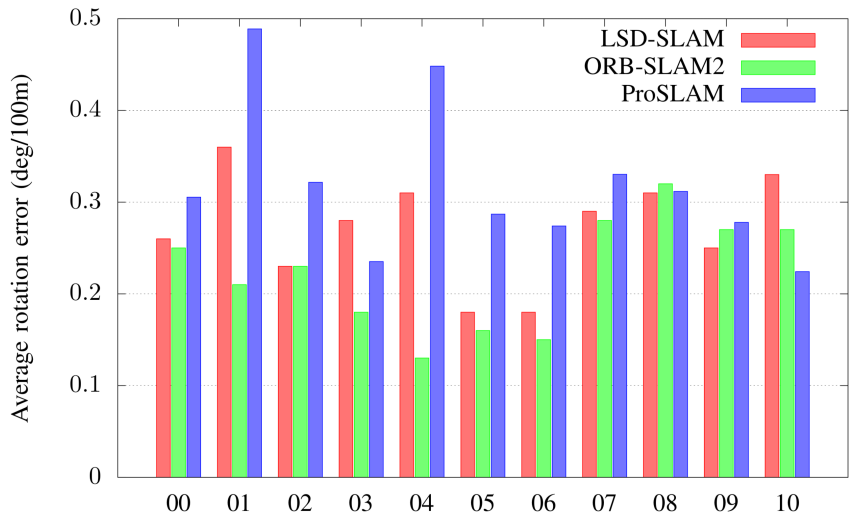
$$M_i \xrightarrow[\text{relocalize}]{} M_j \qquad j : 1, \ldots, i-1 \tag{7}$$

- Compute relative transformations T_{i2j}.
- Validate through **outliers** and **avg error**.

Results

Results: Translational and rotational Errors





Positive:

Same **translational** error as state of the art.

Negative:

Losing in **rotational** error evaluations.

Conclusion

Conclusion

To summarize:

- Results shown **nearly** state of the art performance.
- Algorithm runs on portable computer in real time (*light-weight*).
- Open source and **understable** implementation for newcomers.

Take home message

If you want to learn SLAM, **ProSLAM** is the way to start with.

Demo video

Placeholder for the movie

That's all

Thank you for your attention !

Any **question** *?*

Feel free to ask !