23 Apr 2021

IEEE Transactions and Robotics

31 May 2024

SMLab Talks
Harshit Agarwal

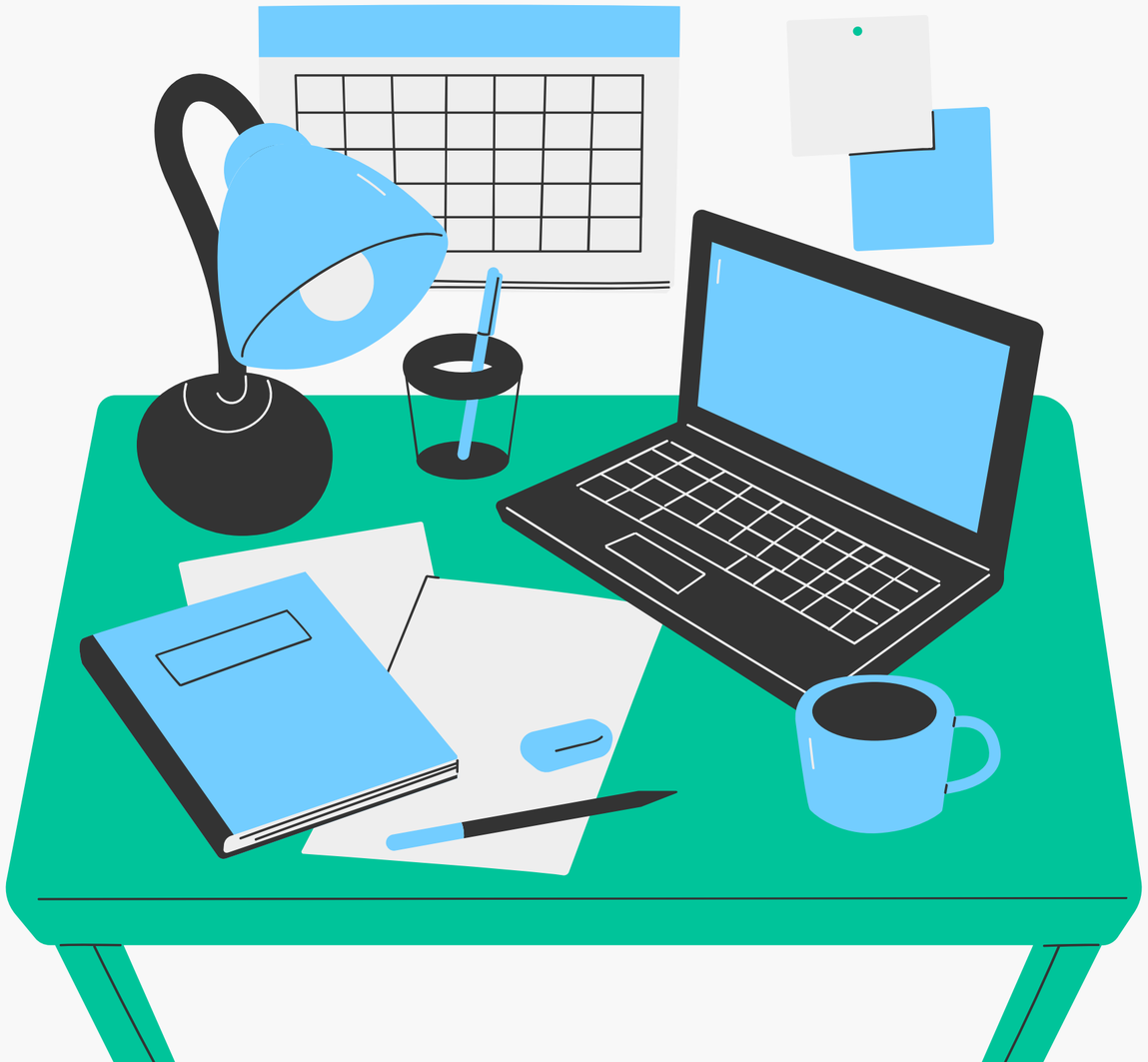
ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM

Carlos Campos , Richard Elvira , Juan J. Gomez Rodr ´íguez, Jose M.M. Montiel and Juan D. Tard'os

Table of Contents

- classic problem of mapping
- used for 3d rendering of structures and for autonomous driving in robots and vehicles
- task of simultaneous localization and mapping arises

use of various algorithms from visual feature extractions and matching to point clouds. photogrammetry, neural radiance fields and splatting in terms of modern innovations of mesh renderings.



Visual, Inertial, LiDAR

Visual : camera (rgb/grayscale)
Inertial : accelerometer, gyroscope, magnetometer etc.
LiDAR : for depth maps

Mapping

The problem of mapping is essentially to use data of visual, inertial and/or depth in nature and create a mathematical model based on computations/approximations that can be represented in a virtual context.

II pose graph optimization

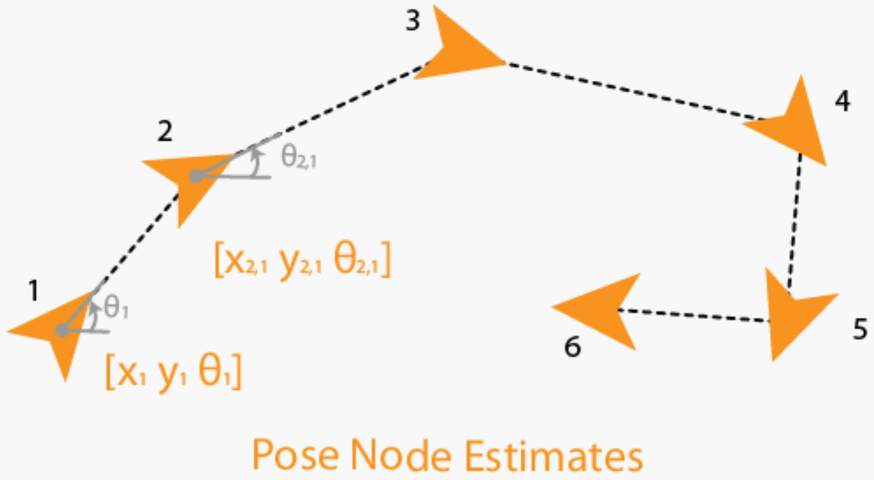
What are pose graphs ?

Graphs having nodes that contain [pose, measurement] linked/constrained with weighted edges

- The problem here essentially is to continuously make new nodes/measurement while optimising their constraints to better represent reality.

Loop Closures

- When the camera reaches a previous pose (location and rotation), it's compared with it's constrained path from it's previous iteration and in case of drift, the constrains are re-projected and adjustments are made.



III keypoints and descriptors

What are Keypoints and Descriptors ?

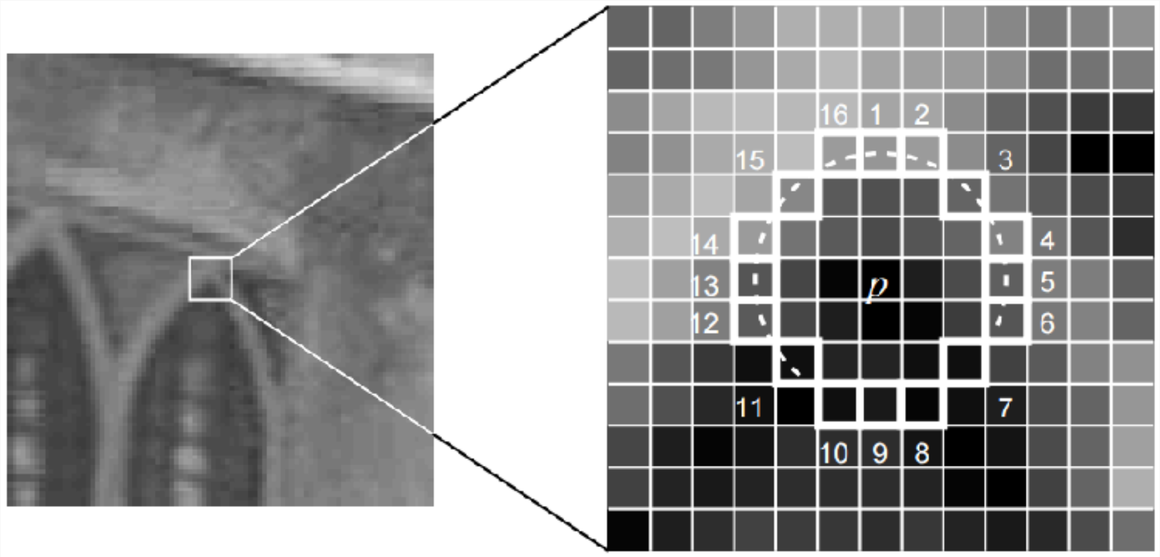
keypoints represent extracting the location of the distinct local feature
descriptors represent how to describe the extracted keypoints (vectors)

- FAST (Features from Accelerated Segment Test) is used as the key-point detector, by basically finding points with local brightness differences that make them key.
- Algorithms like SIFT, SURF, Shi Tomasi and ORB are used for features.
- BRIEF or Binary Robust Independent Elementary Features are used as the keypoint descriptors.
- They describe the features in a binary fashion with regards to the other keypoints thus allowing for easier computation during feature rmatching.

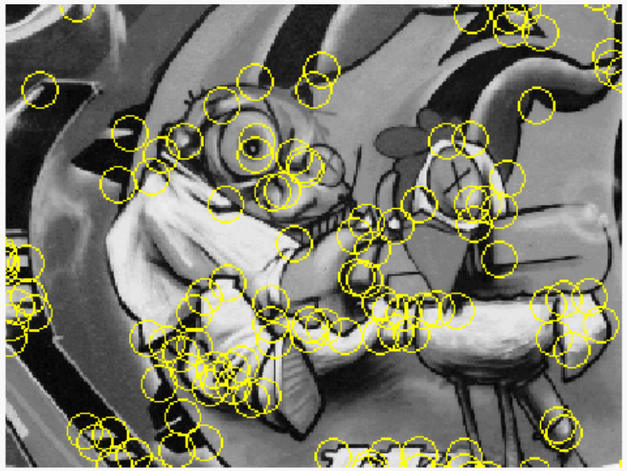
Note: in feature matching, a point in img A is compared with every keypoint in img B

III keypoints and descriptors

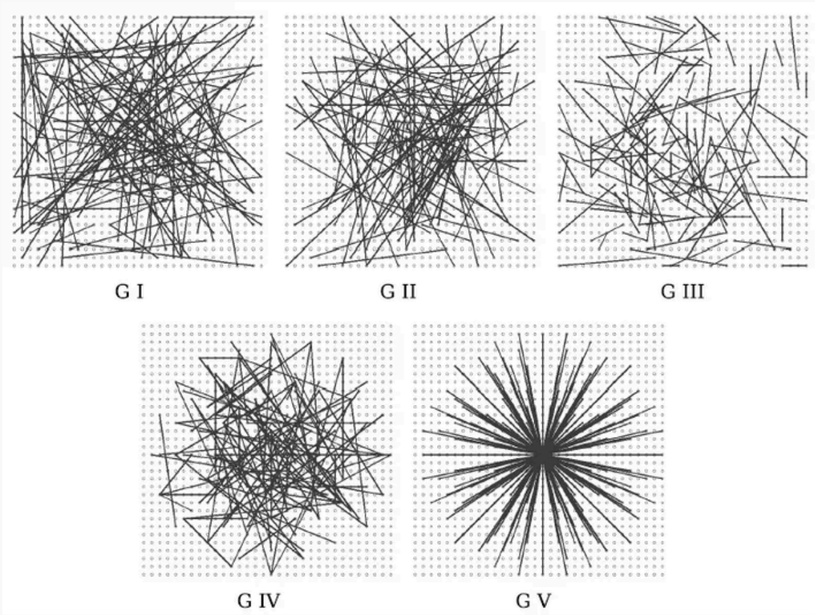
What are Keypoints and Descriptors ?



Features from Accelerated Segment Test, ECCV 06



- ORB (Oriented FAST and Rotated BRIEF), CVPR 11



Binary Test Patterns for BRIEF

Lets talk about ORB - Oriented FAST and Rotated BRIEF

Intensity Centroid -> Rotation -> Matching

- At first an intensity centroid is calculated for all keypoints in an image.
- Then we rotate coordination of all pairs based on its center of mass.
- Now feature matching becomes a computationally lighter task as well as highly rotation invariant.

Our approach uses a simple but effective measure of corner orientation, the *intensity centroid* [22]. The intensity centroid assumes that a corner's intensity is offset from its center, and this vector may be used to impute an orientation. Rosin defines the moments of a patch as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y), \tag{1}$$

and with these moments we may find the centroid:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{2}$$

We can construct a vector from the corner's center, O, to the centroid, $\vec{OC}$. The orientation of the patch then simply is:

$$\theta = \text{atan2}(m_{01}, m_{10}), \tag{3}$$

III ORB keypoint matching live demo

Feature Extraction and Matching Process

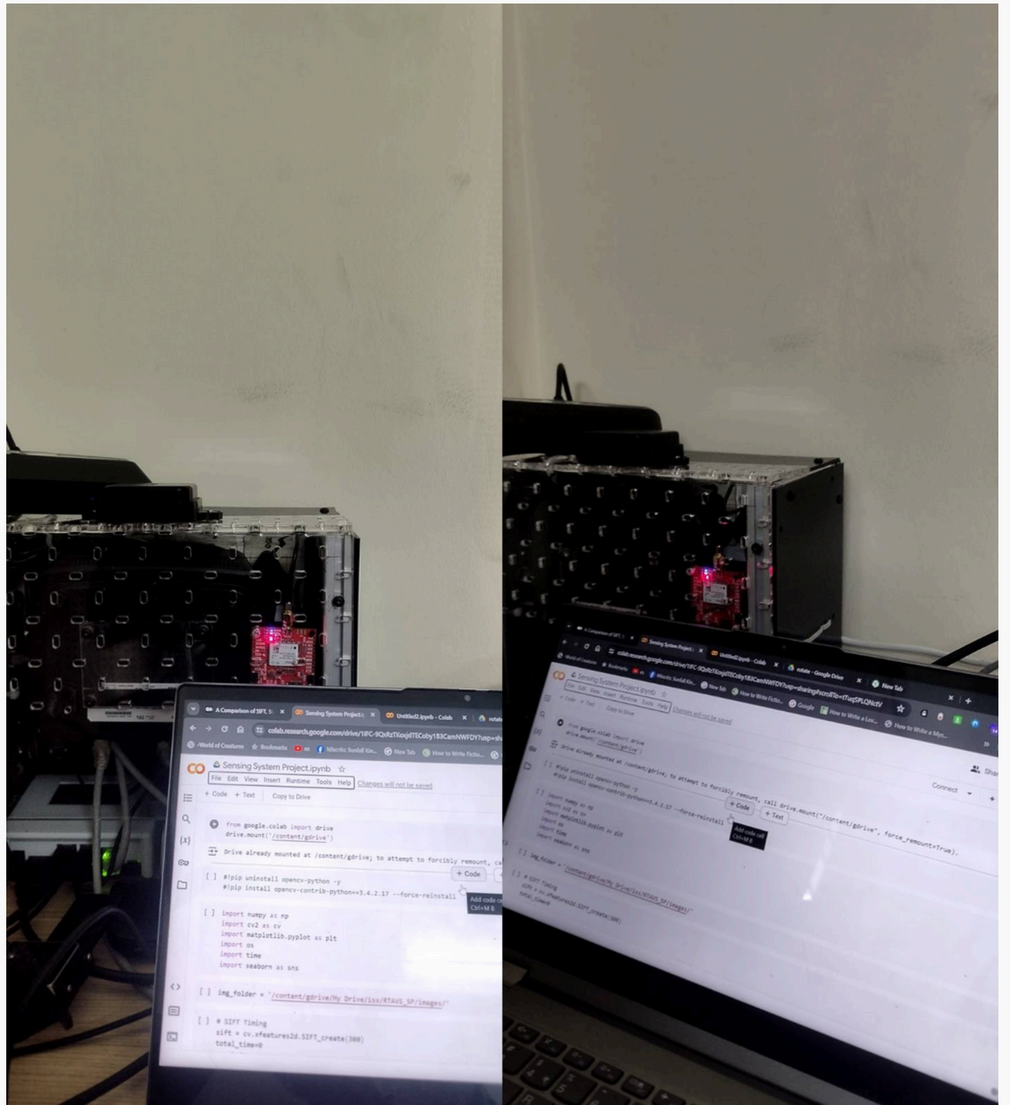
```
orb = cv.ORB_create(1000000)
total_time=0
total_features=0
bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
p_matched = []
dist_list = []

img1 = cv.imread(img_folder+'/img2.png')
img2 = cv.imread(img_folder+'/img1.png')
# print(img1)
kp1,des1 = orb.detectAndCompute(img1,None)
kp2,des2 = orb.detectAndCompute(img2,None)
matches = bf.knnMatch(des1,des2,k=1)
matches = sorted([m for m in matches if len(m) > 0], key=lambda x: x[0].distance)
p_matched.append(len(matches)/len(kp1))

# print(kp1,"\n",kp2,"\n",des1,"\n",des2,"\n")

if reduce:
    matches = matches[:500]
distance = []
for match in matches:
    match = match[0]
    c1 = kp1[match.queryIdx].pt
    c2 = kp2[match.trainIdx].pt
    dist = ((c1[0]-(c2[0]*(-1)+720))**2+(c1[1]-(c2[1]*(-1)+480))**2)**0.5
    distance.append(dist)
    dist_list.append(np.average(dist))

orb_per_r = np.average(p_matched)
orb_dist_r = np.average(dist_list)
print('Percentage of Matched Keypoints: ', orb_per_r)
print('Drift of Matched Keypoints: ', orb_dist_r)
```

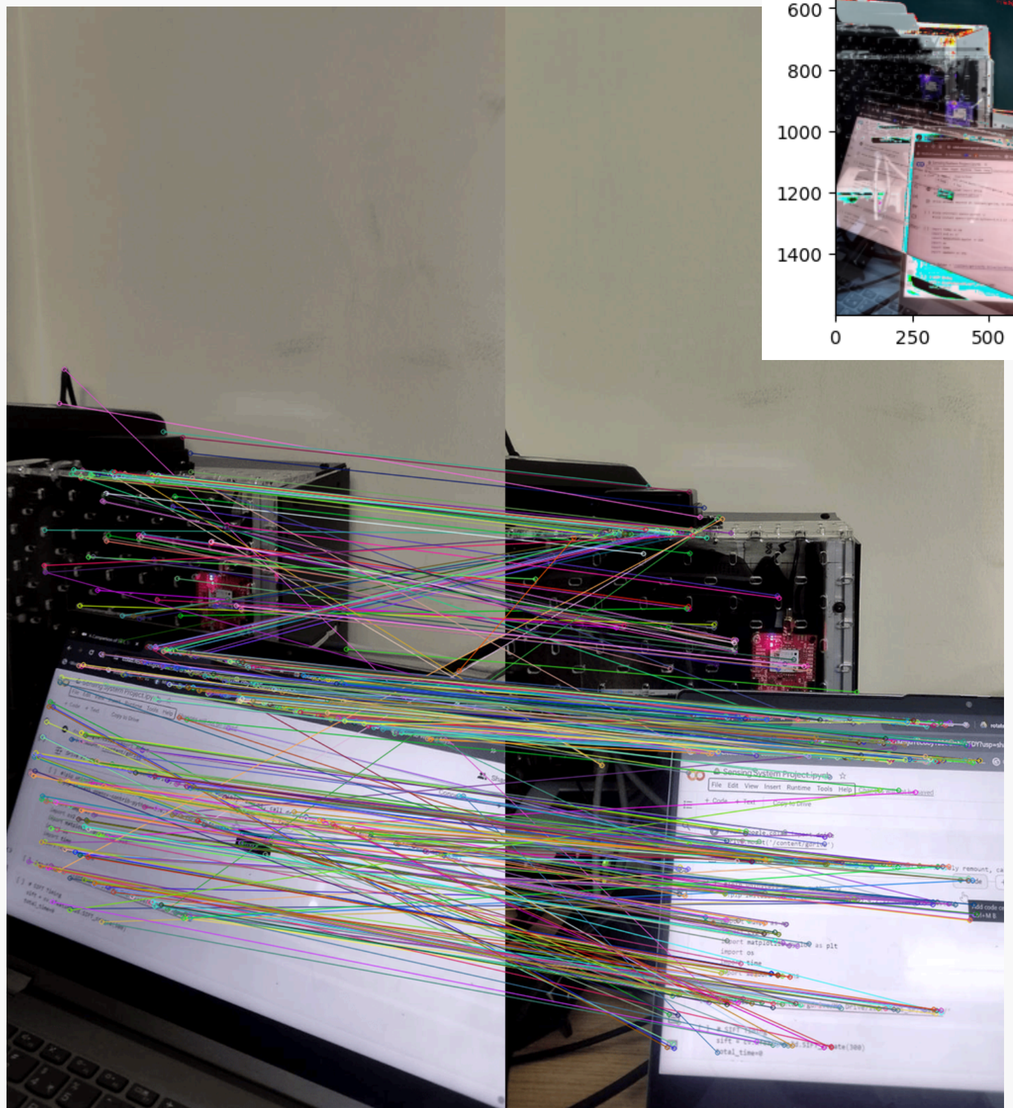


III ORB keypoint matching live demo

Results

```
Average time for ORB features:  0.0195797443389892 56
Average number of ORB features:  3835.2
```

```
Percentage of Matched Keypoints:  0.28125
Drift of Matched Keypoints:  1668.6395368524622
```



- As seen in the two images,
  - there's a huge number of ORB features in each image
  - about 30% of them are matched efficiently in a very less time

IV bag-of-word - DBoW2

Bag of Words

a technique to vectorise words and sentences.

Natural Language Processing

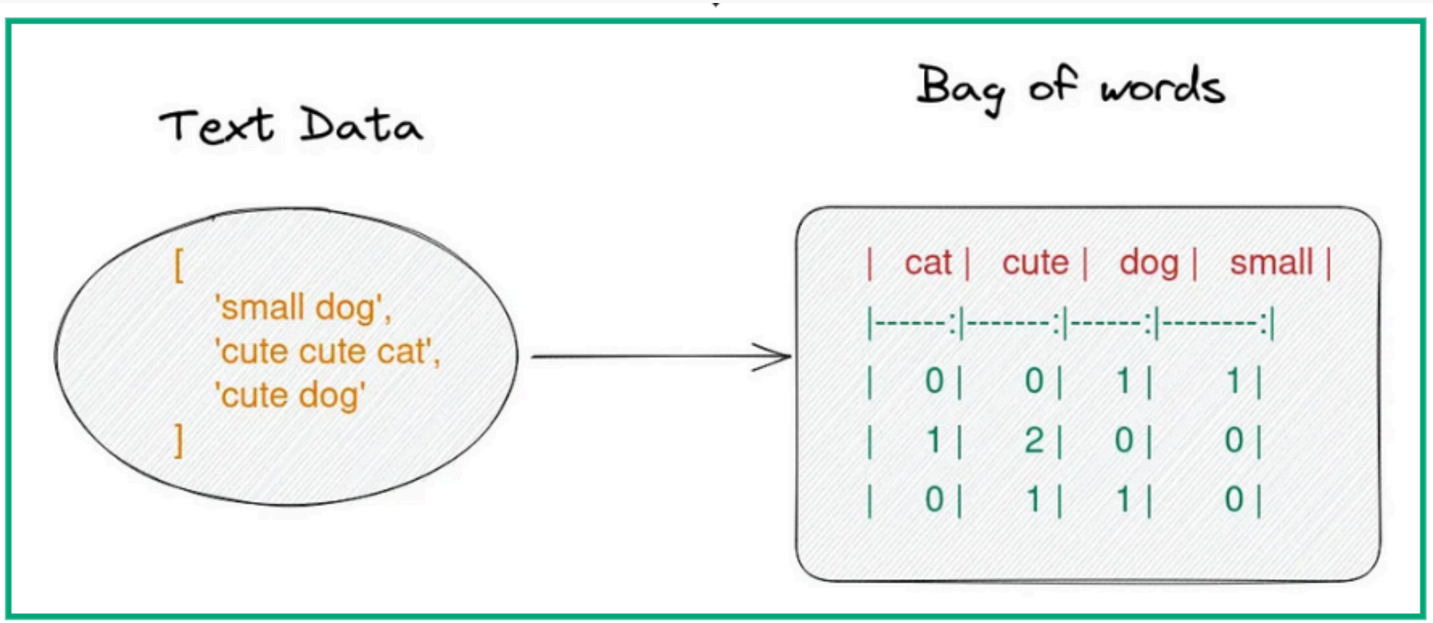
bow is used in NLP for vectorisation of sentences

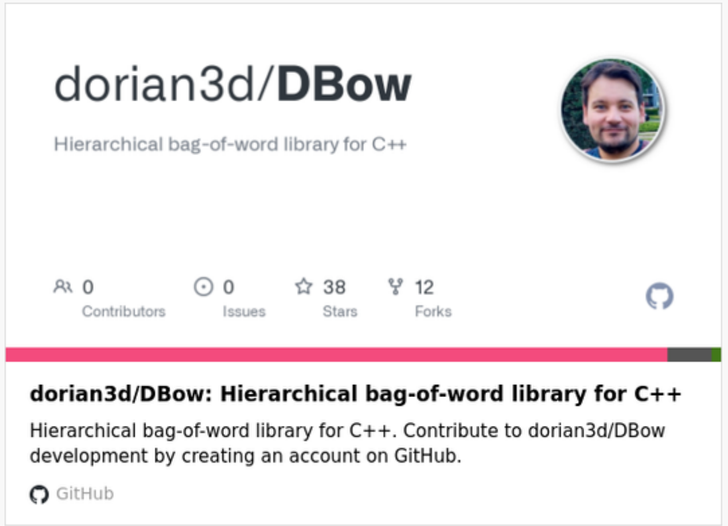
Features treated as words

image features (like sift) used with bag of words

Place recognition

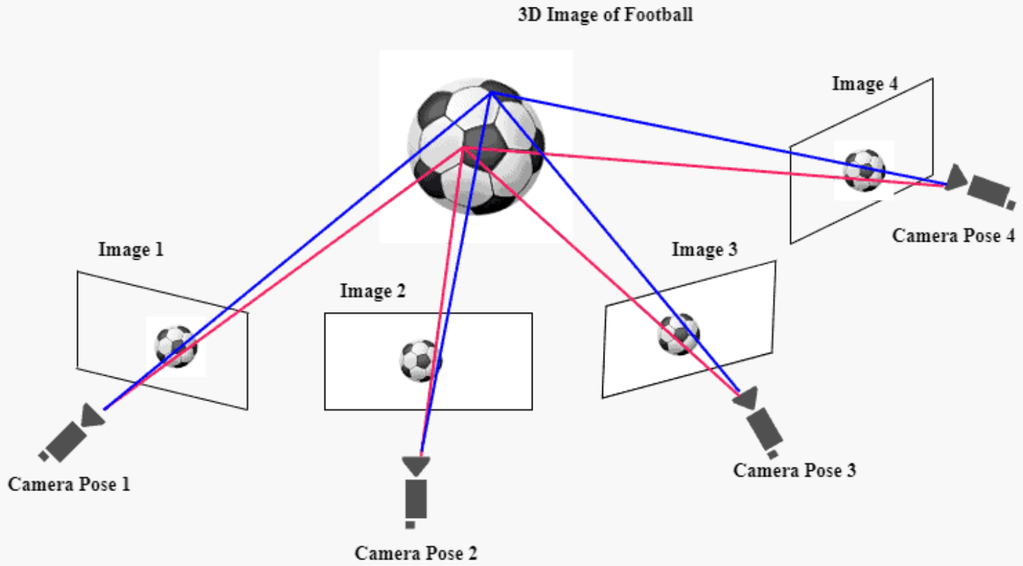
used for easier feature matching as well input for vision ml models.





V bundle adjustment

Bundle adjustment refers to process of refining initial estimates of structure and motion using non-linear optimisation.



$$c^i X_j = x_j^i$$

where xij, denotes the jth point as seen by the ith camera, and we find the ci matrix for the camera movement

Our goal thus become to minimize the image distance between reprojected point and detected image points xij:

$$min \sum_{i,j} d(c^i X_j, x_j^i)^2$$

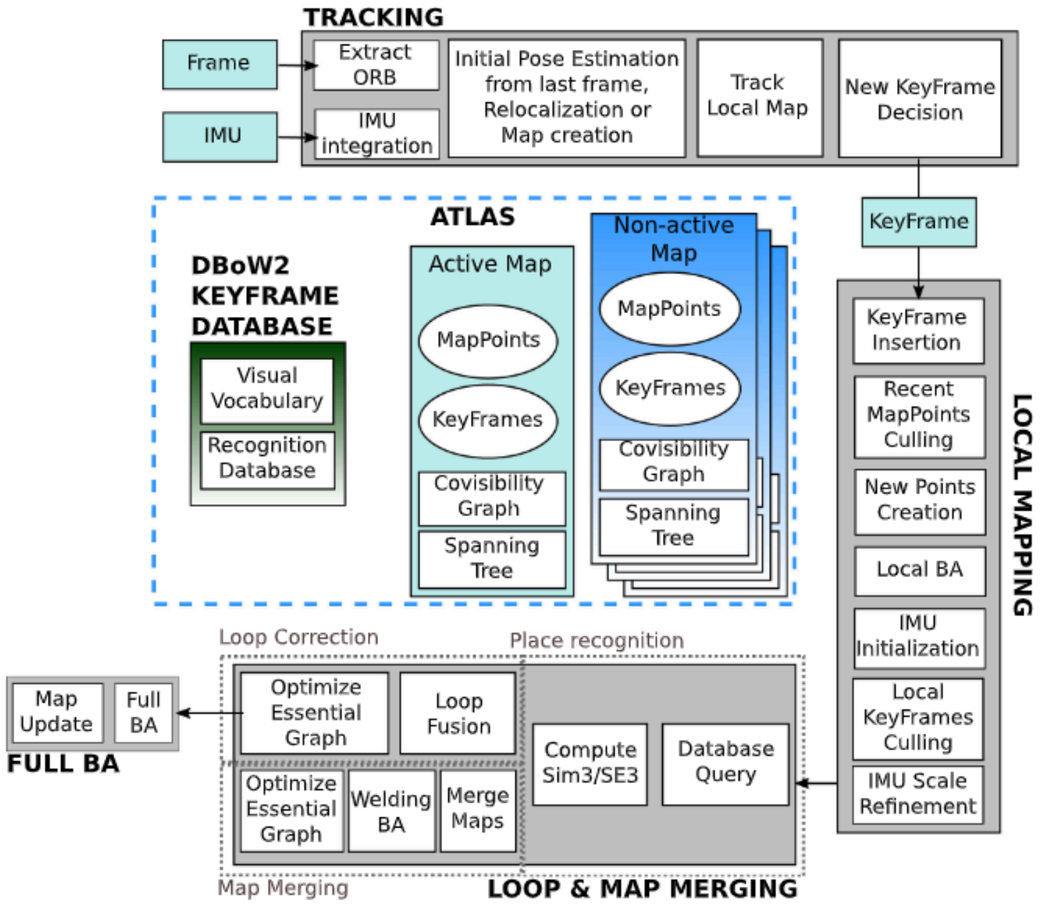
$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \sum_{j=1}^{J} \log[Pr(\mathbf{x}_{ij}|\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j)] \right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \sum_{j=1}^{J} \log \left[\mathrm{Norm}_{\mathbf{x}_{ij}} [\mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j], \sigma^2 \mathbf{I}] \right] \right]\end{aligned}$$

Parallel Threads and a Central Atlas

- tracking
- mapping
- loop closures and map merging
- full bundle adjustment

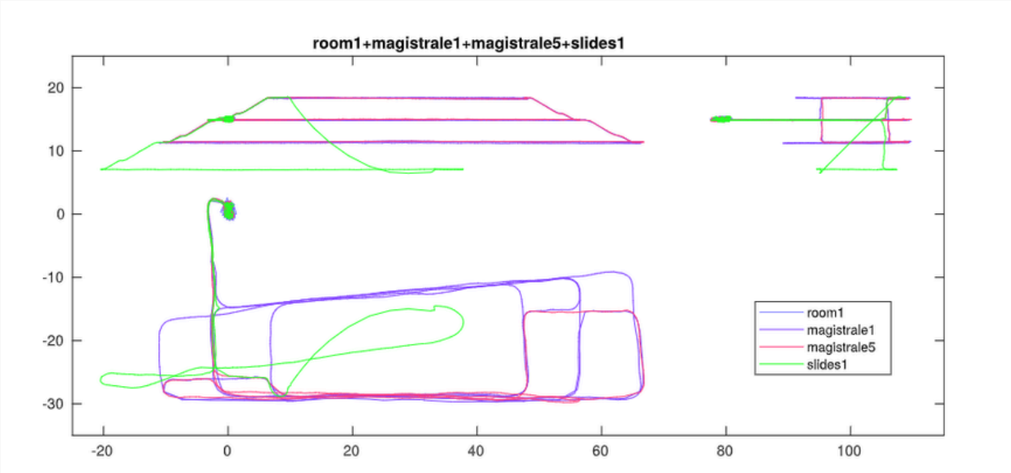
treating images as words, we can vectorize them using Bags of Words approach and use it in place recognition problem of loop closure, created from orb descriptors

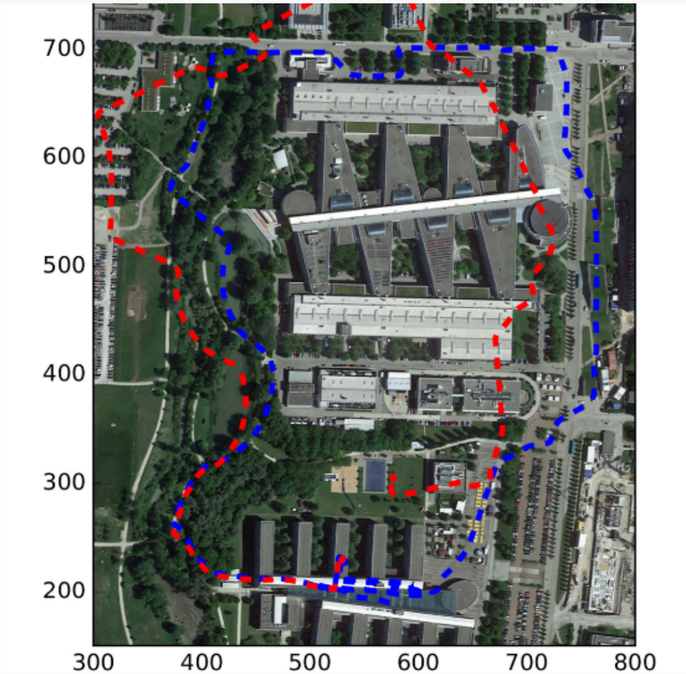
active map - weaved together while map merging older-**non active** maps while making newer maps in the local mapping threads



Tested on EuRoC MAV and TuM VI Datasets

- Visual-Inertial Sensor Unit
  - Stereo Images (Aptina MT9V034 global shutter, WVGA monochrome, 2×20 FPS)
  - MEMS IMU (ADIS16448, angular rate and acceleration, 200 Hz)
  - Shutter-centric temporal alignment
- Ground-Truth
  - Vicon motion capture system (6D pose)
  - Leica MS50 laser tracker (3D position)
  - Leica MS50 3D structure scan
- Calibration
  - Camera intrinsics
  - Camera-IMU extrinsics
  - Spatio-temporally aligned ground-truth





https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets

VII Research Results

In magistrale:

1. indoor sequences, which are up to 900 m long, most tracked points are relatively close, and ORB-SLAM3 obtains errors around 1 m except in one sequence that goes close to 5 m.
2. long outdoor sequences: the scarcity of close visual features may cause drift of the inertial parameters, notably scale and accelerometer bias, which leads to errors in the order of 10–70 m.

The novel place recognition method only takes 10 ms per keyframe.

Times for merging and loop closing remain below 1 s, running only a PG optimization.

For loop closing, performing a full BA may increase times up to a few seconds, depending on the size of the involved maps

			MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg[1]
Monocular	ORB-SLAM [4]	ATE[2,3]	0.071	0.067	0.071	0.082	**0.060**	**0.015**	0.020	-	**0.021**	**0.018**	-	0.047*
	DSO [27]	ATE	0.046	0.046	0.172	3.810	0.110	0.089	0.107	0.903	0.044	0.132	1.152	0.601
	SVO [24]	ATE	0.100	0.120	0.410	0.430	0.300	0.070	0.210	-	0.110	0.110	1.080	0.294*
	DSM [31]	ATE	0.039	0.036	0.055	**0.057**	0.067	0.095	0.059	0.076	0.056	0.057	**0.784**	**0.126**
	ORB-SLAM3 (ours)	ATE	**0.016**	**0.027**	**0.028**	0.138	0.072	0.033	**0.015**	**0.033**	0.023	0.029	-	0.041*
Stereo	ORB-SLAM2 [3]	ATE	0.035	**0.018**	0.028	0.119	0.060	**0.035**	**0.020**	**0.048**	**0.037**	0.035	-	0.044*
	VINS-Fusion [44]	ATE	0.540	0.460	0.330	0.780	0.500	0.550	0.230	-	0.230	0.200	-	0.424*
	SVO [24]	ATE	0.040	0.070	0.270	0.170	0.120	0.040	0.040	0.070	0.050	0.090	0.790	0.159
	ORB-SLAM3 (ours)	ATE	**0.029**	0.019	**0.024**	**0.085**	**0.052**	**0.035**	0.025	0.061	0.041	**0.028**	**0.521**	**0.084**
Monocular Inertial	MCSKF [33]	ATE[5]	0.420	0.450	0.230	0.370	0.480	0.340	0.200	0.670	0.100	0.160	1.130	0.414
	OKVIS [39]	ATE[5]	0.160	0.220	0.240	0.340	0.470	0.090	0.200	0.240	0.130	0.160	0.290	0.231
	ROVIO [42]	ATE[5]	0.210	0.250	0.250	0.490	0.520	0.100	0.100	0.140	0.120	0.140	0.140	0.224
	ORBSLAM-VI [4]	ATE[2,3] scale error[2,3]	0.075 0.5	0.084 0.8	0.087 1.5	0.217 3.5	0.082 0.5	**0.027** 0.9	0.028 0.8	- -	**0.032** 0.2	0.041 1.4	0.074 0.7	0.075* 1.1*
	VINS-Mono [7]	ATE[4]	0.084	0.105	0.074	0.122	0.147	0.047	0.066	0.180	0.056	0.090	0.244	0.110
	VI-DSO [46]	ATE scale error	**0.062** 1.1	0.044 0.5	0.117 0.4	0.132 0.2	0.121 0.8	0.059 1.1	0.067 1.1	0.096 0.8	0.040 1.2	0.062 0.3	0.174 0.4	0.089 0.7
	ORB-SLAM3 (ours)	ATE scale error	**0.062** 1.4	**0.037** 0.3	**0.046** 0.8	**0.075** 0.5	**0.057** 0.3	0.049 2.0	**0.015** 0.6	**0.037** 2.2	0.042 0.7	**0.021** 0.4	**0.027** 1.0	**0.043** 0.9
Stereo Inertial	VINS-Fusion [44]	ATE[4]	0.166	0.152	0.125	0.280	0.284	0.076	0.069	0.114	0.066	0.091	0.096	0.138
	BASALT [47]	ATE[3]	0.080	0.060	0.050	0.100	**0.080**	0.040	0.020	0.030	**0.030**	0.020	-	0.051*
	Kimera [8]	ATE	0.080	0.090	0.110	0.150	0.240	0.050	0.110	0.120	0.070	0.100	0.190	0.119
	ORB-SLAM3 (ours)	ATE scale error	**0.036** 0.6	**0.033** 0.2	**0.035** 0.6	**0.051** 0.2	0.082 0.9	**0.038** 0.8	**0.014** 0.6	**0.024** 0.8	0.032 1.1	**0.014** 0.2	**0.024** 0.2	**0.035** 0.6

Data association overpowered (OP!)

Regarding accuracy, the capability of exploiting short-term, mid-term, long-term, and multimap data associations overpowers other choices such as using direct methods instead of features or performing keyframe marginalization for local BA, instead of assuming an outer set of static keyframes as we do.

Stereo-intertial config best, yay

About the four different sensor configurations, there is no question; stereo-inertial SLAM provides the most robust and accurate solution

Future Works

Use of LLMs in context of the DBoW2 generated MAP Atlas Maps
Working on low-textured datasets for mapping with less available key-point features

Newer work in SLAM

use of detectron2 for eliminating dynamic frames strategically: https://arxiv.org/abs/2210.00278
use of nerf and gaussian splatting for better a-posteriori maps

Thank you for listening!