TÉCNICO LISBOA

Institute for Systems and Robotics | LISBOA

Determining the orientation of a RGB camera embedded on a robotic eye

Mariana Martins

2019

LARSyS
Laboratory of Robotics and Engineering Systems

VIS LAB

Overview

1. Introduction
2. Background
3. Methods
4. Implementation
5. Results and Discussion
6. Conclusions

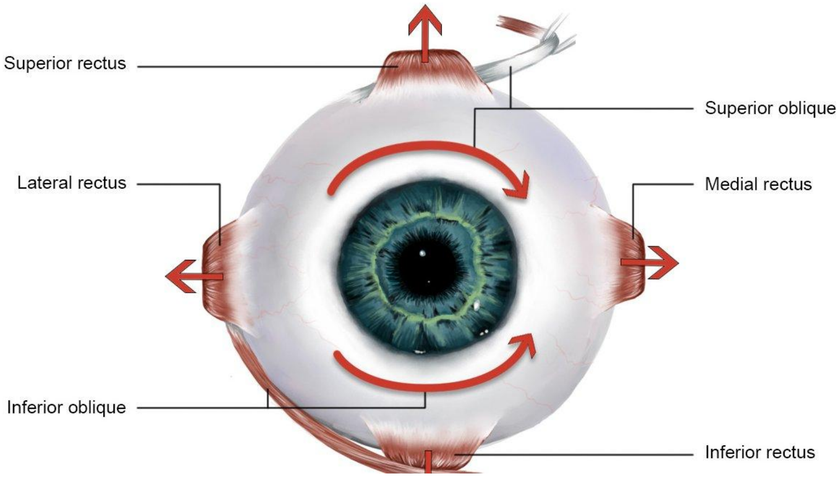
Overview

1. Introduction
2. Background
3. Methods
4. Implementation
5. Results and Discussion
6. Conclusions

1. Introduction: **Context**

How is the eye movement defined?

A humanoid eye-head robot might help us at understanding.

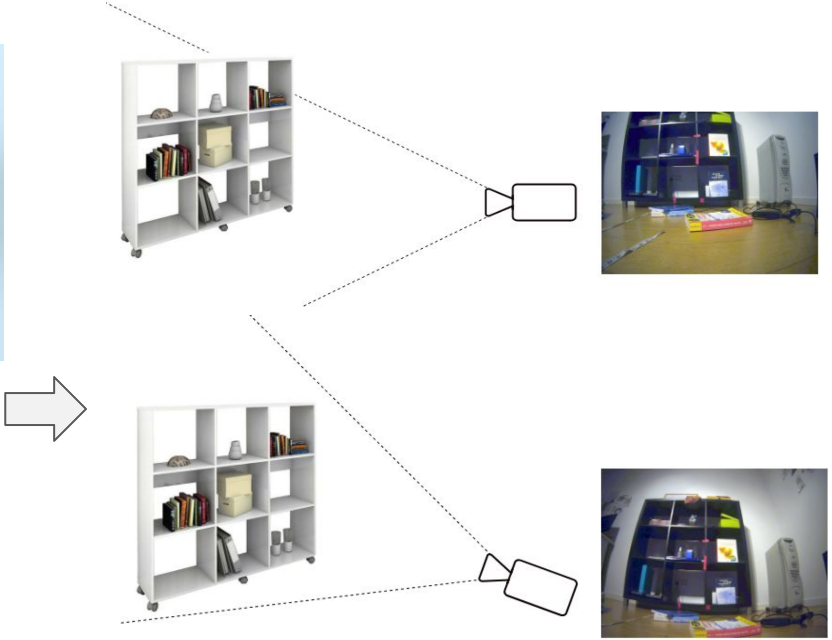


[1]

ORIENT project http://www.mbfys.ru.nl/ johnvo/OrientWeb/orient.html

[1] Eye Movement. https://step1.medbullets.com/neurology/113079/eye-movement. Accessed on November 10th 2019.

1. Introduction: **Problem**

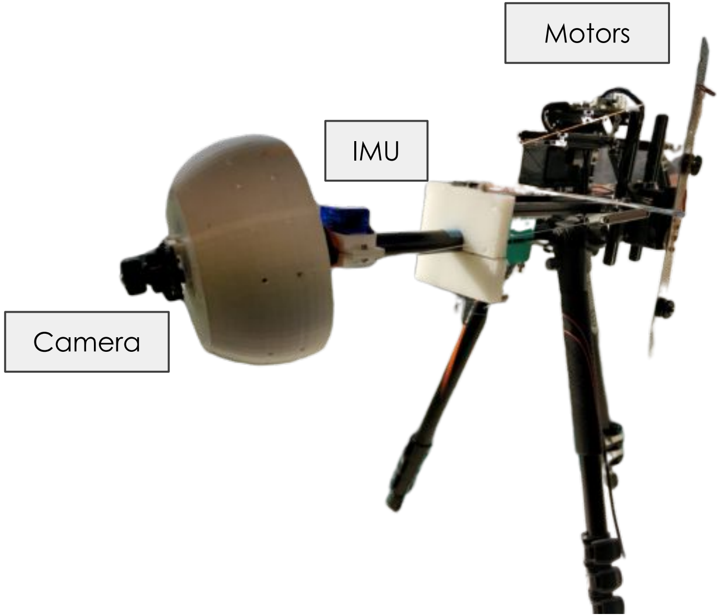


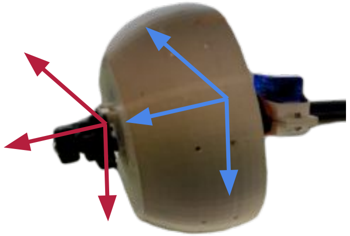
Essential to know eye orientation

IMU is not very accurate due to drift

[2] LPMS-CU. https://lp-research.com/lpms-cu/. Accessed on November 10th 2019.

1. Introduction: **Problem**





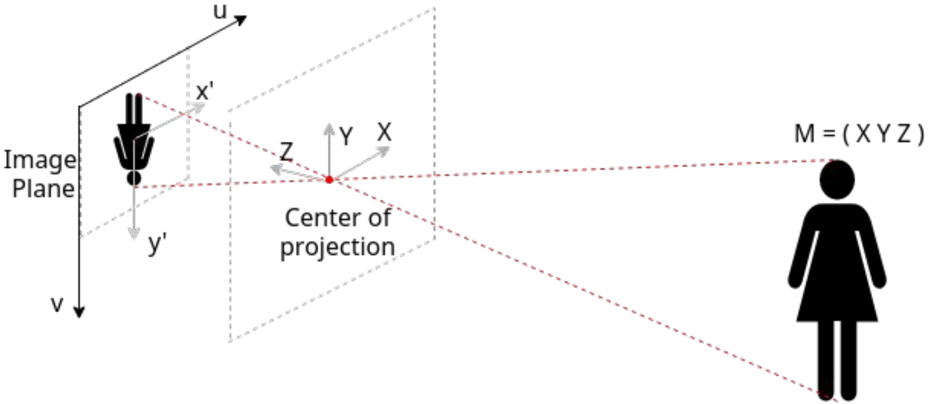
There is known translation associated to the camera movement

1. Introduction: **Objectives**

- Develop an algorithm to determine the 3D orientation
- Determine accuracy improvement using the system's constraints
- Understand how different methods behave in the simulator and in real world
- Evaluate the improvement using the camera over the IMU

Overview

2. Background: **Camera Model**



1) Detect features

2) Find matches

$$\lambda \widetilde{\mathbf{m}}_{\mathbf{p}} \sim K[R|t]\widetilde{M}$$

3) Determine orientation

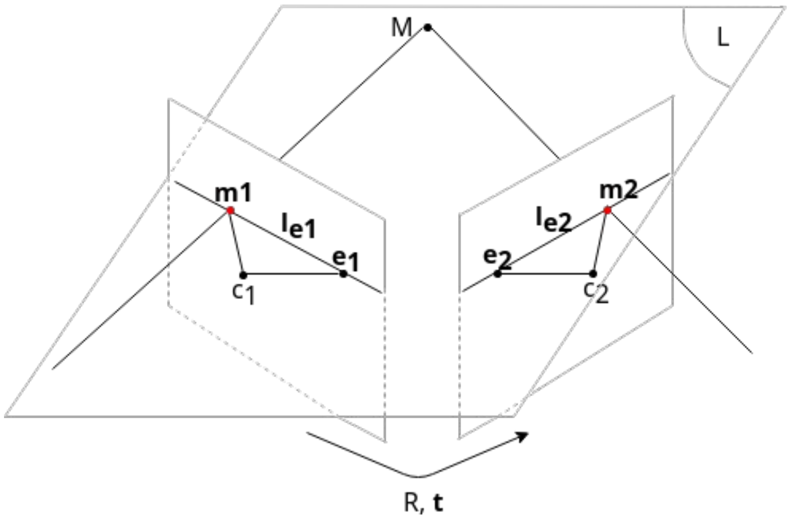
2. Background: **Orthogonal Procrustes Problem (OPPR)**

$$\|M_1 - RM_2\|^2$$

$$SVD(M_1 M_2^T)$$

$$R = VU^T$$

2. Background: **Epipolar Geometry**

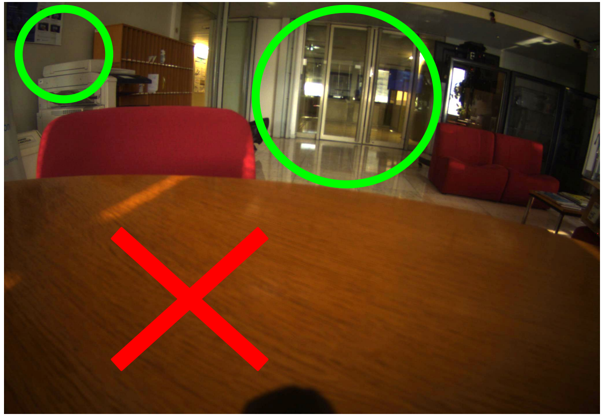


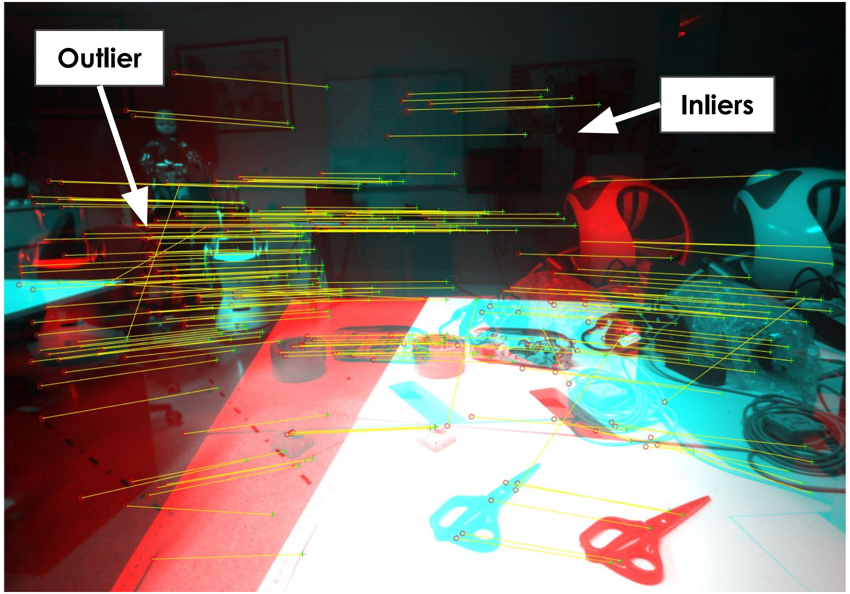
Gradient Based Technique (GRAT)
$$\min_F = \sum_i \left(\frac{\widetilde{\mathbf{m}_{i2}} F \widetilde{\mathbf{m}_{i1}}}{\sigma_i} \right)^2$$ $$\sigma_i^2 = \left[l_{e1i_x}^2 + l_{e1i_y}^2 + l_{e2i_x}^2 + l_{e2i_y}^2 \right]$$

$$\tilde{\mathbf{m}}_{\mathbf{2}}^T F \tilde{\mathbf{m}}_{\mathbf{1}} = 0 \longrightarrow E = K^T F K \longrightarrow E = [t]_\times R$$

2. Background: **Robust estimation**

Eliminate outliers using
RANSAC +OPPR (pure rotation fit)





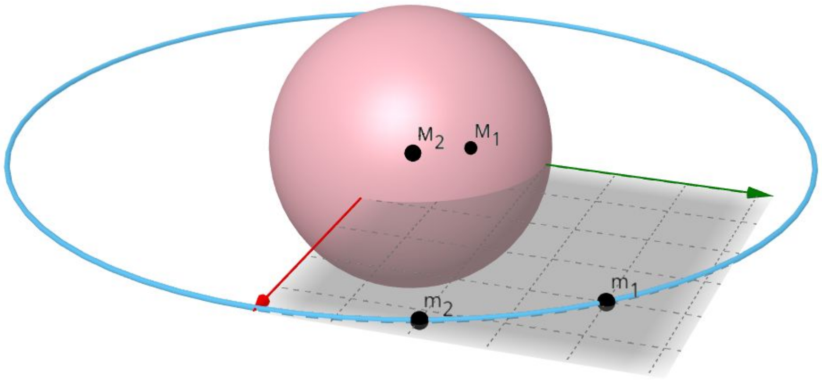
Overview

3. Methods

- Orthogonal Procrustes Problem (OPPR)
- Gradient Based Technique (GRAT)
- Minimization of Backprojection Error (MBPE)

Baseline constraint

3. Methods: **Depth & Orthogonal Procrustes Problem (OPPR)**



Ignores translation constraint

Camera has NO depth (λ) information

$$\|M_1 - RM_2\|^2 \leftarrow M_1 = \lambda\widetilde{\mathbf{m}}_1 \leftarrow \|(\lambda x, \lambda y, \lambda)\| = 1, \lambda = \frac{1}{\sqrt{x^2+y^2+1}}$$

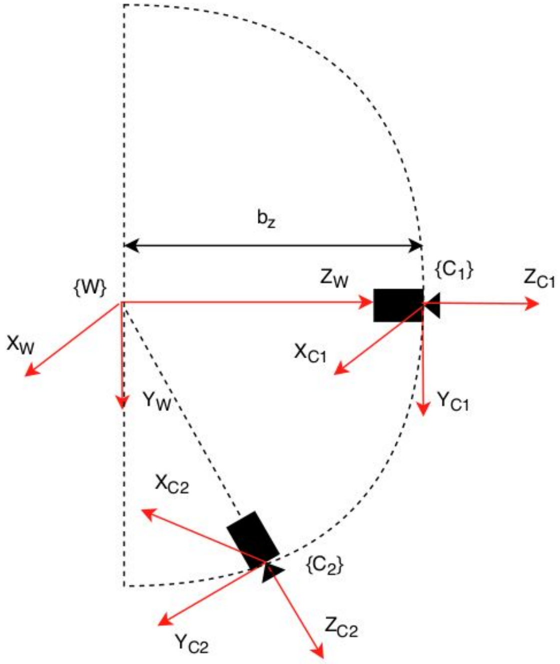
$$\widetilde{\mathbf{m}}_1 = [x\, y\, 1]^T$$

3. Methods: **Baseline constraint**

$$^{C_1}T_W = \begin{bmatrix} I & -\mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \quad ^{C_2}T_W = \begin{bmatrix} R & -\mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$^{C_2}T_{C_1} = ^{C_2}T_W^W T_{C_1} = \begin{bmatrix} R & -\mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} I & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R & R\mathbf{b} - \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\boxed{\mathbf{t}(R, \mathbf{b}) = (R - I)\mathbf{b}}$$



3. Methods: **Gradient-based technique**

$$min_R = \sum_i \frac{\widetilde{\mathbf{m_i}}_2^{\mathbf{T}} F \widetilde{\mathbf{m_i}}_1}{\sigma_i}$$

$$F = K^{-T} [\mathbf{t}]_\times R K^{-1}$$

No need to estimate depth

$$\mathbf{t} = (R - I)\mathbf{b}$$

Baseline constraint

Needs to be initialized….

3. Methods: **Minimization of Back Projection Error**

$$\min_{\theta,\lambda_1^T,\ldots,\lambda_{1n}^T} \sum_{i=1}^{n} \left[\left(u_{1i}^r - u_{1i}\right)^2 + \left(v_{1i}^r - v_{1i}\right)^2 + \left(u_{2i}^r - u_{2i}\right)^2 + \left(v_{2i}^r - v_{2i}\right)^2 \right]$$

$$\mathbf{m}_{1i} = \left[u_{1i} \;\; v_{1i}\right]$$

$$\lambda_1 \tilde{\mathbf{m}}_\mathbf{1}^\mathbf{r} = K \left[R^T \quad -R^T \mathbf{t} \right] K^{-1} \lambda_2 \tilde{\mathbf{m}}_\mathbf{2}$$

$$\mathbf{t} = (R - I)\mathbf{b}$$

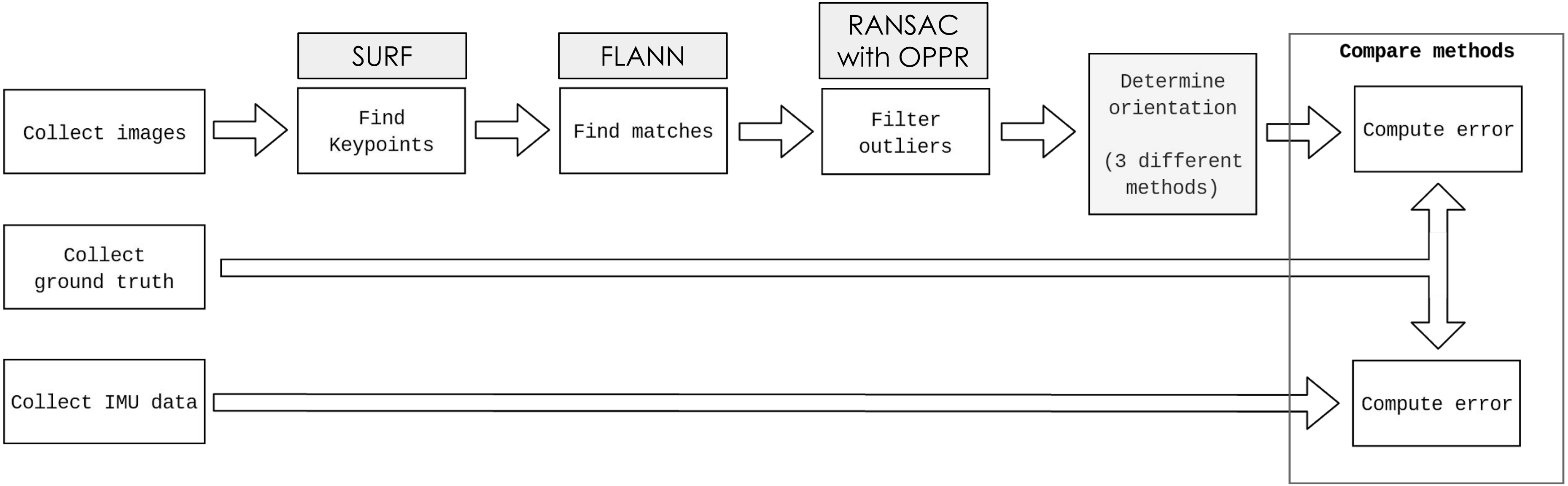
$$\lambda_2 \tilde{\mathbf{m}}_2^\mathbf{r} = K[R \;\, \mathbf{t}] K^{-1} \lambda_1 \tilde{\mathbf{m}}_1$$

May have more accuracy by adapting depth

Needs to be initialized….

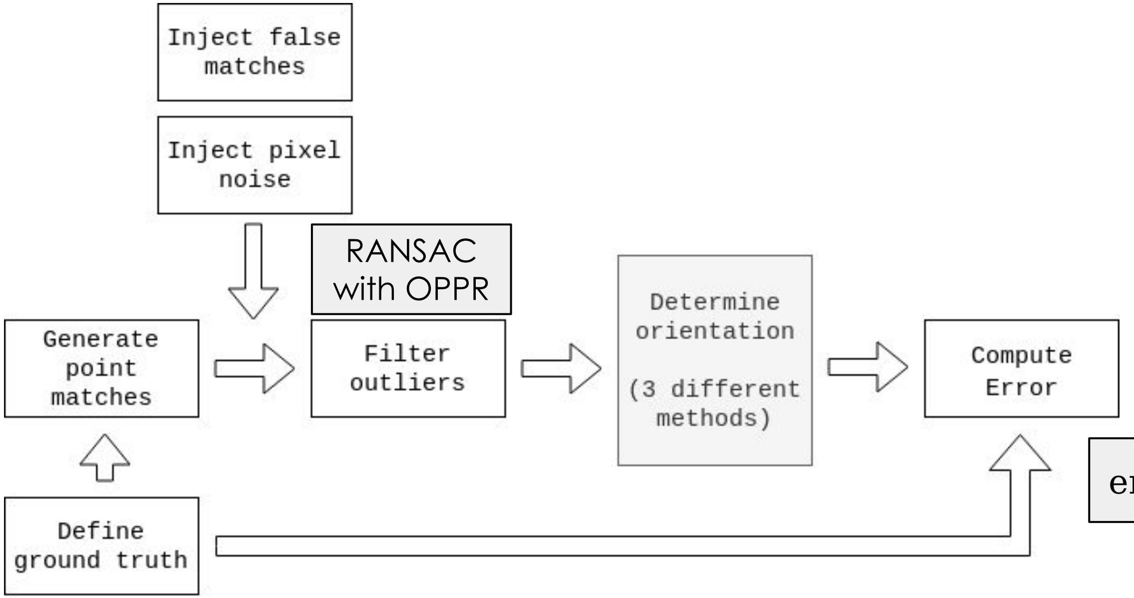
Overview

4. Implementation



$$\text{error} = \|\theta_{gt} - \theta_{est}\|$$

4. Implementation: Simulator

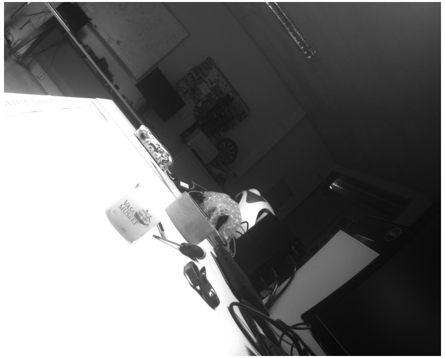


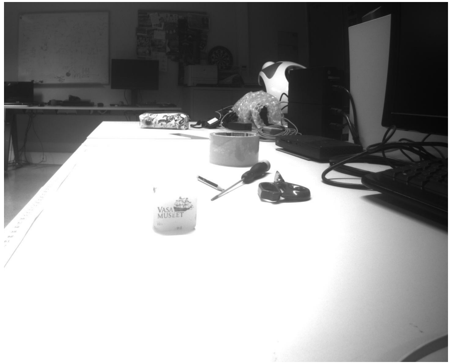
$$\text{error} = \|\theta_{gt} - \theta_{est}\|$$

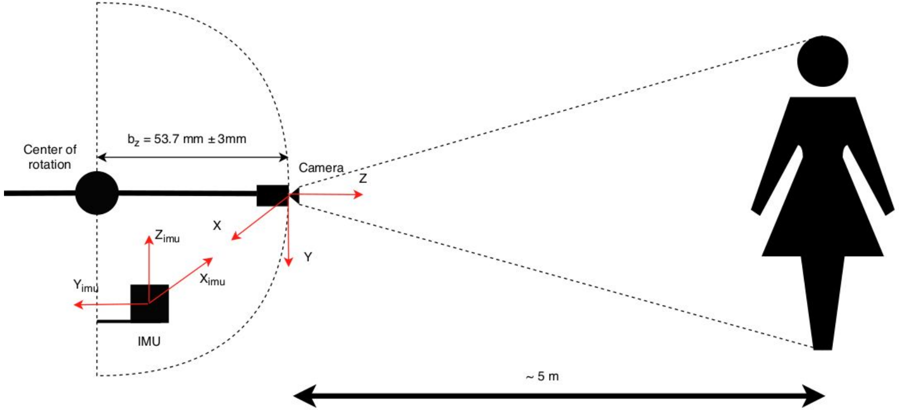
4. Implementation

Collect images

Ueye LE RGB Camera



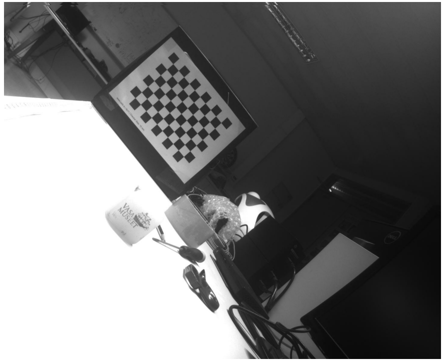


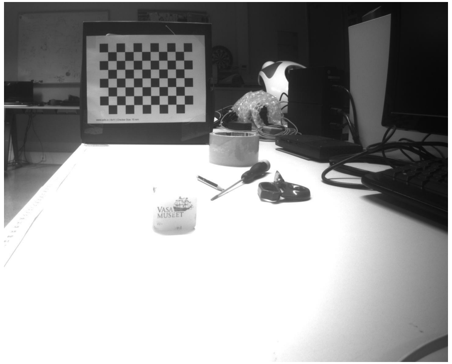


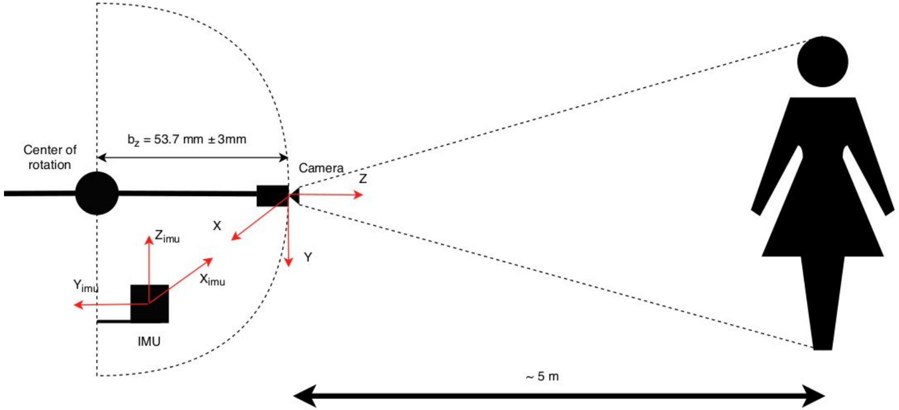
4. Implementation

Collect ground truth

Matlab library







Overview

1. Introduction
2. Background
3. Methods
4. Implementation
5. Results and Discussion
6. Conclusions

5. Results and Discussion: Experiments

Simulator

1. Effect of RANSAC with OPPR
2. Estimation error as a function of rotation amplitude
3. Error per depth
4. Error per pixel noise

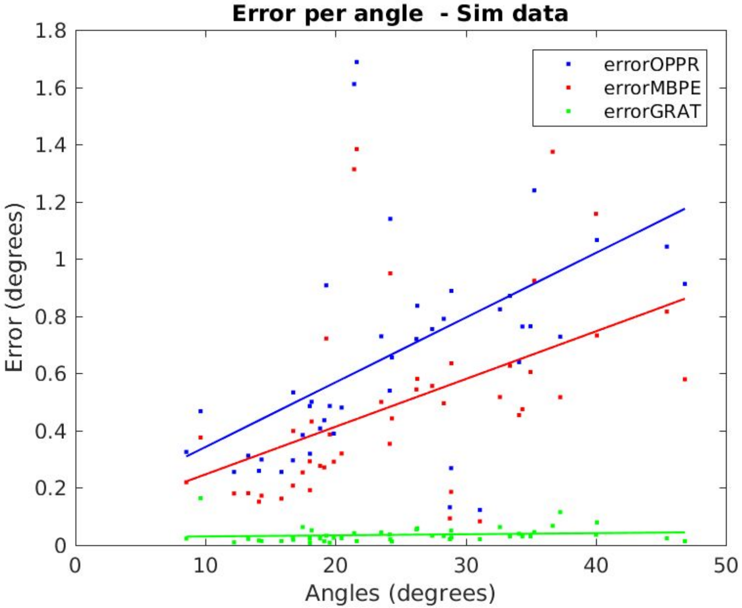
Real data

1. **Camera** Estimation error as a function of rotation amplitude
2. **IMU** estimation error as function of rotation amplitude

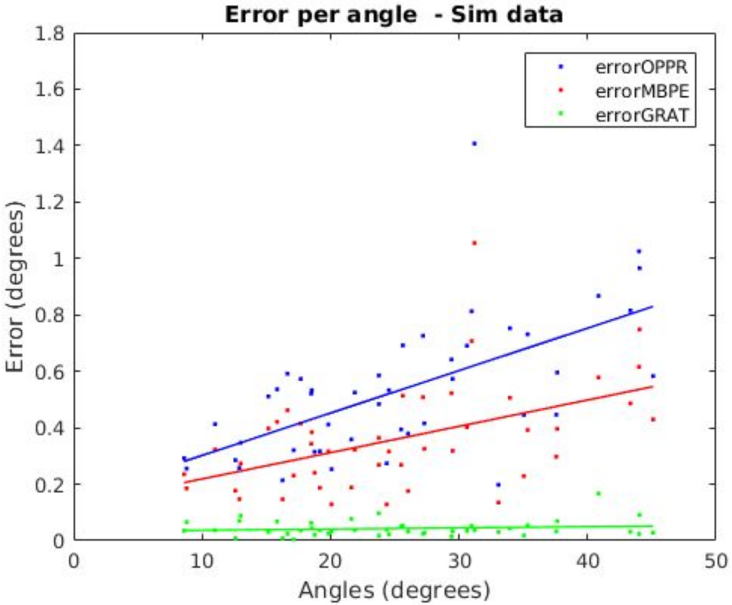
5. Results and Discussion: Simulator

NO noise, NO false matches

Error as a function of amplitude without ...



... and with robust estimation

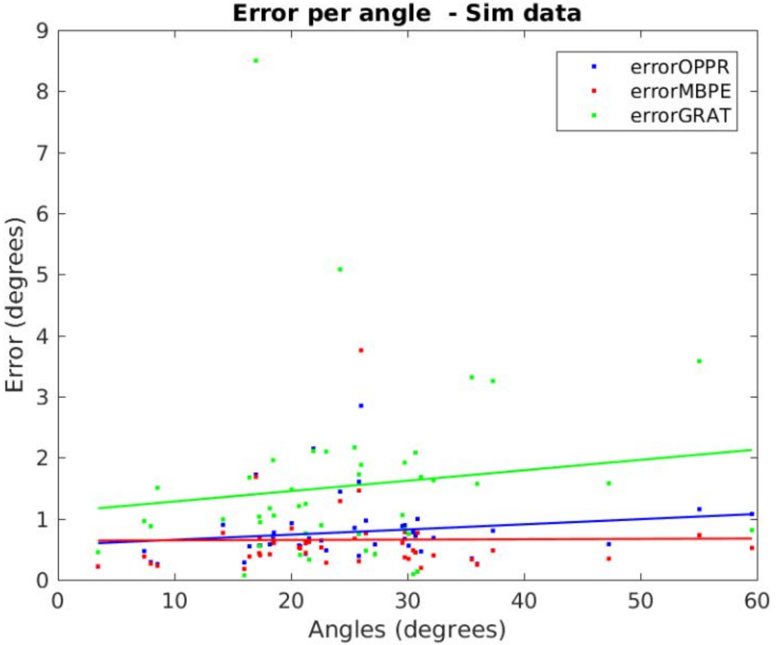


with standard deviation of 15 degrees

5. Results and Discussion: Simulator

With noise, With false matches

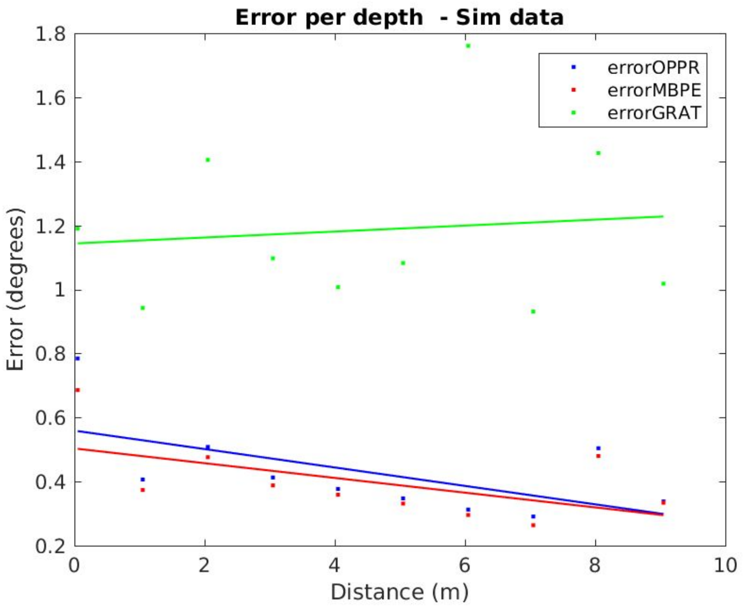
Error as function of amplitude with standard deviation of 15 degrees



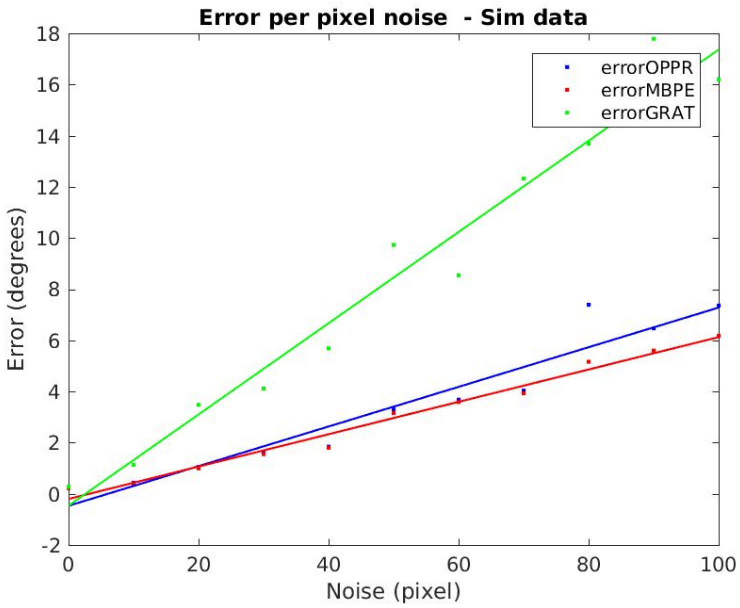
Method	Mean	Standard Deviation
OPPR	0.77 deg	0.49 deg
MBPE	0.65 deg	0.60 deg
GRAT	1.53 deg	1.43 deg

5. Results and Discussion: Simulator

Error per depth



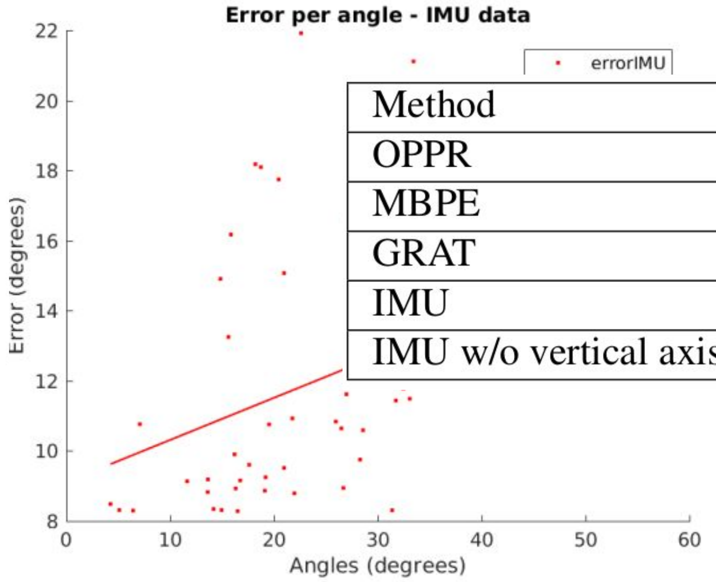
Error per pixel noise



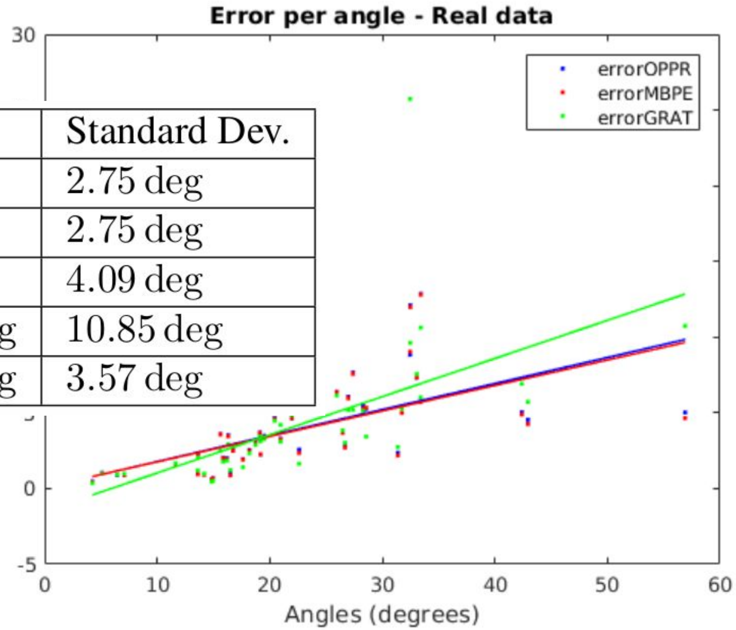
with standard deviation of 4 degrees

5. Results and Discussion: Real data

IMU Error as a function of amplitude



Camera Error as a function of amplitude



Method	Mean	Standard Dev.
OPPR	3.90 deg	2.75 deg
MBPE	3.83 deg	2.75 deg
GRAT	4.13 deg	4.09 deg
IMU	20.34 deg	10.85 deg
IMU w/o vertical axis	11.79 deg	3.57 deg

5. Results and Discussion: Computational times

Process	Time
Image Capture	120 ms
Undistort + Find Keypoints	421 ms
Find Matches	16 ms
Robust Estimation	10 ms
OPPR	13e−2 ms
MBPE	212 ms
GRAT	51 ms
IMU Estimation	40e−4 ms

Worst case scenario
567 ms

Overview

1. Introduction
2. Background
3. Methods
4. Implementation
5. Results and Discussion
6. Conclusions

6. Conclusions

- OPPR is the **fastest**
- MBPE is equally **accurate** to OPPR, but the **slowest**
- GRAT is the **most sensitive to noise**. Does not work for pure rotations
- RANSAC+OPPR improves the estimation
- IMU is faster than the camera estimation but much less accurate

6. Conclusion: **Contributions**

- Empirical study on the best method to estimate 3D orientation with this system's particular constraints
- Derivation of the baseline constraint
- A method with better precision than an IMU
- An open-source C++ library for similar contexts within the community
- A matlab simulator

6. Conclusions: **Future work**

Study how using a Kalman Filter on the IMU and camera together can improve the results in terms of speed

Thank you