Densifying Sparse VIO: a Mesh-based approach using Structural Regularities

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Massachusetts Institute of Technology

Antoni Rosiñol Vidal

Densifying Sparse VIO
A mesh-based approach using structural regularities

Master Thesis

Computer Vision and Geometry Group
Swiss Federal Institute of Technology Zurich*

Sensing, Perception, Autonomy, and Robot Kinetics
Massachusetts Institute of Technology‡

Supervision

Dr. Torsten Sattler*
Prof. Dr. Marc Pollefeys*
Prof. Dr. Luca Carlone‡

September 14, 2018

Table of Contents

List of Figures

List of Tables

Abstract

The ideal vision system for an autonomous robot would not only provide the robot's position and orientation (localisation), but also an accurate and complete model of the scene (mapping). While localisation information allows for controlling the robot, a map of the scene allows for collision-free navigation; combined, a robot can achieve full autonomy.

Visual Inertial Odometry (VIO) algorithms have shown impressive localisation results in recent years [2,3]. Unfortunately, typical VIO algorithms use a point cloud to represent the scene, which is hardly usable for other tasks such as obstacle avoidance or path planning.

In this work, we explore the possibility of generating a dense and consistent model of the scene by using a 3D mesh, while making use of structural regularities to improve both mesh and pose estimates. Our experimental results show that we can achieve a 26% more accurate pose estimates than state-of-the-art VIO algorithms when enforcing structural constraints, while also building a 3D mesh which provides a denser and more accurate map of the scene than a classical point cloud. We also show that our approach does not rely on assumptions about the scene and is general enough to work when structural regularities are not present.

Nomenclature

Symbols

$\mathcal{K}_t$	Set of keyframes up to time t	14
SE(3)	Special Euclidean Group	14
$\mathcal{L}_t$	Set of landmarks seen up to time t	14
$\mathcal{X}_t$	State vector	14
$\boldsymbol{\rho}_l$	Landmark's 3D position	14
Λ_t	Set of landmarks with regularity constraints	14
Π_t	Set of planes detected	15
$\mathcal{C}_i$	Image Measurements at Keyframe i	15
$\mathbf{z}_i^{l_c}$	Landmark measurement with structural regularities	15
$\mathbf{z}_i^{l_s}$	Landmark measurement without structural regularities (structureless)	15
$\mathcal{Z}_t$	Set of measurements collected up to time t	15
$\ominus$	Inverse Compositional Operator	43
$\|A\|_{\mathrm{F}}$	Frobenius norm	43
$\mathcal{F}$	Set of Frames	49

Indicies

$trans$	Translational part	49
rot	Rotational part	50

Acronyms and Abbreviations

VIO	Visual Inertial Odometry	5
EKF	Extended Kalman Filter	5
MSCKF	Multi-State Constraint Kalman Filter	5
ROVIO	RObust Visual Inertial Odometry	5
$\sqrt{SAM}$	square root Smoothing And Mapping	6
OKVIS	Open Keyframe-based Visual Inertial SLAM	6
SDF	Signed Distance Function	9
TSDF	Truncated Signed Distance Field	9
JCBB	Joint Compatibility Branch and Bound	11
MAP	Maximum A Posteriori	16
MAR	Median Absolute Residual	27
EuRoC	European Robotics Challenge	41
MAV	Micro Aerial Vehicle	41
APE	Absolute Pose Error	42
RMSE	Root Mean Squared Error	43
RPE	Relative Pose Error	49
TUM	Technical University of Munich	50

Chapter 1

Introduction

1.1 Motivation

Visual Inertial Odometry (VIO) algorithms combine the visual information from the images of a camera with the accelerometer and gyroscope readings from an Inertial Measuring Unit (IMU) to achieve accurate state estimates. A camera and an IMU are especially interesting sensors in combination as they have complimentary features.

An IMU provides high-frequency motion information, which is accurate at short intervals of time, but drifts over time. Instead, a camera provides images at smaller frequencies, but allows for accurate estimations of the motion long term. In the last decade, multiple real-time and highly-accurate VIO algorithms have been developed such as [4] or [5]. Unfortunately, these systems have nowadays two limitations that we want to address in this thesis, and which we detail below.

1.1.1 Enhancing sparse point clouds

Classical VIO approaches use a sparse point cloud to represent the environment. This representation, although practical because it allows efficient and accurate inference, is nevertheless of limited use. Indeed, a sparse representation of the scene removes large amounts of information about the environment that is essential for other important tasks. In the field of autonomous systems, a sparse point cloud is not suitable for obstacle avoidance or path planning, where a dense map is necessary instead. In augmented reality, it would be more useful to have a map that captures the geometry of the objects in the scene to allow the seamless projection of virtual elements. To this effect, we want to make the estimated map by a VIO algorithm more amenable to other potential applications.

1.1.2 Enforcing structural regularities

Common man-made indoor scenes are characterized by strong regularities such as parallel and planar walls, which are not used as prior information in classical VIO pipelines. Nevertheless, using this information could potentially improve both the quality of the map and the accuracy of the state estimation. Our hypothesis originates from the simple observation that landmarks estimated by a VIO pipeline are not consistent with the underlying geometry of the scene. When running a VIO pipeline over a planar surface, landmarks do not seem to lie on a single plane, but

they are instead scattered approximately around the planar surface – with some landmarks even deviating significantly from the surface. Therefore, we will also explore the influence of using prior information about the structure of the scene on the overall performance of a VIO pipeline.

1.2 Approach

In the literature, few implementations can achieve accurate pose estimates and simultaneously build a map useful for other tasks than localization [6–8]; they rely nevertheless on RGB-D cameras that provide dense estimates of the scene. These approaches suffer from high computational loads since they use all pixels in a camera, which leads to vast amounts of information having to be processed. Therefore, they often require the use of dedicated hardware like GPUs – which consume relatively high amounts of power, thereby making these algorithms not applicable to computationally constrained systems such as Micro Aerial Vehicles (MAV).

In this respect, the motivation of this thesis is to develop a new VIO pipeline for computationally restricted systems that provides accurate state estimation while mapping the environment densely and precisely.

We will use sparse keypoint-based visual inertial odometry, with regular RGB cameras, to estimate the scene in a dense manner, while achieving accurate state estimates, using off-the-shelf CPUs.

To achieve these results, we perform a 2D Delaunay triangulation over the 2D keypoints in the image, and project the triangulation as a mesh in 3D. We then use the 3D mesh to detect regularities in the scene: such as points lying on the same plane or high-level regularities between planes (parallel planes, like the floor and a table top, or orthogonal planes, such as vertical walls and the floor) typically found on man-made environments. We then encode these regularities as constraints in a factor-graph; which has as variables, among others, the camera poses, the landmarks' positions and the planes observed.

In order to achieve real-time performance, we frame the optimization problem in an incremental fashion, and we bound the problem to a fixed time-horizon. Such an approach is commonly referred in the literature as Fixed-lag Smoothing, and it is used in several state-of-the-art VIO pipelines [3, 9].

After optimization of the factor graph, we obtain both accurate pose estimates of the camera, and a mesh-based representation of the scene.

Recent work with similar objectives as us have so far worked in a decoupled way. For example, [10] estimates a mesh of the environment assuming given ground truth estimates of the position and orientation. [11] also relies on an external algorithm for state estimation and mapping, while they focus on building the mesh. In our case, we want to estimate both the mesh and the state in a tightly coupled manner with the expectation of making both pose and map estimates more accurate.

1.3 Thesis outline

Concerning the strucutre of the thesis, we will first present a review of previous work related to the subject of visual inertial odometry, dense mapping and the usage of structural regularities in state estimation, thereby covering the state-of-the-art. Then, we will present the mathematical formulation of our approach, providing details on both the structure of the backend and the

frontend of our VIO. Finally, we will present the experimental results achieved, and conclude with the insights that we have gained in the process.

Chapter 2

State of the art

Since we are not simply trying to improve the accuracy of the pose estimation in VIO, but also tackle the problem of dense map estimation, we found convenient to do a literature review of the fields of Visual Inertial Navigation (§ 2.1) and dense mapping (§ 2.2). We also review previous work using structural regularities in § 2.3.

2.1 Visual Inertial Navigation

In this work, we focus on Visual Inertial Odometry (VIO) algorithms. Using Visual and Inertial sensors for state estimation has been extensively studied over the past decades, leading to many different types of implementations. We can segment these implementations depending on the type of optimization used (backend). In particular, we distinguish three types of VIO algorithms based on the number of state variables that are being estimated:

- **Filtering** based methods are the oldest way of solving the problem of state estimation, and consist in finding the optimal state for the most recent time where sensor data is available. Initial work was based on the Extended Kalman Filter (EKF), used by Smith et al. [12]. The EKF simply estimates a Gaussian density over the last state in a recursive fashion; where the state typically includes the current pose of the robot and the observed 3D landmarks. Nevertheless, the computational complexity of the EKF grows quadratically with the number of variables used in the state, and it becomes quickly intractable when dealing with many landmarks [13]. This is because the underlying uncertainty representation of the EKF is the actual covariance matrix, which does not stay sparse (unlike the information matrix in smoothing), and requires dense matrix operations instead of the much more efficient sparse methods [14]. To avoid this problem, the Multi-State Constraint Kalman Filter (MSCKF) [15] makes use of a measurement model that avoids including landmarks in the state, therefore reducing significantly the number of state variables. While we will not be using a filtering method, we will also reduce the number of variables from the optimization problem by removing landmarks positions from the state vector as MSCKF [15].

  Another issue with filtering is that it is shown to become inconsistent when using nonlinear models as in SLAM [16]. This is caused by introducing linearization inaccuracies that cannot be corrected after marginalization. To counter such linearization errors, care must be taken to reject outlier measurements [17]. The open-source Robust Visual Inertial Odometry (ROVIO) algorithm [18] achieves good results by being robust to bad initial guesses and image blur.

While we will not be using a filtering approach, both [15] and [18] achieve state-of-the-art performance in VIO, and therefore we ought to compare them with our approach. The reader can find more information about filtering based methods in the survey [19].

- **Full-smoothers** use instead all the data collected until the last sensor measurements. Hence, they attempt to solve a large optimization problem each time new sensor measurements are available (incremental methods) or after collecting all measurements (batch methods) [20,21]. This technique is referred to, in the field of photogrammetry, as "bundle adjustment" [22], and in computer vision as "structure from motion" [23]. These methods avoid marginalizing the states, achieving an accurate state estimation that might include not only the robot trajectory, but also all landmarks. Nevertheless, the optimization problem rapidly becomes intractable, as the number of variables grows unbounded. One way that state-of-the-art methods try to tackle this problem is by using *keyframes* – selected frames that provide more and better information than merely using consecutive frames – and discard redundant frames [3, 24]. Our approach will also rely on keyframes to remove redundant information and speed-up the inference time.

  keyframes

  On the backend side, it has been shown that using the information matrix and exploiting its sparsity structure is a better approach than using an EKF [25, 26]. Since this approach is based on factorizing the information matrix on a square-root form, they are referred to as square root SAM (or $\sqrt{SAM}$). Kaess et al. [27] proposed a fast incremental square-root SAM that effectively updates the square-root information matrix by reusing previous computations. It was nevertheless necessary to perform batch steps to keep the consistency of the estimation. To solve this issue, the same authors first presented the use of the Bayes tree [28], and further exploited this novel data structure to develop iSAM2 [29], a state-of-the-art incremental exact inference method. Their Bayes tree representation allows relinearization of a small set of variables, therefore providing a boost in efficiency while keeping the sparse structure of the problem. SVO-GTSAM [2] made use of iSAM2 combined with SVO's frontend [30] (a semi-direct visual odometry algorithm) to achieve state-of-the-art results in VIO. In our approach, we will also use iSAM2, but contrary to the full-smoothing approach of SVO-GTSAM, we will be using a fixed-lag smoothing technique. We will also compare our approach with SVO-GTSAM.

- **Fixed-lag smoothing** makes use of marginalization to avoid the problem of an ever increasing number of variables. Marginalization removes older variables without removing information, which allows for reducing the estimation problem to the variables that are within a fixed time window, also known as sliding-window or time-horizon [31–34]. More specifically, a fixed-lag smoother proceeds recursively by interleaving two operations: a measurement update, followed by a marginalization step. Fixed-lag smoothers hence trade accuracy for efficiency depending on the size of the time window used. It can actually be shown that the standard Kalman filter corresponds to using a fixed-lag smoother with a window size including only the last state [14].

  Marginalization

  Open Keyframe-based Visual Inertial SLAM (OKVIS) [3] makes use of a fixed-lag approach were keyframes older than a given time are marginalized. OKVIS has achieved the best results for fixed-lag smoothing approaches for a long time, although new methods have shown excellent results as well; such as VINS-MONO [9], which makes use of the IMU pre-integration theory of [2] to improve on OKVIS (besides introducing other new features). In our case, we will also use the pre-integration theory of Forster et al. [2]. Since both OKVIS and VINS-MONO achieve state-of-the-art performances, we will compare our approach to these algorithms as well.

Delmerico et al. [1] have taken on the important and time-consuming task of benchmarking the above mentioned state-of-the-art monocular VIO algorithms for the scientific community. We will use their results to compare our approach in the EuRoC MAV dataset in chapter 4.

2.2 Dense Mapping

Retrieving a dense map of the scene has always been a tempting endeavour in computer vision because of the multiple uses a faithful map has. In robotics, dense maps are a valuable asset since they can be used to avoid obstacles [35], to plan optimal paths [36], and even to allow manipulation of objects in the scene [37, 38]. Dense maps also provide means for visual inspection of structures [11] as they are visually appealing [39].

As we have seen previously, SLAM has had a great deal of success with a flurry of works achieving unprecedented results [2, 3, 9, 15, 18]. The performance of SLAM sparked a renewed interest to achieve dense scene reconstructions instead of sparse point clouds that SLAM pipelines use for accurate localization. While the first works making use of the newly acquired accuracy of SLAM for dense mapping consisted in using a 3D mesh built over the sparse 3D landmarks [35, 40], the scientific community focused later instead into "every pixel" methods. One of the earliest real-time dense reconstruction systems is the one from Pollefeys et al. [41], who achieved dense reconstructions of urban scenes from a rig with eight cameras mounted on a vehicle, having as well a GPS and an IMU. Their approach uses real-time structure-from-motion combined with plane sweep stereo and depth map fusion [42]. As a last step, they still used a multi-resolution triangular mesh to generate a model of the scene, by using a quadtree[1]. Newcombe et al. [6], by re-using the parallel architecture of PTAM [44], proposed a dense reconstruction algorithm performing real time depth estimation for each pixel using Total Variation regularisation. They also used a mesh to have a first approximation of the depth of the scene, which they then warp into a depth map. Impressive results were also achieved using RGB-D cameras, for example KinectFusion [8] demonstrated a real-time dense mapping using the Kinect camera. DTAM [7] was nevertheless one of the first to prove that online dense reconstruction is possible using standard cameras. Unfortunately, all these approaches necessitate GPUs, since they are computationally demanding. This constraint hindered their applicability because GPUs require significantly more power than CPUs, making it difficult to apply such alogrithms in computationally constrained platforms such as MAVs or smartphones.

Recently, new approaches have tried to avoid the caveats of an every-pixel approach, which oversamples simple geometries such as planes. Indeed, trying to perform dense stereo matching seems to be unnecessary for simple geometries, such as planar surfaces, where depth at each pixel might be redundant, or just too difficult to find due to a lack of texture. The alternative to every-pixel approaches seems to go in two directions:

- **Semi-dense approaches**, that try to use direct methods (feature-less) or attempt to densify feature-based algorithms. Semi-dense methods have been in vogue during the past years as they provide a good trade-off between computational performance and denser maps.

  In an attempt to avoid the use of GPUs, Engel et al. [45] proposed LSD-SLAM, a multi-threaded algorithm running on CPUs that achieves semi-dense maps by performing direct photometric alignment of images to estimate the depth and the position of the camera. Their approach effectively uses all the information in the image which allows for capturing most of the scene.

  Alternatively, the authors of ORB-SLAM [46] tried to densify their feature-based pipeline and proposed a probabilistic semi-dense mapping [47]. Their approach uses a parallel thread to their SLAM algorithm and computes the depth estimates using the same stereo correspondence search and inverse depth uncertainty derivation as [48], but on a per-keyframe basis, instead of on consecutive frames.

  The debate about whether using direct methods or feature-based methods is nevertheless still

[1] Their mesh did not keep the manifold property (defined below), which would require the use of a restricted quadtree [43], but that would hinder processing speed.

open; and it seems that one might be better than the other depending on the applications, with some authors combining even both [49].

Other authors try to achieve ever denser maps by densifying semi-dense maps from direct SLAM pipelines. DPPTAM [50] achieves a dense reconstruction of the scene by combining the natural output of a direct monocular SLAM pipeline, meaning a semi-dense map, and adding piecewise planar low-gradient regions (detected using superpixel segmentation) to the map. They rely nevertheless on the assumption that low-gradient regions correspond to planar surfaces, which seems to be reasonable for many man-made environments.

- **Mesh-based approaches**, on the other hand, start from sparse features and use some form of tesselation or meshing to recover the topology of the scene. Parallel to semi-dense methods, mesh-based methods have seen a renewed interest over the last years.

  Teixeira [11] and Greene [10] both use sparse features and accurate camera poses from a SLAM pipeline, but, contrary to [47], they use a 3D mesh generated by doing a 2D delaunay triangulation over the image keypoints and subsequently projecting the mesh in 3D. The resulting mesh accurately captures the scene at the current camera viewpoint, and leads to very fast and scalable algorithms, as also acknowledged by other works [51]. We have therefore followed this same approach, although our 3D mesh does not only cover the currently visible frame as in [11], but rather the whole scene observed by all the keyframes present in the time-horizon of the optimization (we use a fixed-lag smoothing approach, thereby optimizing over multiple keyframes simultaneously).

  The issue with meshes built this way is that they need to be regularised, otherwise they would be too noisy or have unwanted artifacts. Teixeira [11] therefore implements a Laplacian smoother over the mesh to remove outlier vertices. Greene et al. [10] proposed to use a Non-Local Second Order Total Generalized Variational ($NLTGV^2$) regularisation cost over the inverse depth of the vertices of the mesh, thereby efficiently regularising the mesh so that it keeps planar surfaces while preserving edges. Greene et al. were inspired by previous work from Piniés [52] who tackled the problem of dense reconstruction from an every-pixel approach instead of a sparse feature approach; showing that what is learned from dense approaches can be applied to sparse featured-based methods, and potentially the other way around.

  In our case, we also need to regularise the mesh. For this, we used simple outlier filters detailed in § 3.2.1, similarly to Teixeira, although we did not use the same filters. We also use non-local regularisation as Greene et al., but contrary to Greene, we encoded the regularisation term[2] directly on the same optimization problem solved in SLAM, instead of decoupling SLAM and mesh regularisation.

  Indeed, none of the algorithms presented above seem to have tackle the optimization of the pose estimate and the dense reconstruction in a tightly couple manner were the mesh is used to improve as well the SLAM performance. In this respect, KinectFusion [7] seems to be one of the few that use their dense model to calculate the pose of the camera using ICP.

  Rather than making the densification of the point cloud and the SLAM algorithm agnostic of each other, our approach is to close a feedback loop between mesh estimation and SLAM. We do this by first generating a mesh over the sparse features of the SLAM algorithm, as most of the authors above, but we subsequently use the mesh to enforce structural constraints for the SLAM problem, thereby improving both the localization and the reconstruction of the scene.

  In order to keep real-time performance, we decided to proceed as Teixeira et al. by performing a 2D Delaunay triangulation and projecting it in 3D. Nevertheless, our 3D mesh captures all 3D landmarks that are present in the time-horizon of our optimization problem, thereby describing a larger portion of the map, instead of limiting the 3D mesh to the current frame.

[2] We will not use the appellation "regularisation term", instead we will refer to it as structural regularity constraints, in § 3.1.5.

The methods that we have mentioned so far represent just one branch of possible solutions to achieve dense reconstructions. There is also a volumetric approach which deals with dense reconstructions by discretizing the 3D space. We will not be using a volumetric approach, nevertheless it is instructive to see what are their benefits and drawbacks. Here, we discuss two types of volumetric approaches:

- 3D Delaunay: Some works attempt to build a 3D Delaunay tetrahedralisation directly from the sparse 3D landmarks. Incrementally building a mesh from sparse 3D points is nevertheless non-trivial. The estimated 3D points are typically noisy, some are outliers, and their density is highly irregular. On top of this, the point cloud is growing over time. Most methods extract surfaces from sparse point clouds by labeling what is "outside" or "inside", alternatively what is "free" versus "occupied", by using visibility information. The actual surface corresponds therefore to the boundary between free and occupied space. A 3D Delaunay triangulation on the 3D points provides an irregular discretization of the space, which is advantageous since it adapts nicely to the underlying density of the point cloud. The difference between algorithms using this approach resides mainly in the way free and occupied space is classified. Unfortunately, this operation typically requires costly space carving algorithms [53].

  The most ambitious attempts also try to enforce the 3D mesh to have the manifold property. A mesh is said to be a manifold if each vertex v is regular, which is the case if and only if the edges connecting the vertices opposite to v are homeomorphic to a disk, or in other words, if they form a closed path without loops [54]. This property is useful to improve the mesh for smoothing [55] or photometric optimization [56]. While some works have enforced manifoldness incrementally [57, 58], these approaches are not able to run in real-time. Nevertheless, recent approaches have achieved impressive results. Piazza et al. [59] improve the work of [57, 58] by making it run in (near) real time and on a single CPU core, while reaching similar accuracy as the original algorithms and maintaining the manifold property of the mesh as well. They achieved these results by changing certain parts of the original algorithms (shrinking process and ray tracing), yet they achieve a boost in processing time, while also making the algorithm work for points whose positions are being refined, as it occurs in SLAM.

- Zero set methods: Which use a distance function that gives each point in the space a signed distance to the closest surface, thereby gaining the title of "implicit" surface representations). Different types of functions can be used: radial-basis functions [60], signed distance functions [61], and truncated signed distance functions [62].

  Signed Distance Functions (SDF) represent surfaces as the set of 3D points whose coordinates zero out a given function. An SDF represents free space as positive values that increase with the distance to the closest surface, while potentially occupied space is represented with negative values.

  To avoid volumes interfering each other, it is necessary to truncate the SDF to a truncation radius around the boundary of the surface as detailed in [63]. Truncated SDFs (TSDF) are a useful implicit surface representation for dense reconstruction as they are relatively fast to build, offer increased robustness to sensor noise and can be used to generate interpretable meshes with high resolution [7]. TSDFs became popular with KinectFusion [7]; which used a TSDF as a volumetric representation of the scene in order to achieve an accurate and high resolution 3D reconstruction in real-time (using a GPU), while also being able to track the camera with the built 3D model using a coarse-to-fine ICP algorithm. A voxel-based representation is nevertheless a redundant way of defining continuous surfaces and empty space, which limit the applicability of TSDFs to large scale 3D mapping. Nevertheless, following KinectFusion, other works focused on expanding to larger scenes, such as Kintinuous [64] (by using a TSDF only locally), and also managed to reduce the computational load [65] (by using octrees) associated to SDF representations.

There are other mapping representations that are difficult to classify using the labels above. For example, RGB-D SLAM [66] uses surfels[3] [67], which are small planar surfaces with color, and are an alternative to polygonal modelling or point clouds. Although Bylow et al. [68] show that using TSDF-maps it is possible to reach the same level of accuracy as RGB-D SLAM. Alternatively, dense reconstructions can be achieved on the level of objects. Salas-Moreno et al. [69] demonstrate a SLAM system where landmarks are actually objects such as chairs and tables. Their system, named SLAM++, uses prior dense models to represent the landmark objects in the map.

It is also interesting to see that Krüsi et al. [70] have tried to bypass the recovery of a dense map, and instead considered using a raw point cloud, for path planning of a ground robot; therein omitting a method to extract the topology of the scene, and opening questions on whether it is really necessary to do so. Their approach nevertheless still requires a rather dense point cloud provided by a LIDAR.

2.3 Structural Regularities

In the SLAM literature, it is well-known that adding high-level features in the optimization problem leads to increased localization and mapping accuracy [71–74]. There are many types of high-level features that can be used. Common features are lines [75, 76], planes [77, 78], vanishing points [79, 80], quadrics[4] [81, 82], and even full 3D objects [69, 83]. Each feature has its own benefits and associated difficulties [84]. In our case, we will focus on detecting co-planar landmarks by extracting potential planes, therefore, we will just review SLAM algorithms using planes.

Planes are coincidently the most common high-level feature used both for its simple parametrisation and descriptive power. Planar regularities are common in man-made environments: such as in urban scenes with roads and buildings, or indoor scenes with corridors. Unfortunately, planes are usually textureless, therefore, most often than not, SLAM algorithms working with planes rely on RGB-D cameras, which use structured light to extract depth information. For this reason, while we are using a standard camera, most of the literature we will review uses RGB-D cameras.

Weingarten et al. [85] presented one of the early works that used planes in the optimization problem. They used an EKF approach for inference, which, as we have seen previously in § 2.1, has the drawback that the computational cost increases quadratically with the number of variables used, thereby limiting the approach to a few plane estimates. Nevertheless, the intrinsic limitation of the EKF pushed some authors [86] to device methods to reduce the size of the state space by collapsing redundant point features that could be explained instead by a plane. While subsequent works still used an EKF approach [87–89], most recent works use instead a graph formulation [74, 90, 91]. By using a fixed-lag smoother, we follow as well this last approach.

Besides the optimization framework used, SLAM algorithms using planes can differ on the type of plane parametrization used. Parametrizing a plane is rather simple, but there are multiple approaches to do so with their own benefits and caveats:

- **Spherical parametrization**. Lee et al. [71] use a spherical parametrization that uses three parameters to describe a plane, which is the minimum number of parameters necessary. These are the plane's orientation given in altitude and azimuth (in spherical coordinates), and the distance of the plane to the origin. Despite its common usage [85, 87], this type of parametrization suffers from singularities similar to the feared gimbal lock for Euler angles. Singularities complicate the optimization problem when encountered.

[3] Surfel is an abbreviation for "**surf**ace **el**ement".
[4] Generalization of conic sections (ellipses, parabolas and hyperbolas) to higher dimensions.

- **Non-homogeneous parametrization (Normal and distance)**. Another common parametrization [77,92] consists in using four parameters: three to describe the plane's normal and one to define the distance of the plane from the origin. Since a plane only has three degrees of freedom, the normal and distance represent an over-parametrization of a plane. Unfortunately, such over-parametrizations lead to a singular information matrix. Hence, a Gauss-Newton optimization will fail (since we need to invert the information matrix), and a Levenberg-Marquardt optimization would need to be used instead, which adds a regularization term that would slow convergence speed.
- **Homogeneous parametrization**. The plane is just represented by an homogeneous vector $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)^T \in \mathbb{P}$ in projective space [23]. This is as well an over-parametrized representation.
- **Minimal homogeneous parametrization**. Kaess [93] presented an homogeneous plane parametrization which is minimal, yet it does not present singularities. This parametrization is therefore especially suitable for incremental solvers such as iSAM [27].

Similarly to Trevor et al. [92], we will be using a non-homogeneous parametrization for our plane estimates. In our case, it will allow us to express the distance from a point to a plane in a neat way, resulting on a mathematically simple jacobian, while also avoiding undesirable singularities. This will result in an over-parametrization, but we will solve a constrained optimization problem using the properties of manifolds (§ 3.1.7)to overcome the associated problems with over-parametrizations.

Even if multiple methods might share the type of optimization framework used and the type of plane parametrization, they might differ on the way planes are detected or associated. Trevor et al. [92] uses a Joint Compatibility Branch and Bound (JCBB) technique for data association as in [94]. To reduce the computational burden of plane detection, [78] proposed to do plane tracking. Nevertheless, faster plane extraction algorithms were presented in [95] and [96] (using RGB-D cameras).

While we are using a standard camera, by being able to extract 3D landmark information, we can detect planes by observing the 3D structure instead of relying solely on 2D images of the scene. We will nevertheless use simple 2D histograms over the set of mesh faces to detect the orientation and distance to the origin of walls (assuming they are vertical) and a 1D histogram to detect horizontal surfaces, thereby extracting the most common planes in urban and indoor scenes. With this approach, we can also avoid expensive iterative plane fitting methods such as RANSAC [97]. Histograms have been used in other works as well as an effective way to get rough estimates of the underlying structure in a scene [98].

Concerning plane association, it is common to just compare normals and distances to the origin between planes, and decide if both planes correspond to the same entity depending on the difference between their normals and distances to the origin. This is the approach we will follow. Other authors might add extra criteria for robustness, for example Hsiao et al. [91] presented a plane association algorithm which also checks that the original plane should describe the points of the new plane accurately.

Finally, using solely planes is difficult since the pose of the camera can be easily underconstrained if not enough planes are observed (at least three planes must be observed per pose). This is the major problem in formulations such as the one from Kaess [93]. Nevertheless, [93] was eventually extended to work with visual odometry constraints [91], and later with inertial constraints as well [74]. Points and planes were combined in [77] to also avoid underconstrained variables. In our case, we will avoid such issues by always relying on points; planes will be only used as structural information about the geometry of these points.

There are few works that ressemble nevertheless to ours in terms of the optimization problem

they try to solve.

On the one hand, Hsiao et al. [74] is close to ours in that it uses inertial constraints while using planes in the optimization problem. Our work instead does not encode planes as landmarks that we observe, but rather as high-level constraints that provide information about the relative position of the 3D point landmarks (in particular co-planarity constraints between these point landmarks). This difference is natural; Hsiao et al. use a RGB-D camera that makes plane observations straightforward, while we use a standard camera that works best by extracting keypoints and inferring sparse 3D landmarks. In this sense, our work is the first, to the best of our knowledge, to use inertial and co-planarity constraints in the same optimization problem. Moreover, we seem to be the first to have used structureless, inertial and stereo/mono reprojection constraints in the same formulation; and, by induction, the first to use this formulation with co-planarity constraints as well.

On the other hand, co-planarity constraints in SLAM have been investigated in a limited extend, yet insightful way, in [73], and have been applied in multiple setups [71, 99]. While [99] formulated the co-planarity constraint between landmarks and planes on the image space because of the scale ambiguity of monocular SLAM, we will formulate the co-planarity constraint geometrically as in [71].

As an end note, our pipeline is versatile in that it can work on unstructured environments, and does not rely on assumptions about the world (such as the commonly used Manhattan World assumption [72]). Nevertheless, it is only capable to detect structural regularities of two types due to its simple frontend: planes that are vertical, such as walls, and planes that are horizontal, such as the floor, the ceiling or the surface of tables. Hence, if the scene has structural regularities of this type, our approach will capture and encode them gracefully in the optimization problem, and, if no structural regularities were to be present, it will downgrade to a classical VIO.

Chapter 3

Mathematical Formulation

The VIO algorithm we propose in this thesis shares the same structure than classical VIO pipelines. In essence, it has two modules that work in a sequential manner:

(i) A *backend* that builds an optimization problem with the variables to estimate. In our case, we will be using a state-of-the-art VIO backend, similar to [2], which we will extend to encode a new set of constraints and variables necessary to add information about the structural regularities of the scene. We explain in § 3.1 the mathematical formulation of these new constraints and variables.

(ii) A *frontend* that processes the raw measurements from the sensors to extract meaningful information about the state of the system and the scene. In our case, we are receiving a pair of images from the stereo camera, as well as acceleration and angular velocities from the IMU. In § 3.2, we explain how we process these measurements, and how we build a 3D mesh to estimate the geometry of the scene.

3.1 Backend

A VIO algorithm tries to infer the state of a system, such as a robot, equipped with two sensors: a camera and an IMU.

It is common in the literature to define as the state of the robot the actual position and orientation of the IMU sensor (mainly to avoid unnecessary transformations between frames of reference) [2]. In our case, we further simplify the formulation by assuming that the IMU frame and the camera frame is fixed and that we know it a priori. The actual transformation between IMU and camera is typically found through calibration, using for example Kalibr [100].

We define the state of the system in a standard way, following the same notation and formulation of [2], which we will later extend to incorporate structural regularities.

3.1.1 State of the system

If we denote the set of all keyframes up to time t by $\mathcal{K}_t$, the state of the system $\mathbf{x}_i$ at keyframe $i \in \mathcal{K}_t$, is described by the IMU orientation ($\mathtt{R}_i$), position ($\mathbf{p}_i$), velocity ($\mathbf{v}_i$) and biases ($\mathbf{b}_i$):

$$\mathbf{x}_i \doteq [\mathtt{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_i], \tag{3.1}$$

where the pose $(\mathtt{R}_i, \mathbf{p}_i) \in \mathrm{SE}(3)$, $\mathbf{v}_i \in \mathbb{R}^3$, and $\mathbf{b}_i = [\mathbf{b}_i^g \quad \mathbf{b}_i^a] \in \mathbb{R}^6$, where $\mathbf{b}_i^g, \mathbf{b}_i^a \in \mathbb{R}^3$ are the gyroscope and accelerometer bias, respectively.

At this point, we combine two different ways of defining the map of the scene in the state vector. Let us start first introducing the most classical way of representing the map in § 3.1.1. We will then proceed with the other more recent, and efficient, way in § 3.1.1. Finally, in § 3.1.1, we formulate our combined state vector, which we will use for the remaining of the thesis.

Classical state formulation

Typically, the way the map is represented in a landmark-based VIO pipeline is by simply adding the 3D positions $\boldsymbol{\rho}_l \in \mathbb{R}^3$ of each landmark $l \in \mathcal{L}_t$ in the state vector; where we define as $\mathcal{L}_t$ the set of all the landmarks seen up to time t. This results in a state vector of the following form:

$$\mathcal{X}_t \doteq \{\mathbf{x}_i, \boldsymbol{\rho}_l\}_{i \in \mathcal{K}_t, l \in \mathcal{L}_t} \tag{3.2}$$

For example, OKVIS [3] formulates the state vector similarly to eq. (3.2).

Structureless state formulation

Alternatively, one can use a structureless approach, where the landmarks are not included to the state vector. This is convenient because the optimization's complexity increases with the number of variables to be estimated. Hence, using a structureless approach avoids optimizing over the landmarks, thereby reducing the time to converge to a solution. In § 4.4 we will see the actual impact in computational performance of one approach over the other.

Mathematically speaking, this simplification is possible through the Schur complement trick, which we restrain from explaining here by referring the reader to [2]. This approach is for example used in MSCKF [15] to make their filtering based optimization tractable.

Combined state formulation

In our approach, we will also estimate the 3D positions $\boldsymbol{\rho}_l$, but we will only do so for a subset Λ_t of all landmarks $\mathcal{L}_t$ visible up to time t: $\{\boldsymbol{\rho}_l\}_{l \in \Lambda_t}$, where $\Lambda_t \subseteq \mathcal{L}_t$. We will avoid optimizing over the rest of the landmarks $\mathcal{L}_t \setminus \Lambda_t$ by using a structureless approach as detailed in [2, Sec. VII].

The set Λ_t corresponds to the landmarks which we consider to satisfy a structural regularity, such as co-planarity between landmarks. We hence use a non-structureless approach for these landmarks, because we will need the explicit landmark variables to formulate contraints between them. The resulting state vector remains as eq. (3.2), but the modelled landmarks are on Λ_t instead of $\mathcal{L}_t$:

$$\mathcal{X}_t \doteq \{\mathbf{x}_i, \boldsymbol{\rho}_l\}_{i \in \mathcal{K}_t, l \in \Lambda_t} \tag{3.3}$$

State vector for structural regularities

Finally, we will also use the subset of landmarks Λ_t to estimate planar surfaces. We define the set of planes detected as Π_t.

Therefore, the variables to be estimated comprise the state of the system $\{\mathbf{x}_i\}_{i\in\mathcal{K}_t}$, the landmarks which we consider to satisfy structural regularities $\{\boldsymbol{\rho}_l\}_{l\in\Lambda_t}$, and the planes $\{\boldsymbol{\pi}_\pi\}_{\pi\in\Pi_t}$. The variables to be estimated at time t are:

$$\mathcal{X}_t \doteq \{\mathbf{x}_i, \boldsymbol{\rho}_l, \boldsymbol{\pi}_\pi\}_{i\in\mathcal{K}_t, l\in\Lambda_t, \pi\in\Pi_t} \tag{3.4}$$

Landmarks are represented in homogeneous coordinates as in [101, Ch. 3.4], in order to allow seamless integration of close and very far landmarks: $\boldsymbol{\rho}_l \in \mathbb{R}^4$. Planes are parametrized using a normal $n \in \mathrm{S}^2$[1] and distance to the origin $d \in \mathbb{R}$.

Nevertheless, we are taking a fixed-lag approach to the optimization problem. Therefore, we limit the estimation problem to the sets of variables that depend on the keyframes in a temporal window of size Δ_t. Hence, we limit the set $\mathcal{K}_t$ to all keyframes from time $t-\Delta_t$ to t. Conversely, Λ_t and Π_t will contain the sets of landmarks and planes visible from the set of keyframes $\mathcal{K}_t$. To avoid cluttering the notation, we skip the dependence of the sets $\mathcal{K}_t$, Λ_t and Π_t on the parameter Δ_t, but we nevertheless keep the time dependence explicit by the index t: $\mathcal{K}_t$, Λ_t and Π_t.

By reducing the number of state variables to a given window of time Δ_t, we will make the optimization problem tractable and solvable in real-time.

3.1.2 Measurements

The input for our system consists on measurements from the camera and the IMU.

We denote with $\mathcal{C}_i$ the image measurements at keyframe i. The camera can observe multiple landmarks l, hence $\mathcal{C}_i$ contains multiple image measurements $\mathbf{z}_i^l$, where we distinguish the landmarks that we will use for further structural regularities $\mathbf{z}_i^{l_c}$ (where the index c in l_c stands for constrained landmark), and the landmarks that will remain as structureless $\mathbf{z}_i^{l_s}$ (where the s in the index of l_s stands for structureless).

We denote with $\mathcal{I}_{ij}$ the set of IMU measurements acquired between two consecutive keyframes i and j. Therefore, we define the set of measurements collected up to time t by $\mathcal{Z}_t$:

$$\mathcal{Z}_t \doteq \{\mathcal{C}_i, \mathcal{I}_{ij}\}_{(i,j)\in\mathcal{K}_t}. \tag{3.5}$$

3.1.3 Maximum a Posteriori Estimation

We want to estimate our state $\mathcal{X}_t$, as defined in eq. (3.4), using a set of measurements $\mathcal{Z}_t$, defined in eq. (3.5). Recovering the actual values of $\mathcal{X}_t$ is difficult due to noisy measurements, and we must content ourselves with recovering the probability of having $\mathcal{X}_t$ given the measurements $\mathcal{Z}_t$. Therefore, we want to estimate the posterior probability of the state, given the visual and inertial measurements:

$$p(\mathcal{X}_t|\mathcal{Z}_t). \tag{3.6}$$

[1] $\mathrm{S}^2 \doteq \{(n_x, n_y, n_z)^T \quad | \quad \|\mathbf{n}\| = 1\}$

Applying the Bayes rule, we obtain:

$$p(\mathcal{X}_t|\mathcal{Z}_t) = \frac{p(\mathcal{Z}_t|\mathcal{X}_t)p(\mathcal{X}_t)}{p(\mathcal{Z}_t)} \propto p(\mathcal{Z}_t|\mathcal{X}_t)p(\mathcal{X}_0), \tag{3.7}$$

where we make use of the prior information on the initial state $p(\mathcal{X}_0)$ and we absorb the normalization factor over the measurements because it will not influence the result.

We use an independence assumption among the measurements to factorize eq. (3.7) into:

$$\begin{aligned} p(\mathcal{Z}_t|\mathcal{X}_t)p(\mathcal{X}_0) &= p(\mathcal{X}_0) \prod_{(i,j)\in\mathcal{K}_t} p(\mathcal{C}_i, \mathcal{I}_{ij}|\mathcal{X}_t) \\ &= p(\mathbf{x}_0) \prod_{(i,j)\in\mathcal{K}_t} p(\mathcal{I}_{ij}|\mathbf{x}_i, \mathbf{x}_j) \prod_{i\in\mathcal{K}_t} \prod_{l_c\in\mathcal{C}_i} p(\mathbf{z}_i^{l_c}|\mathbf{x}_i, \boldsymbol{\rho}_{l_c}) \prod_{l_s\in\mathcal{C}_i} p(\mathbf{z}_i^{l_s}|\mathbf{x}_i) \end{aligned} \tag{3.8}$$

With slight abuse of notation, we write $l_s \in C_i$ or $l_c \in C_i$ when a landmark l_s or l_c, respectively, is seen at time i by the camera.

In practice, we are not actually interested in the full probability density over the states $\mathcal{X}_t$, but rather on a single estimate of what $\mathcal{X}_t$'s most likely value is. To this end, we can try to find the $\mathcal{X}_t$ that maximizes the posterior density $p(\mathcal{X}_t|\mathcal{Z}_t)$.

The estimator that maximes $p(\mathcal{X}_t|\mathcal{Z}_t)$ is named *maximum a posteriori* (MAP), and we will refer to it as $\mathcal{X}_t^{MAP}$:

$$\mathcal{X}_t^{MAP} = \arg\max_{\mathcal{X}_t} p(\mathcal{X}_t|\mathcal{Z}_t) \tag{3.9}$$

Using the bayes rule in eq. (3.7), we have:

$$\mathcal{X}_t^{MAP} = \arg\max_{\mathcal{X}_t} p(\mathcal{Z}_t|\mathcal{X}_t)p(\mathcal{X}_0) \tag{3.10}$$

Maximizing eq. (3.10) is nevertheless not as convenient as minimizing the negative logarithm of the posterior probability, which simplifies to eq. (3.11) for zero-mean Gaussian noise:

$$\begin{aligned} \mathcal{X}_t^{MAP} &= \arg\min_{\mathcal{X}_t} -\ln\big(p(\mathcal{Z}_t|\mathcal{X}_t)p(\mathcal{X}_0)\big) \\ &= \arg\min_{\mathcal{X}_t} \|\mathbf{r}_0\|^2_{\Sigma_0} + \sum_{(i,j)\in\mathcal{K}_t} \left\|\mathbf{r}_{\mathcal{I}_{ij}}\right\|^2_{\Sigma_{ij}} + \sum_{i\in\mathcal{K}_t} \Big\{ \sum_{l_c\in\mathcal{C}_i} \left\|\mathbf{r}_{\mathcal{C}_i^{l_c}}\right\|^2_{\Sigma_\mathcal{C}} + \sum_{l_s\in\mathcal{C}_i} \left\|\mathbf{r}_{\mathcal{C}_i^{l_s}}\right\|^2_{\Sigma_\mathcal{C}} \Big\} \end{aligned} \tag{3.11}$$

where $\mathbf{r}$ represent the residual errors associated to the measurements, and $\boldsymbol{\Sigma}$ the covariance matrices.

We present now the optimization problem in eq. (3.11) using the expressiveness of probabilisitic graphical models.

3.1.4 Factor Graphs

A factor graph is a bipartite graph $F = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ with two types of nodes: factors $\phi_i \in \mathcal{U}$ and variables $x_j \in \mathcal{V}$. Edges $e_{ij} \in \mathcal{E}$ are always between factor nodes and variables nodes. The set of variable nodes adjacent to a factor ϕ_i is written as $\mathcal{N}(\phi_i)$, and we write X_i for an assignement

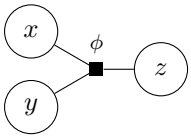
to this set. With these definitions, a factor graph F defines the factorization of a global function $\phi(X)$ as

$$\phi(X) = \prod_i \phi_i(X_i). \tag{3.12}$$

where dependence relationships are encoded by the edges e_{ij} of the factor graph, with each factor ϕ_i a function of the variables X_i in its adjacency set $\mathcal{N}(\phi_i)$.

We will use the notation detailed in table 3.1, and proposed in [102], to present the factor graph.

Table 3.1: Edges and Node types

Type	Symbol
Variable x_j	
Factor ϕ_i	
Factor graph edges	

Classical VIO factor graph

As an example of a common factor graph, we present in fig. 3.1 the one used in typical VIO pipelines, where the posterior is defined as:

$$\begin{aligned} p(\mathcal{Z}_t|\mathcal{X}_t)p(\mathcal{X}_0) &= p(\mathcal{X}_0) \prod_{(i,j)\in\mathcal{K}_t} p(\mathcal{I}_{ij}|\mathbf{x}_i, \mathbf{x}_j) \prod_{i\in\mathcal{K}_t} \prod_{l_c\in\mathcal{C}_i} p(\mathbf{z}_i^{l_c}|\mathbf{x}_i, \boldsymbol{\rho}_{l_c}) \\ &= \phi_0(\mathbf{x}_0) \prod_{(i,j)\in\mathcal{K}_t} \phi_{IMU}(\mathbf{x}_i, \mathbf{x}_j) \prod_{i\in\mathcal{K}_t} \prod_{l\in\mathcal{C}_i} \phi_l(\mathbf{x}_i, \boldsymbol{\rho}_{l_c}) \end{aligned} \tag{3.13}$$

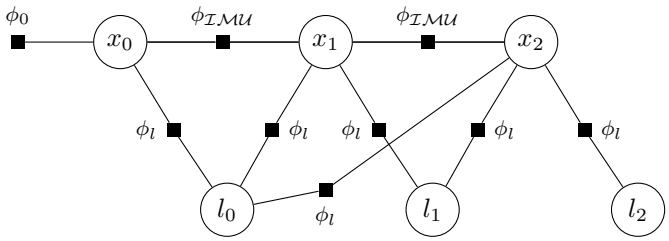


Figure 3.1: Typical VIO factor graph.

The factor graph in fig. 3.1 has the following factors:

- Pre-integrated IMU factors ϕ_{IMU}: we use the on-manifold pre-integration theory of Forster et al. [2] for the IMU factors.
- Projection factors ϕ_l: we use the standard reprojection error, here for a single image measurement $\mathbf{z}_i^{l_c}$ the residual is:

$$\mathbf{r}_{\mathcal{C}_i^{l_c}} = \mathbf{z}_i^{l_c} - \pi(\mathtt{R}_i, \mathbf{p}_i, \boldsymbol{\rho}_l), \tag{3.14}$$

where $\boldsymbol{\rho}_l \in \mathbb{R}^3$ denotes the position of the l-th landmark, and $\pi(\cdot)$ is a standard perspective projection.

Structureless VIO factor graph

The factor graph of a structureless VIO does not include landmarks as variables (fig. 3.2), as it uses instead a residual function $\mathbf{r}_{\mathcal{C}_i^{l_s}}$ that does not depend on the landmarks' 3D positions. We refer to [2] for the mathematical mecanisms to re-formulate the $\mathbf{r}_{\mathcal{C}_i^{l_c}}$ such that it does not depend on the landmark's position. Note that in fig. 3.2, we omit the prior on x_0 to simplify the factor graphs from now on.

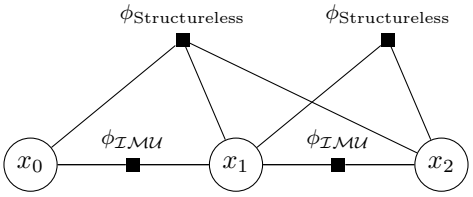


Figure 3.2: Structureless VIO factor graph.

Combined VIO factor graph

Our new formulation combines both projection and structureless factor graphs into the graph shown in fig. 3.3. This approach will allow us to add structural regularities between the landmark variables, while enabling fast inference times by conserving structureless factors for landmarks which are not subject to structural regularities.

We will denote the VIO solving the factor graph in fig. 3.3 as a Structureless and Projection (S + P) pipeline for the remaining of the thesis.

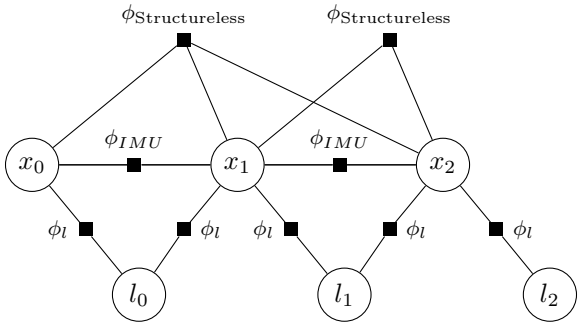


Figure 3.3: VIO factor graph combining both Structureless and Projection factors (S + P).

VIO factor graph with structural regularities

One of the structural regularities we want to detect and encode is co-planarity between landmarks. To do so, we first extract potential planes from the scene and we encode them in the opimization problem as a new set of variables $\{\boldsymbol{\pi}_\pi\}_{\pi \in \Pi_t}$, using the same notation as in § 3.1.1. Then, we can add regularity constraints between landmarks and planes, denoted by $\phi_{\mathcal{R}_1}$, to enforce co-planarity between the landmarks.

Figure 3.4 shows the regularity factors $\phi_{\mathcal{R}_1}$ between landmarks and plane estimates.

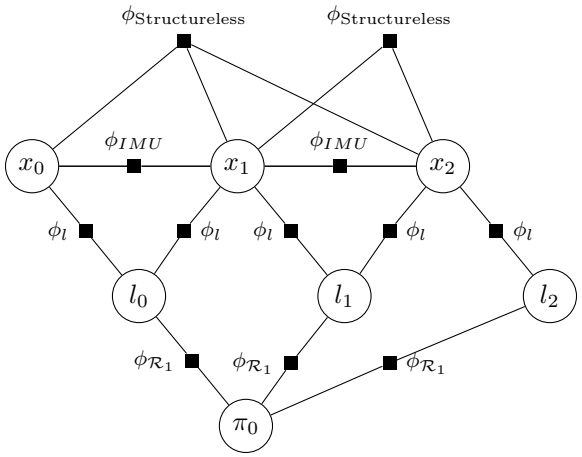


Figure 3.4: VIO factor graph combining Structureless, Projection and Regularity factors (S + P + R). The factor $\phi_{\mathcal{R}_1}$ encodes relative constraints between a landmark l_i and a plane π_0.

The factor graph in fig. 3.4 has the following types of factors:

- Pre-integrated IMU factors: as detailed in [2].
- Structureless factors: as detailed in [2].
- Projection factors ϕ_l: classical observation model for landmarks eq. (3.14).
- Regularity constraints $\phi_{\mathcal{R}_1}$: to be defined in § 3.1.5 below.

A regularity factor $\phi_{\mathcal{R}_1}$ embeds a probability distribution that depends on a landmark and a plane. There is no measurement involved. Instead, the regularity factors $\phi_{\mathcal{R}_1}$ encode relative constraints between landmarks and planes in the optimization problem.

Above we are not taking into account other constraints such as parallelism/orthogonality constraints between planes, which are typical regularities in an indoor environment. Hence, another interesting formulation is the one depticted in fig. 3.5. Where we are enforcing constraints between planes via the factor $\phi_{\mathcal{R}_2}$. In this case, there is also no measurement involved, and we are simply enforcing relative constraints between planes.

The maximum a posteriori corresponding to the factor graph in fig. 3.5 results in:

$$
\begin{aligned}
\mathcal{X}_t^{MAP} = \arg\min_{\mathcal{X}_t} & \left\|\mathbf{r}_0\right\|_{\Sigma_0}^2 \\
& + \sum_{(i,j)\in\mathcal{K}_t} \left\|\mathbf{r}_{\mathcal{I}_{ij}}\right\|_{\Sigma_{ij}}^2 \\
& + \sum_{i\in\mathcal{K}_t} \left\{ \sum_{l\in\mathcal{C}_i^c} \left\|\mathbf{r}_{\mathcal{C}_{il}^c}\right\|_{\Sigma_\mathcal{C}}^2 + \sum_{l\in\mathcal{C}_i^s} \left\|\mathbf{r}_{\mathcal{C}_{il}^s}\right\|_{\Sigma_\mathcal{C}}^2 \right\} \\
& + \sum_{l_c\in\Lambda_t} \sum_{\pi\in\Pi_t} \delta(l_c,\pi) \left\|\mathbf{r}_{\mathcal{R}_1}^{l_c}\right\|_{\Sigma_{\mathcal{R}_1}}^2 \\
& + \sum_{(\pi_1,\pi_2)\in\Pi_t} \delta(\pi_1,\pi_2) \left\|\mathbf{r}_{\mathcal{R}_2}\right\|_{\Sigma_{\mathcal{R}_2}}^2 ,
\end{aligned}
\tag{3.15}
$$

where $\delta(l_c,\pi)$ is the data association term for landmark to plane association, and $\delta(\pi_1,\pi_2)$ is the term for plane to plane association. Both functions return a value of 1 if the given arguments are

linked in the factor graph (returns 1 only once to avoid duplicate factors), 0 otherwise. We explain in § 3.2.3 how the data association is actually done.

In the following § 3.1.5, we specify the exact formulation for the regularity factors $\phi_{\mathcal{R}_1}$ and $\phi_{\mathcal{R}_2}$.

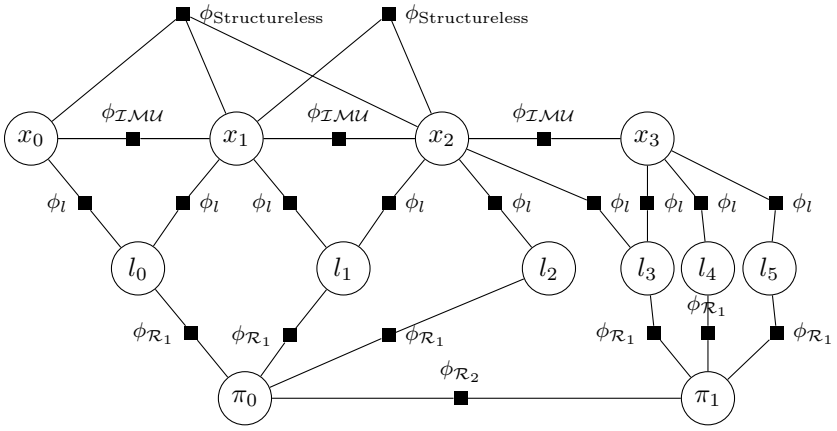


Figure 3.5: VIO factor graph combining Structureless, Projection and Regularity factors (S + P + R). In this graph we add the factor $\phi_{\mathcal{R}_2}$ to enforce relative constraints between planes.

Notice that only landmarks which could be effectively associated to a planar surface are used for regularization, the ones that could not be associated with a plane are kept as structureless factors. In practice, this is not exactly the case. Since we have to ensure that when adding a plane variable it remains fully constrained, we have to be especially careful with degenerate configurations. For example, given three landmarks on a plane, if these landmarks are aligned in a line, then a plane defined by these landmarks would be underconstrained, leading to an ill-posed linear system. To avoid this, we proceed in a rather naive but practical way by requiring that at least a given number of landmarks are associated to a plane before adding the plane in the optimization problem. This nevertheless creates the scenario where there might be some landmarks in the optimization problem that are constrained by projection factors, yet they are not constrained by any regularity factor.

3.1.5 Regularity constraints

$\phi_{\mathcal{R}_1}$: Constraints between landmarks and planes

Let us take a toy example for the planarity constraints. Point-clouds of triangulated landmarks in a typical VIO pipeline are usually scattered around the real surfaces they represent. Take a textured wall for example, a VIO pipeline will be able to triangulate the 3D points of this wall, but these 3D map points will not lie on a plane due to noise in the measurements. These landmarks will be distributed (ideally in a Gaussian way) around the plane primitive as in fig. 3.6.

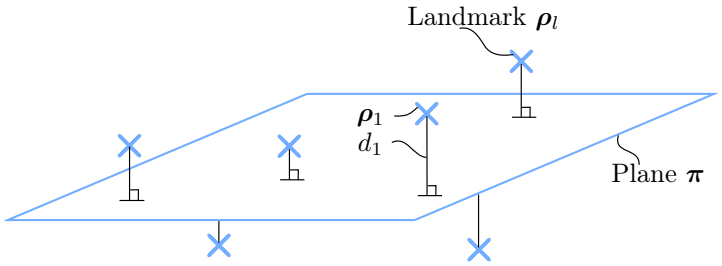


Figure 3.6: A possible distribution of 3D landmarks $\boldsymbol{\rho}_j$ obtained from triangulation. The distance from $\boldsymbol{\rho}_1$ to the plane $\boldsymbol{\pi}$ is represented by d_1.

Assuming we have been given the association of points to planes, we can use different metrics to enforce planarity constraints for these landmarks.

1. A simple planarity constraint would consist in taking the sum of the distances from each landmark $\boldsymbol{\rho}_l$ to its corresponding plane $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)$. The cost function for a single landmark $\boldsymbol{\rho}_l$ would then be:

$$d_\perp(\boldsymbol{\pi}, \boldsymbol{\rho}_l) = \frac{|\pi_1 x_j + \pi_2 y_j + \pi_3 z_j + \pi_4|}{\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2}} \tag{3.16}$$

The optimal landmarks $\boldsymbol{\rho}_l^*$ would then minimize the following optimization problem:

$$\boldsymbol{\rho}^* = \underset{\boldsymbol{\rho}}{\operatorname{argmin}} \sum_{l=0}^{N} d_\perp(\boldsymbol{\pi}, \boldsymbol{\rho}_l) \tag{3.17}$$

and in case we want to optimize over the plane parameters as well:

$$\boldsymbol{\pi}^*, \boldsymbol{\rho}^* = \underset{\boldsymbol{\pi}, \boldsymbol{\rho}}{\operatorname{argmin}} \sum_{l=0}^{N} d_\perp(\boldsymbol{\pi}, \boldsymbol{\rho}_l), \quad N \geqslant 2 \tag{3.18}$$

In this case, we need at least 3 points to avoid an infinite number of solutions, therefore $N \geqslant 2$. We have also to be careful to avoid degenerate configurations such as the one where all points lie on a single line, which would allow for an infinite number of solutions.

If we parametrize the plane $\boldsymbol{\pi}$ as having a normal $\boldsymbol{n}$ and a distance to the origin d. The normal of the plane being $\boldsymbol{n} = \frac{(\pi_1, \pi_2, \pi_3)}{\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2}}$ and the distance $d = \frac{-\pi_4}{\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2}}$. We have the following optimization problem:

$$\boldsymbol{n}^*, d^*, \boldsymbol{\rho}^* = \underset{\boldsymbol{n}, d, \boldsymbol{\rho}}{\operatorname{argmin}} \sum_{l=0}^{N} d_\perp(\boldsymbol{n}, d, \boldsymbol{\rho}_l), \quad s.t. \ \|\boldsymbol{n}\|^2 = 1, \quad N \geqslant 2 \tag{3.19}$$

where we re-define $d_\perp(\boldsymbol{n}, d, \boldsymbol{\rho}_l)$ as:

$$d_\perp(\boldsymbol{n}, d, \boldsymbol{\rho}_l) = \left\| \boldsymbol{n}^\mathsf{T} \cdot \boldsymbol{\rho}_l - d \right\|^2 \tag{3.20}$$

Also, we need to make sure the optimization leads to a set of plane parameters which indeed represent a plane. In other words, the normal of the plane should be of unit norm $\|\boldsymbol{n}\|^2 = 1$.

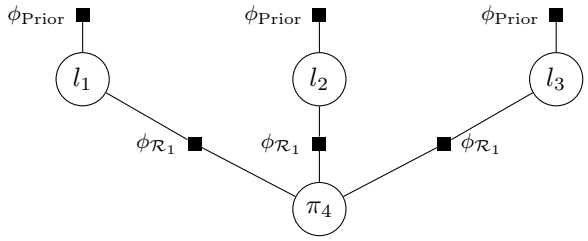


Figure 3.7: Minimalistic factor graph for the cost function of Eq. 3.19: three landmarks l_i, with given prior factors ϕ_{Prior}, and a plane π_4.

Example: A minimalistic factor graph representation of the cost function 3.19 would involve three landmarks and a plane, as shown in fig. 3.7. Notice that prior factors ϕ_{Prior} are added to the landmarks to avoid having under-constrained variables.

To implement the factors $\phi_{\mathcal{R}_1}$ in GTSAM [103], we must first define what is a plane. For this, we have used a previous implementation from [92], where they define an OrientedPlane3 by using a normal/distance parametrization of a plane. Where the normal is an element of the ordinary unit 2-sphere $S^2 = \left\{x \in \mathbb{R}^3 : \|x\| = 1\right\}$[2].

Once we have defined what a plane is, we need to create a factor class which will take as parameters the variables involved, in our case a landmark and a plane, and use them to compute the error, as well as the jacobians of the error with respect to the variables. In this case, the error between the 3D landmark $\boldsymbol{\rho}$ and the plane $\boldsymbol{\pi} = (\boldsymbol{n}, d)$ corresponds to $\boldsymbol{n}^\mathsf{T} \cdot \boldsymbol{\rho} - d$, and we are left finding the Jacobian of the error. Unfortunately, since the normal lives on the two-dimensional manifold S^2, we cannot directly optimize over it. Instead, we need a way to know how to move in the neighborhood of our initial estimate of the normal. Using a vector addition + to update a normal, as if it was a simple vector, would likely result in an element which is not a normal as it will not lie in the S^2 manifold. In addition, we know that normal vectors have only two degrees of freedom, not three, because of the unit norm constraint.

For a normal vector in the unit sphere, a good candidate for two-dimensional, vector-valued increments can be obtained from the tangent space $T_{\boldsymbol{n}}S^2$; defined as the plane orthogonal to the normal vector $\boldsymbol{n}$ on the sphere S^2. Mathematically, we can define $T_{\boldsymbol{n}}S^2$ as all vectors tangent to S^2 at $\boldsymbol{n}$:

$$T_{\boldsymbol{n}_0}S^2 \doteq \left\{\hat{\xi} \in \mathbb{R}^3 | \boldsymbol{n}^T \hat{\xi} = 0\right\} \tag{3.21}$$

If we parametrize the error using the retraction $\mathcal{R}_{\boldsymbol{n}}(\boldsymbol{v}) : T_{\boldsymbol{n}_0}S^2 \to \mathrm{S}^2$ from the tangent space $T_{\boldsymbol{n}}S^2 \in \mathbb{R}^2$ to S^2 at the linearization point $\boldsymbol{n}$, we arrive to an error formulation as follows:

$$e(\boldsymbol{n}, d) = \boldsymbol{n}^\mathsf{T} \cdot \boldsymbol{\rho} - d \Leftrightarrow e(\boldsymbol{v}, d) = \mathcal{R}_{\boldsymbol{n}}(\boldsymbol{v})^\mathsf{T} \cdot \boldsymbol{\rho} - d \tag{3.22}$$

Now, we can define how to move around the normal $\boldsymbol{n}$ by taking the jacobian of $e(\boldsymbol{v}, d)$ with respect to $\boldsymbol{v} \in \mathbb{R}^2$ and $d \in \mathbb{R}$.

By implementing the factor graph defined in fig. 3.7 and optimizing using Gauss-Newton, we obtain the following results in GTSAM, where we called `BasicRegularPlane3Factor` the regularity factor $\phi_{\mathcal{R}_1}$:

```
Factor Graph:
size: 6

Factor 0: PriorFactor on 1
```

[2]Named Unit3 in GTSAM.

```
prior mean: [0, 0, 1]'
isotropic dim=3 sigma=0.1
Factor 1: PriorFactor on 2
prior mean: [1, 0, 1]'
isotropic dim=3 sigma=0.1
Factor 2: PriorFactor on 3
prior mean: [0, 1, 1]'
isotropic dim=3 sigma=0.1

BasicRegularPlane3Factor Factor on point 1, 4
isotropic dim=1 sigma=0.5
BasicRegularPlane3Factor Factor on point 2, 4
isotropic dim=1 sigma=0.5
BasicRegularPlane3Factor Factor on point 3, 4
isotropic dim=1 sigma=0.5

Initial Estimate:
Values with 4 values:
Value 1: (N5gtsam6Point3E) [0, 19, 3]'
Value 2: (N5gtsam6Point3E) [-1, 2, 2]'
Value 3: (N5gtsam6Point3E) [0.3, -1, 8]'
Value 4: (N5gtsam14OrientedPlane3E)  : 0.107833
                                       0.215666
                                       0.970495
                                       0

Initial error: 21577.9
newError: 0.208035
newError: 0.00477048502535
newError: 4.07647603461e-10
newError: 5.42341872339e-31
newError: 0

Final Result:
Value 1: (N5gtsam6Point3E) [0, 0, 1]'
Value 2: (N5gtsam6Point3E) [1, 0, 1]'
Value 3: (N5gtsam6Point3E) [0, 1, 1]'
Value 4: (N5gtsam14OrientedPlane3E)  :  2.5972326623e-17
                                        1.38818376841e-17
                                        1
                                        1
```

In the above example, the landmarks are anchored by the priors such that they lie on a plane with a normal pointing in the z direction and at a height of 1 meter. The plane instead is not directly constrained by a prior but by the three regularity factors $\phi_{\mathcal{R}_1}$.

Before the optimization, we set as initial estimates for the landmarks to be scattered in the world. Nevertheless, the plane's initial estimate is set to be relatively close to the optimial value. The actual initial values can be seen in `Initial Estimate`. For the `OrientedPlane3`, the first three numbers represent the normal's parameters, while the last one is the distance.

After the optimization, all landmarks are situated right where the priors suggest that the landmarks should be, while the plane updates its parameters to pass through the landmarks, up to numerical errors, as can be seen in `Final Result`.

2. Alternatively, co-planarity constraints can be defined in image space, which is especially

suitable to avoid the scale ambiguity in monocular visual SLAM [99]. Since in our case the scale is observable, we did not need to proceed this way. Nevertheless, it would be interesting to explore what are the effects of using one against the other.

$\phi_{\mathcal{R}_2}$: Constraints between planes

Another set of regularities are the ones that we can define between estimated planes themselves. If we parametrize the planes by their normals and distances to the origin, it is straightforward to define these error metrics.

Below we detail the possible metrics to be used. Unfortunately, for this thesis, we will not enforce these regularities in the factor graph as they require a powerful frontend capable of extracting plane to plane constraints while ensuring that no false positives are added.

1. Parallelism: minimizing the difference between the normal vectors $\boldsymbol{n}$ of both plane estimates $\hat{\boldsymbol{\pi}}_i$ and $\hat{\boldsymbol{\pi}}_j$ should encourage the two planes to be parallel. Hence, the factor ϕ_{R_2} could encode the following residual (where the word residual is misleading because there is no proper measurement):

$$\mathbf{r}_{/\!/} = \left\| \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_i} - \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_j} \right\|^2_{\Sigma_I} \tag{3.23}$$

   where i and j correspond to the indices of the planes which we believe are parallel, the covariance matrix being isotropic Σ_I.

   In order to find the Jacobian of Eq.3.23, we need to define the same error metric using instead the retraction from tangent space to the unit 2-sphere, $\mathcal{R}_{\boldsymbol{n}_i}(\boldsymbol{v}) : T_{\boldsymbol{n}_i} S \to \mathrm{S}^2$:

$$\mathbf{r}_{/\!/}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \left\| \mathcal{R}_{\boldsymbol{n}_1}(\boldsymbol{v}_1) - \mathcal{R}_{\boldsymbol{n}_2}(\boldsymbol{v}_2) \right\|^2_{\Sigma_I} \tag{3.24}$$

   Now, we can calculate how to move in the tangent space in order to reduce the error metric while keeping the constraints on the norms of the normals.

   One might ask at this point, why we do not define the error metric in tangent space instead of doing it in $\mathbb{R}^3$. Where the error would be defined by ξ, the $\mathbb{R}^2$ vector in tangent space.

   Tangent space metric corresponds to the geodesic distance (technically, it is called orthodromic distance, which is the shortest distance between two points on the surface of a sphere), while the one in $\mathbb{R}^3$ corresponds to the chordal distance.

   The geodesic metric corresponds to the projection of one normal to the tangent space spanned by the other normal. The error is therefore defined in $\mathbb{R}^2$, instead of $\mathbb{R}^3$ as in eq. (3.19). Previous work on geodesic and chordal metrics for rotations has been done, and showed that both formulations are tightly related [104].

2. Parallelism + separation: A more general and expressive formulation of parallelism would be to allow for setting distances between planes, the cost function of eq. (3.23) is then augmented to:

$$\mathbf{r}_{/\!/\Delta} = \left\| \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_i} - \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_j} \right\|^2_{\Sigma_I} + \left\| d_{\hat{\boldsymbol{\pi}}_i} - d_{\hat{\boldsymbol{\pi}}_j} - \Delta_{ij} \right\|^2_{\Sigma_I} \tag{3.25}$$

   where Δ_{ij} corresponds to the measured distance between planes i and j in Υ. Co-planarity then becomes a particular case where $\Delta_{ij} = 0$.

   Nevertheless, in our case, we will potentially detect sets of landmarks which should be co-planar, and then initialize a plane estimate which links to these landmarks using regularity constraints of type R_1. Once we do this for two or more sets of landmarks, we can apply R_2 regularity constraints.

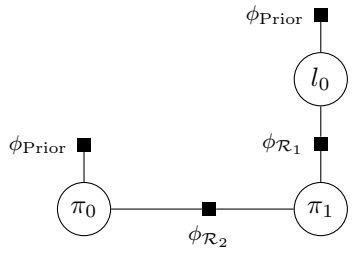


Figure 3.8: Minimalistic factor graph for the cost function of Eq. 3.23: a plane, with prior factor ϕ_{Prior}, and another plane constrained to a landmark $\phi_{\mathcal{R}_1}$ (with a prior itself ϕ_{Prior}) and the first plane $\phi_{\mathcal{R}_2}$.

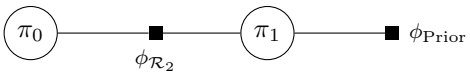


Figure 3.9: Minimalistic factor graph for the cost function of Eq. 3.25: a plane, with a prior factor ϕ_{Prior} constrained to another plane.

3. Orthogonality: for orthogonal planes we would have:

$$\mathbf{r}_{\perp} = \left\| \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_i}^{\mathsf{T}} \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_j} \right\|_{\Sigma_I}^2 \tag{3.26}$$

   Unfortunately, this allows for multiple solutions. Fixing one of the normals, all the vectors generated by rotating a vector perpendicular to this normal would minimize the cost, therefore allowing for multiple planes to solve the problem.

4. Orthogonality + separation: a single solution appears if we measure as well the difference between distances to the origin of each plane (Δ_{ij}).

$$\mathbf{r}_{\perp\Delta} = \left\| \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_i}^{\mathsf{T}} \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_j} \right\|_{\Sigma_I}^2 + \left\| d_{\hat{\boldsymbol{\pi}}_i} - d_{\hat{\boldsymbol{\pi}}_j} - \Delta_{ij} \right\|_{\Sigma_I}^2 \tag{3.27}$$

   There are also degenerate configurations, for example if the planes pass through the origin, then the distances are always zero, and we have many solutions again.

5. Another extra generalization would be to allow any angle between planes, where the dot product between normals would be used to measure the difference against the cosine function of the angle. Nevertheless, it is difficult to measure the angle between planes besides orthogonality and parallelism, at least when using an RGB camera.

$$\mathbf{r}_{\theta} = \left\| \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_i}^{\mathsf{T}} \boldsymbol{n}_{\hat{\boldsymbol{\pi}}_j} - \cos(\theta) \right\|_{\Sigma_I}^2 \tag{3.28}$$

3.1.6 Robust cost functions

The optimization should take into account that certain observations might be outliers – such as when wrongly assigning a new observation to an earlier observation § 3.2.3 – and consequently can corrupt the estimated variables. This is especially problematic in least-squares optimization, which is particularly sensitive to outliers.

A robust cost function has the ability to reduce the influence of outliers. They achieve this property by reducing the cost (residual cost) associated to large errors, compared to a quadratically increasing function.

Therefore, instead of minimizing the square of the residuals $\sum_{i=1}^{N} = r_i^2$, we minimize another function of the residuals $\sum_{i=1}^{N} = \rho(r_i)$, where $\rho(\cdot)$ is ideally a robust function to outliers.

GTSAM [103] provides the option to use robust error functions in the factors. We have considered three cost functions for three types of factors. The factors that we consider are: point-plane regularity factors and projection factors. The cost functions that we tested are:

(i) L2, i.e. the standard least-squares error function defined as r^2.

(ii) Huber loss: defined as $\rho_\delta(r) = \begin{cases} \frac{1}{2}r^2 & \text{for } |r| \leq \delta, \\ \delta(|r| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$ [105]

(iii) Tukey loss: defines as $\rho_c(r) = \begin{cases} \frac{1}{6}c^2(1 - [1 - (r/c)^2]^3) & \text{for } |r| \leq c, \\ (c^2/6), & \text{otherwise.} \end{cases}$

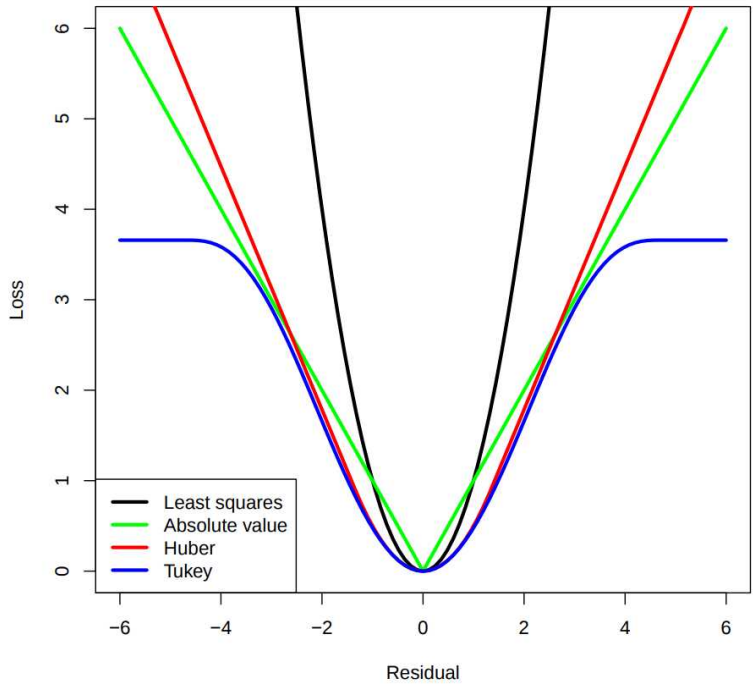


Figure 3.10: Curves for different cost functions. Source: web.as.uky.edu/statistics/pbreheny

Figure 3.10 shows the slope of these functions, and fig. 3.11 shows the actual effect of using the robust cost functions that we are considering on a simple linear regression problem where some of the data points are outliers. It can be seen that while the least-squares error function is largely influenced by the outlier data points, the Huber error function better fits the inlier data. On the extreme, the Tukey error function is even capable of completely ignoring outliers, due to its constant term for errors larger than its tuning constant c.

Decreasing the tuning constants δ and c make the error functions more robust to outliers, but the efficiency is lower when the errors are normally distributed. Where we define the efficiency in an informal way: in the case that there are no outliers in the data and it is corrupted by Gaussian noise, an efficient estimator is expected to have an approximately normal distribution (we refer the reader to [106] for details).

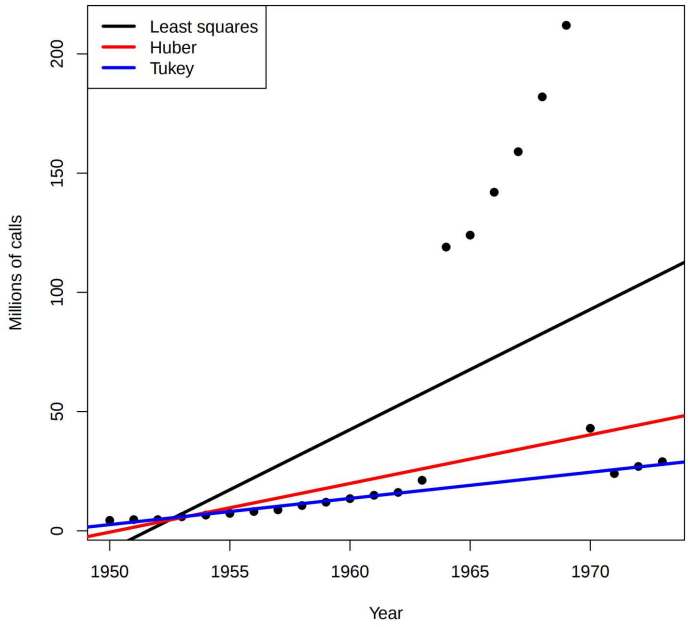


Figure 3.11: Linear regression over data with outliers when using different cost functions. Source: web.as.uky.edu/statistics/pbreheny

The tuning constant should be chosen to have a high efficiency, yet be robust to outliers. For the Huber norm the tuning constant is recommended to be $k = 1.345\sigma$ to have the highest efficiency, and for Huber $c = 4.6851\sigma$, where σ is the standard deviation of the errors.

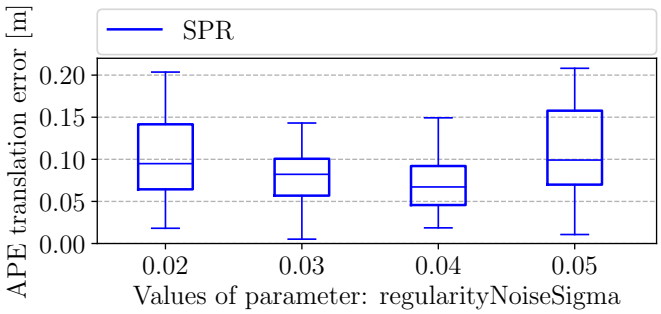
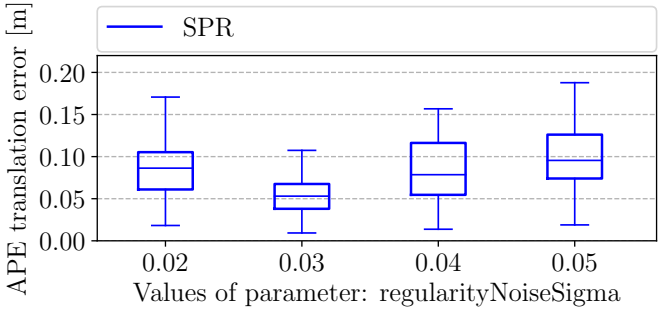
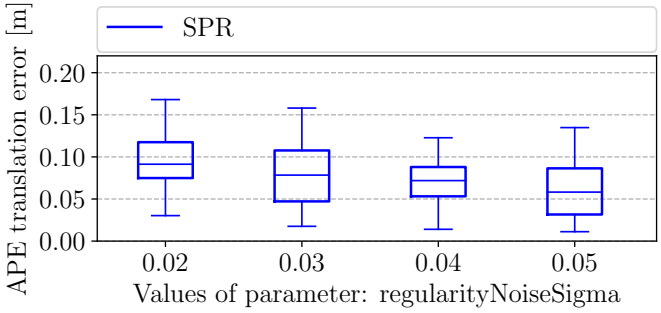
In practice, we need a good estimate of the standard deviation of the errors to find the right tuning constants. Nevertheless, it is recommended to use a robust measure of spread instead of the standard deviation of the errors. For example, using the median absolute residual (MAR) $\hat{\sigma} = \text{MAR}/0.6745$ [107], similarly to what the authors of PTAM [108] used.

In our case, we find the right tuning constant by running the VIO multiple times with different parameters for the standard deviation of the errors, the type of error function, and different values for the tuning constant (in the neighborhood of the median of the standard deviation of all the residuals). We proceed in this way for the two types of factors we consider: regularity and projection factors. Table 3.2 shows the results after several of these evaluations for a subset of tested values and just for the regularity factors. We use boxplots to observe the performance of the pipeline depending on the noise sigma (σ) for the given factor, the type of norm used, and the tuning constant (which is given proportional to the actual σ). The boxplots show the maximum, the minimum, the median and the first and third quartile of all Absolute Position Errors at each pose of the camera compared with the ground-truth, more details about this error can be found in § 4.2.2. We can see that using a Huber or Tukey robust error function might lead to better results than using an L-2 norm.

3.1.7 Optimization

The optimization problem formulated in eq. (3.15), corresponding to the factor graph in fig. 3.5, can be solved in an incremental and efficient manner by using iSAM2 [29]. iSAM2 makes use of a Bayes Tree to perform a fast incremental update of the square-root information matrix by

Table 3.2: Absolute Pose Error depending on Noise Model, Noise Model Parameter, and Noise Sigma for Point-Plane Regularity factors. The VIO also uses Structureless, Projection, and Regularity factors (S + P + R).

Norm Type	Norm Parameter [$\times\sigma$]	APE translational error w.r.t. Regularity Noise Sigma (σ)
L-2	No Parameter	
Huber	1.345	
Tukey	4.6851	

reusing previous computations. The Bayes tree representation allows relinearization of a small set of variables, therefore providing a boost in efficiency while keeping the sparse structure of the problem. Unfortunately, since we are adding multiple constraints between variables, we are contributing to making the optimization problem less sparse, which negatively affects the optimization convergence time, as we will se in § 4.4.

3.2 Frontend

Our frontend has the same components than a keyframe-based indirect (extracts sparse keypoints) visual inertial odometry pipeline, but it also incorporates a module to generate a 3D mesh, and another one to detect structural regularities from the 3D mesh. Therefore, our frontend consists of:

1. Keypoint detector: we use a Harris keypoint detector [109].
2. Feature tracker: we use an iterative Lucas-Kanade method with pyramids [110]. We also use RANSAC to discard inconsistent tracked keypoints [111].
3. Keyframe selector: which selects a subset of images for further processing (called keyframes) that are expected to provide better motion information than consecutive frames, similarly to [3].
4. Keyframe to IMU synchronization: visual and inertial information need to be synchronized to be fused accurately in the optimization problem.
5. 3D Mesh generator: uses the output from the keypoint detector to triangulate features and construct a 2D mesh. This mesh is then projected in 3D, and filtered to remove faces of the mesh that do not represent an actual surface.
6. Regularities detector: detects planes from the 3D mesh, and does data association between planes and landmarks, and between planes in the optimization problem and newly detected planes.

Except for the 3D Mesh generator and the regularities detector, we use a rather standard approach for the rest of the modules. The reader may refer to the references given above for more details.

3.2.1 3D Mesh generation

Building a consistent 3D Mesh of the environment using a sparse point cloud from a SLAM system is difficult because:

- The 3D positions of the landmarks are noisy, and some are outliers.
- The density of the point cloud is highly irregular.
- The point cloud is morphing by removing and adding points, and the points that remain are also changing positions at each iteration.

Therefore, we avoid performing a 3D tetrahedralisation directly from the sparse 3D landmarks, which would require expensive algorithms such as space carving, to extract meaningful

surfaces. Instead, we perform a triangularisation on the keypoints in the image, and project the 2D triangulation in 3D, resulting in a 3D mesh at a fraction of the computational cost of 3D tetrahedralisation.

This approach does not come without caveats, as we are not yet solving any of the issues raised above. To solve the first issue we will need to remove outliers in a post-processing step. For the second, there is not much we can do. And for the third, while there are works that present interesting approaches for incremental mesh reconstruction [59], these are meant for 3D Meshes, so we will need to process the incremental part from a 2D perspective. Some algorithms use the mesh in a per-frame basis instead of expanding the mesh incrementally [11, 51, 112]. Nevertheless, we want to enforce structural regularities on possibly all landmarks that are currently present in the optimization problem.

To achieve this we will need to propagate the 3D mesh both temporally and spatially. We define temporal propagation of a mesh as the process by which the mesh is updated when the positions of the 2D keypoints, and therefore the positions of 3D points are updated. Similarly, we define spatial propagation of a mesh as the way the mesh is updated when new keypoints and thereby landmark positions appear; or when keypoints are removed; or when landmarks are marginalized from the optimization problem. Note that the last two cases can occur independently: it can happen that a keypoint is no longer tracked in the current image, but its corresponding landmark is still present in the optimization problem; and conversely, it can happen that the landmark is marginalized out of the optimization problem, while the keypoint is still being tracked in the current frame. We define these concepts ourselves as we could not find similar terminology in previous literature.

Below we present the process to go from 2D points in the image to a 3D Mesh in space, and its temporal and spatial propagation.

Keypoint Detection and Tracking

Figure 3.12 shows the keypoints detected in the image as green squares, as well as the feature tracks as green lines. The tracks are shortened to the previous observation of the keypoint to avoid cluttering the image. Red crosses with yellow circles represent keypoints that were wrongly tracked and thereby classified as outliers by RANSAC. We then use the extracted keypoints for triangulation.

2D Delaunay Triangulation from Sparse Keypoints

Amongst all the keypoints that are successfully tracked, we discard keypoints that do not have a 3D landmark associated. This is because we use the 3D position of the landmarks to project the mesh in 3D, as explained in § 3.2.1. We also discard those keypoints that do not have a valid stereo correspondence. Figure 3.14 shows as blue dots the keypoints used for triangulation; all the discarded keypoints, either because they lack an associated 3D landmark or because they do not have a stereo correspondence, are represented by red dots.

Figure 3.13 shows an schematic representation of which keypoints are used for 2D Delaunay triangulation and which keypoints are discarded. In particular, only those keypoints which have a stereo correspondence and have an associated 3D landmark are considered valid.

Using the valid keypoints in the image, we compute a 2D Delaunay triangulation as shown in fig. 3.14. Mathematically, a Delaunay triangulation for a given set of points is a triangulation such that no point is inside the circumcircle of any triangle.

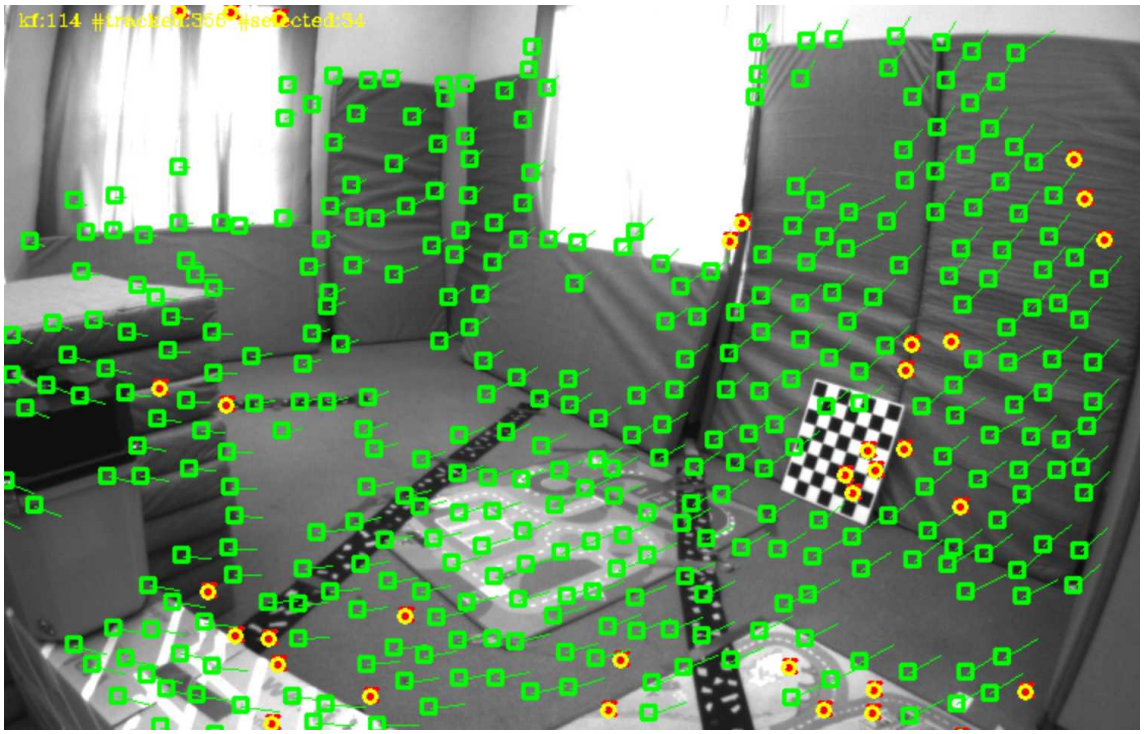


Figure 3.12: Keypoint detection and tracking on image. Green squares represent successfully tracked keypoints, with their previous location pointed at by a green line. Red crosses with yellow circles correspond to keypoints that were not tracked correctly (RANSAC's outliers).

The Delaunay triangulation maximizes the minimum angle of all the angles of the triangles in the triangulation; thereby avoiding triangles with one or two extremely acute angles (silver triangles). Since we want to promote triangles that represent planar surfaces, this is a desirable property, as it will result in near isotropic triangles that cover a good extent of a potentially planar surface. Silver triangles instead, are typically long and thin, so they do not provide much information about the planarity of the underlying surface.

However, the Delaunay triangulation does not necessarily minimize the length of the edges. Which will potentially lead to large triangles that cover an unreasonable portion of the image. We can nevertheless easily detect and discard these triangles, as explained in § 3.2.1. Similarly, while the Delaunay triangulation reduces the number of silver triangles, there will still be some that are still present, and that we will need to deal with.

2D to 3D Mesh Projection

With the newly obtained triangulation from keypoints having a corresponding landmark, it is possible to simply extrapolate the triangles obtained in 2D connecting triplets of keypoints to triangles in 3D connecting the corresponding triplets of landmarks, as shown in fig. 3.15.

Mesh Propagation

Concerning mesh propagation:

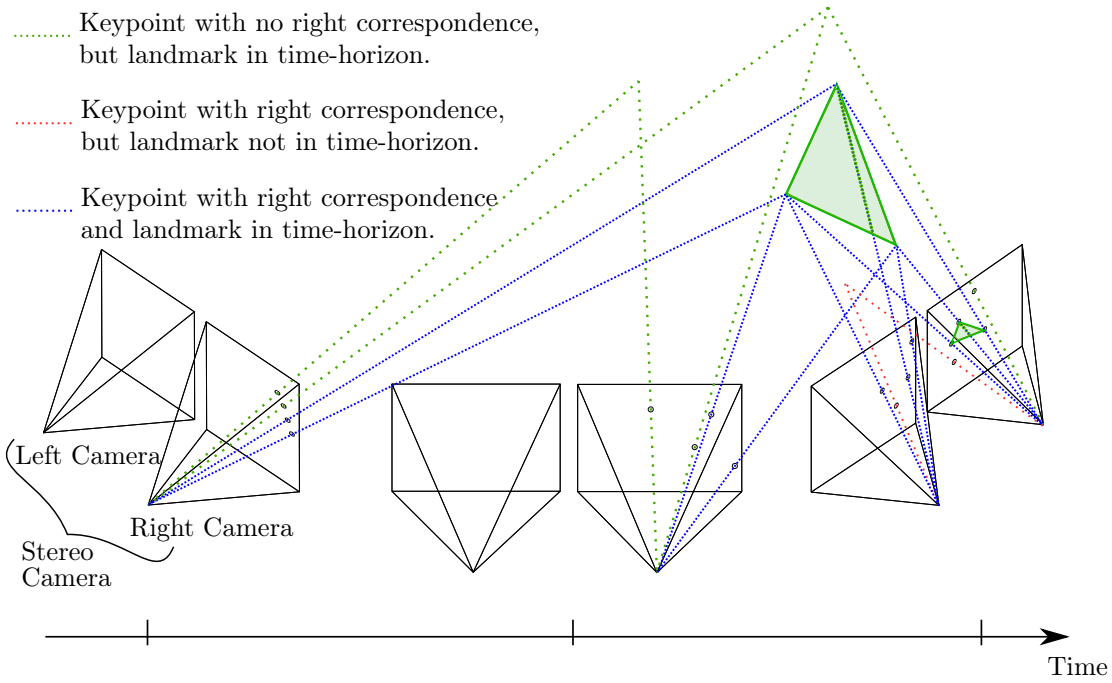


Figure 3.13: Landmark's typology in the optimization's time-horizon.

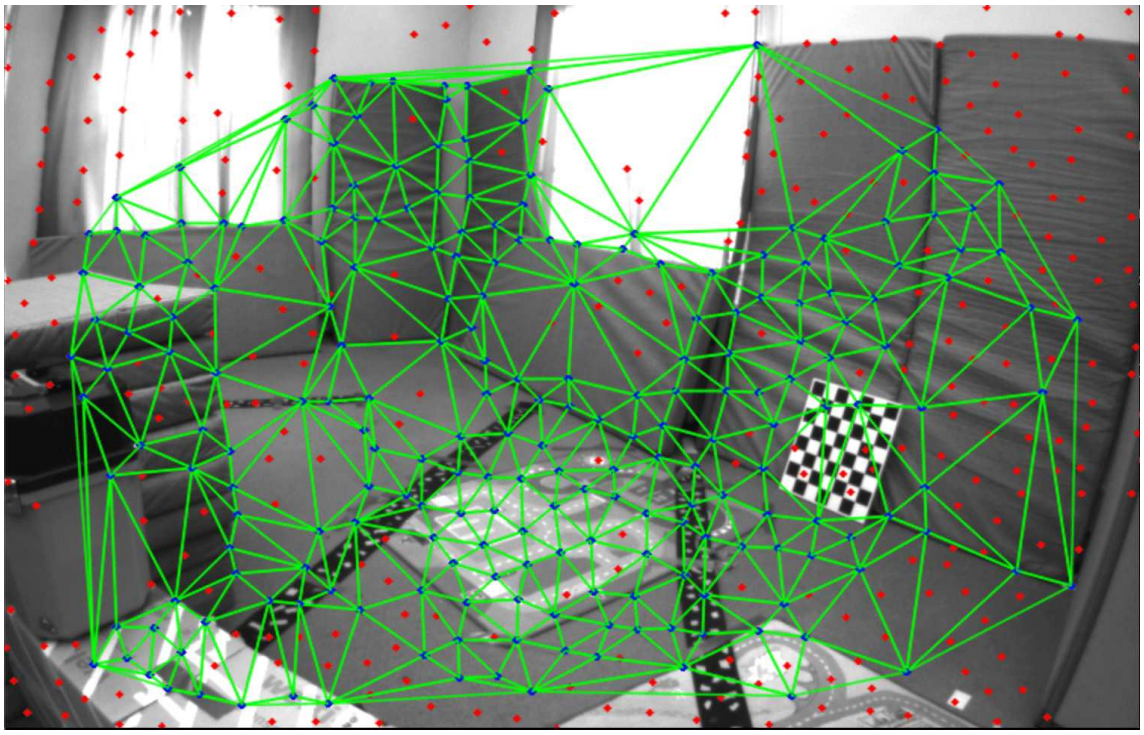


Figure 3.14: Delaunay triangulation on successfully tracked and triangulated keypoints, with valid stereo correspondence (blue dots). Red dots represent two types of keypoints: (i) keypoints with valid stereo correspondences that were not tracked in the previous keyframe. (ii) keypoints that were successfully tracked in the previous keyframe, but have no valid stereo correspondence.

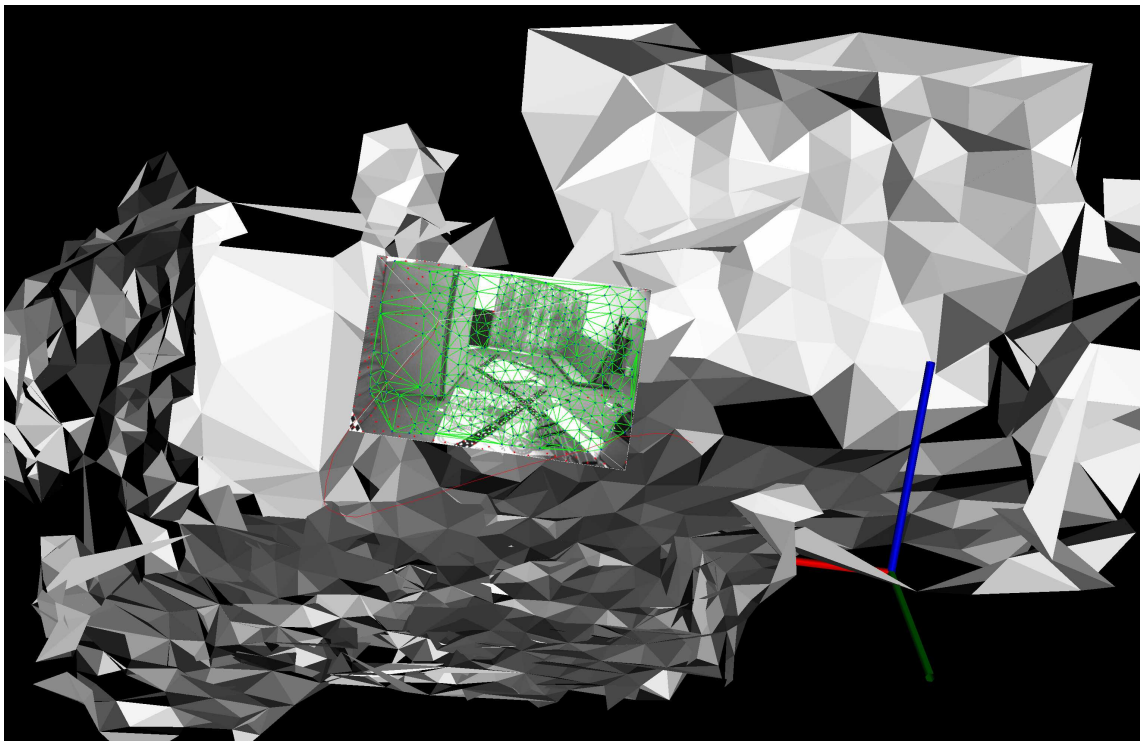


Figure 3.15: 2D to 3D Mesh projection. The rectangle in the center of the figure corresponds to the 2D Delaunay triangulation (in green) projected on the frustum of the camera; the 3D Mesh is represented by the grey triangles.

- *Spatial propagation*, or what to do when new keypoints appear and/or old ones disappear. Unfortunately, most of the keypoints' positions on the 2D image change each time the camera moves. Therefore, using an incremental 2D Delaunay triangulation with the expectation of reducing computational cost does not seem a viable approach, as we would incurr the cost of bookkeeping which keypoints have changed pixel positions, which have been added and those that have been removed. Hence, we will be computing a 2D Delaunay triangulation from scratch over the keypoints of the current frame.

  Alternatively, we could have kept the original triangulation between keypoints of the previous frame on the current frame, potentially breaking the triangulation's Delaunay property. We would thereafter remove the edges linking a keypoint which is not tracked anymore and perform a Delaunay triangulation for the new keypoints (and its neighbours). This would result in an hybrid triangulation, with some subset of triangles having the Delaunay property while the triangulation as a whole would not.

- *Temporal propagation*, or what to do to with the 3D mesh when a new frame – with an updated 2D triangulation – is available, and when old landmarks disappear from the optimization's time-horizon.

  - A new 2D triangulation is available: in this case, since we are keeping track of which keypoint is associated to each landmark, we can convert all the triangles of the 2D triangulation to 3D mesh faces, while avoiding to duplicate faces.

    More specifically, the 2D triangulation is stored as a list of triplets of keypoints: a $n \times 3$ matrix, where each row contains the ids of the vertices (the keypoints) of a single triangle. The 3D mesh is stored in a similar way: as a $n \times 3$ matrix where the ids correspond instead to the landmarks' ids. Therefore, after calculating the 2D triangulation matrix,

it is straightforward to build the corresponding 3D mesh matrix, by just changing the keypoints ids for corresponding landmark ids.

The problem comes when appending this new 3D mesh to the previous 3D mesh. As we said before, we are not just keeping a 3D mesh in the current frame, but one that spans the whole optimization's time-horizon – which potentially has multiple frames, and certainly multiple landmarks.

When the 2D Delaunay triangulation is performed from scratch, some rows of the matrix storing the 2D triangulation will be the same as the previous matrix (some triangles are still using the same triplets of keypoints, and thereby the same triplets of landmarks). Therefore, we will update the old global 3D mesh by appending the new local 3D mesh on the current frame, but making sure we do not add duplicated faces to the mesh.

– Old landmarks disappear, as they get marginalized from the optimization's time-horizon: in this case, we remove any row in the 3D mesh matrix that contains the offending landmark id. In this way, we keep the mesh anchored in the time-horizon of the optimization problem, which bounds the memory usage and the computational complexity of the problem. This is not without problems, since the removed landmark might have been at the center of a wall for example, thereby leaving a hole when surrounding faces of the mesh are deleted. While we did not attempt to solve this issue, the problem usually appears on the portion of the mesh that is not currently visible by the camera.

Mesh Filtering

Some artifacts are generated on the 3D mesh when projecting the 2D triangulation in 3D. In fact the connectivity at the 2D image level does not necessarily translate to 3D points lying on the same surface. Figure 3.16 shows two of the problems encountered. In particular, fig. 3.16a shows the problem of trying to mesh the edge between a vertical wall and the floor. If the 2D triangulation does not have an edge at the same location than the edge between the wall and the floor, it will result in a 3D mesh face that does not represent the actual surface. This problem is referred as feature preservation in the computer graphics literature [113]. Another source of errors, comes from the fact that the landmarks associated to the keypoints in the 2D triangulation have a noisy position estimate, and some might even be outliers. Figure 3.16b shows mesh faces that have one vertex which is a landmark outlier.

In our implementation, we have dealt with some of these misrepresentative faces of the mesh as follows:

- Mesh triangles with two acute angles: these are the silver triangles mentioned before that give little information about the planarity of the scene.
- Mesh triangles with a large ratio between largest and smallest side. These are triangles that have a very long edge compared to their smallest side. This filter removes those triangles that have an outlier landmark as vertex, since they tend to have two long edges touching the outlier, while conserving a small edge between the accurate landmarks. Two of the angles are close to 90 degrees, thereby skipping the first filter.
- We also remove triangles that have at least one edge that is longer than a threshold. Thereby removing triangles that are unreasonably large.

The outlier removal approaches described above promote isotropic triangles in the 3D mesh. These triangles are then used to extract high-level regularities of the scene.

We have also tried to refine the mesh using similar heuristics as in DPPTAM [50]. The most relevant one being the assumption that regions with an homogeneous color tend to belong to planar

surfaces. Unfortunately, applying this filter is very sensitive to the threshold on the color gradients in the image and as such we did not use it. Also it requires a great deal of computational load for a single-threaded application as ours, since it most compute the gradient over all the triangles surface to determine if the color is homogeneous.

3.2.2 Regularity Detection

The 3D mesh is key to extract high-level regularities in the scene. By reasoning in terms of the triangular faces of the mesh, we can extract the geometry in the scene in a non-iterative way (unlike RANSAC approaches [97]). The regularities between triangles that we are interested in are:

- Co-planarity: landmarks from the same planar surface in the scene should lie on the same plane.
- Parallelism/Orthogonality: pairs of plane estimates that correspond to parallel/orthogonal surfaces in the real world should also be parallel/orthogonal.

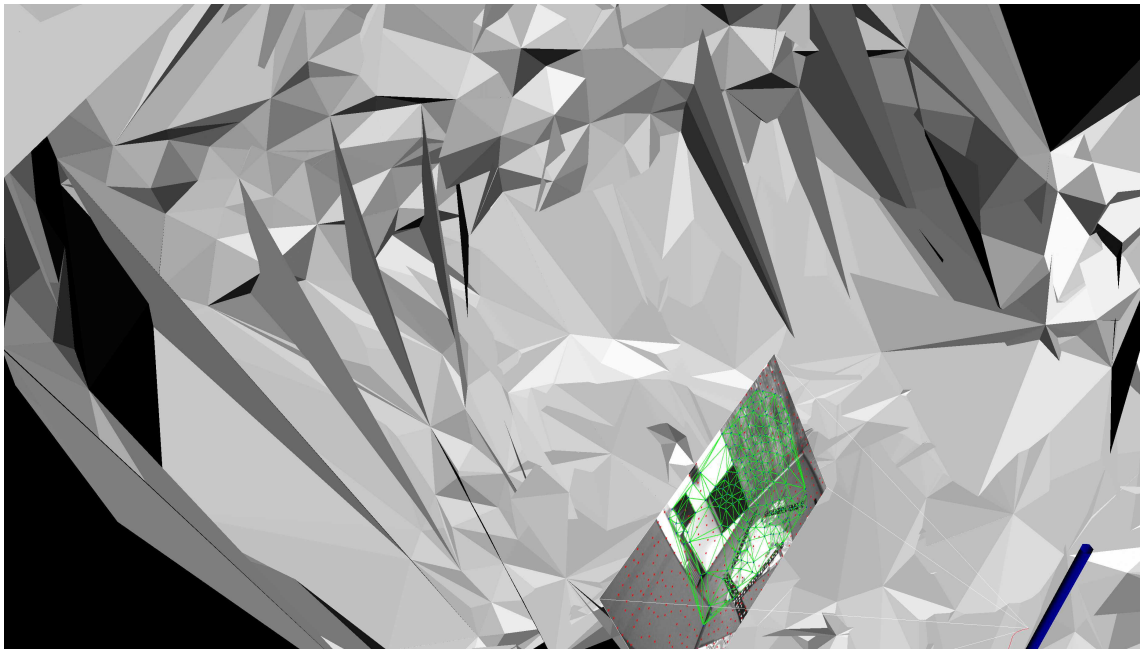
To find these properties we need to first find planar surfaces in the scene. Our approach, which we detail below, is versatile in that it can work on unstructured environments, and does not rely on assumptions about the world (such as the commonly used Manhattan World assumption [72]). Nevertheless, it is only capable to detect planes of two types due to its simplicity: planes that are vertical, such as walls, and planes that are horizontal, such as the floor, the ceiling or the surface of tables. Hence, if the scene has structural regularities of this type, our approach will capture and encode them gracefully in the optimization problem, and, if no structural regularities were to be present, it will downgrade to a classical VIO.

We rely nevertheless on having a good estimate of what the vertical direction is. This is an acceptable assumption since the gravity direction is observable in our system as we are using an IMU. For VIO pipelines, the error in pitch and roll is usually small [2].

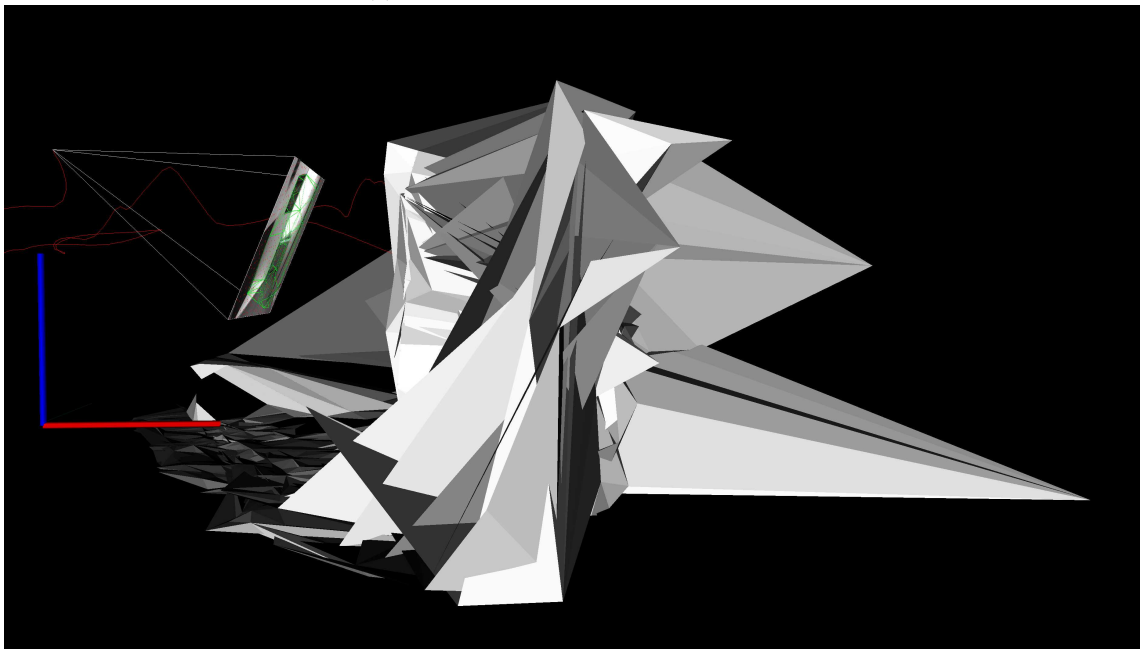
Co-Planarity

To find which landmarks in the optimization problem should have co-planarity constraints, we need to extract the planes in the scene. We attempt to extract planes that are horizontal and vertical:

- Horizontal planes
  1. Cluster faces of the mesh which have their normal parallel to the vertical direction.
  2. Build a 1D histogram of the height of the clustered faces' vertices, as in fig. 3.17a.
  3. Extract local maximums from the histogram. To extract local maximums we need to first apply a Gaussian filter over the data, otherwise multiple local maximums are present as can be seen in fig. 3.17a. The resulting smoothed histogram is then used to extract local maximums. We only keep the maximums that are supported by a minimum number of points, to make sure we are considering an extended surface, and that we are not dealing with a degenerate configuration (such as all points lying on a line, as explained in § 3.1.5). These local maximums correspond to estimates for the distance to the origin of planes with a normal vector $n = (0, 0, 1)^T$. In other words, horizontal surfaces such as the floor (which in histogram fig. 3.17b appears to be at a height of -0.38m) and a table top (which in the same histogram appears to be at -0.1 m). The fact that the



(a) Slanted triangles from ground to wall.



(b) Large triangles due to outlier landmarks.

Figure 3.16: Mesh without filters.

distances are negative should not bother the reader, since the world origin is set at the first pose estimate of the VIO (which is actually slightly on top of the detected table in the histogram), and not at the actual floor level.

4. From the plane candidates, extract a set of landmark inliers to be considered for the co-planarity constraint, as explained in more detail in the Data Association section 3.2.3. § 3.1.5 shows nevertheless how we enforce these regularities in the backend optimization once they are detected.

- Vertical planes.

1. Cluster faces of the mesh which have their normal perpendicular to the vertical direction.
2. Build a 2D histogram of the clustered mesh faces, where one axis represents the distance to the origin of the plane that the face represents, and the other is the azimuth of the normal with respect to the gravity direction (vertical axis). Therefore, the 2D histogram shows the density of faces of the mesh which represent a particular plane at a certain distance from the origin and with a certain orientation perpendicular to the vertical axis. Intuitevely, planar surfaces such as vertical walls will generate several mesh's faces that fall in one particular bin of the 2D histogram.
3. Extract local maximums from the 2D histogram. To extract local maximums we proceed as in the 1D histogram, where we need to first apply a Gaussian filter over the data to avoid multiple local maximums (the kernel of the filter is simply of size 5×5). The resulting smoothed histogram is then used to extract local maximums. We only keep the maximums that are supported by a minimum number of faces, to make sure we are considering a vertical wall.
4. From the wall candidates, we extract a set of landmark inliers to be considered for the co-planarity constraint as for the horizontal planes.

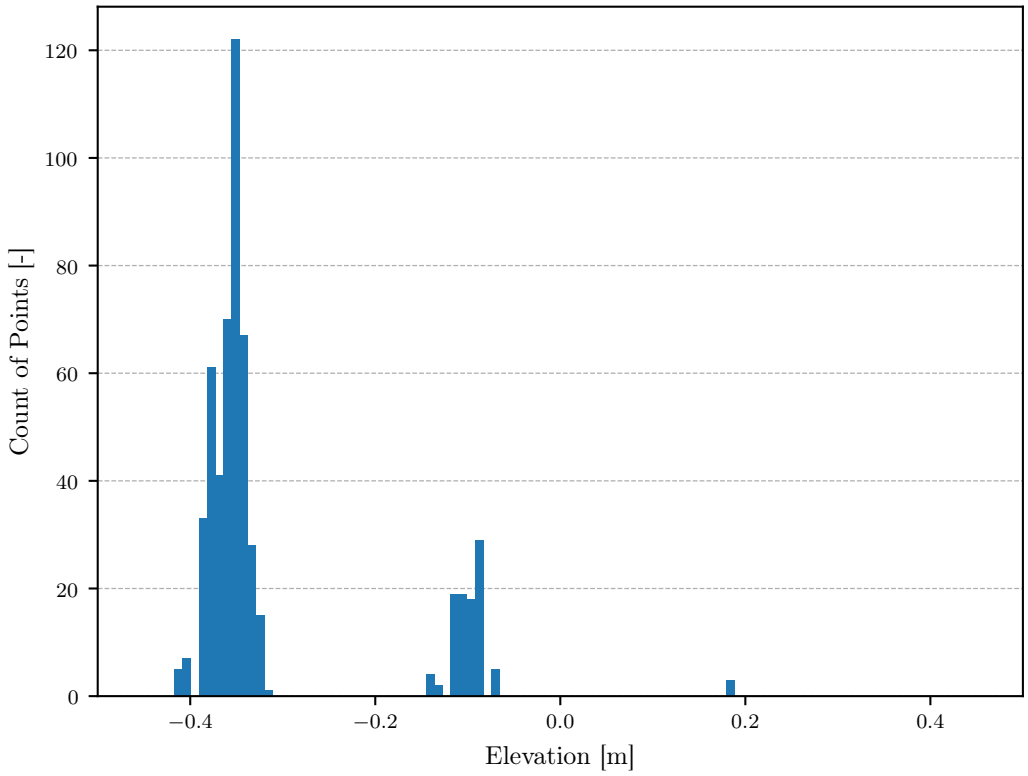
The result is that we are able to extract both horizontal and vertical planes. Figure 3.18 shows the actual planes detected at a given time. Painted in blue are the faces that support horizontal planes, and in green the faces that support vertical planes. The estimated horizontal planes are depicted as blue squares (in fig. 3.18 we can see the estimated floor plane at the origin of the coordinate system). A green square is also shown for the walls, but the density of mesh faces hides it. We also show as white lines the co-planarity constraints between the landmarks and the corresponding plane.

Parallelism

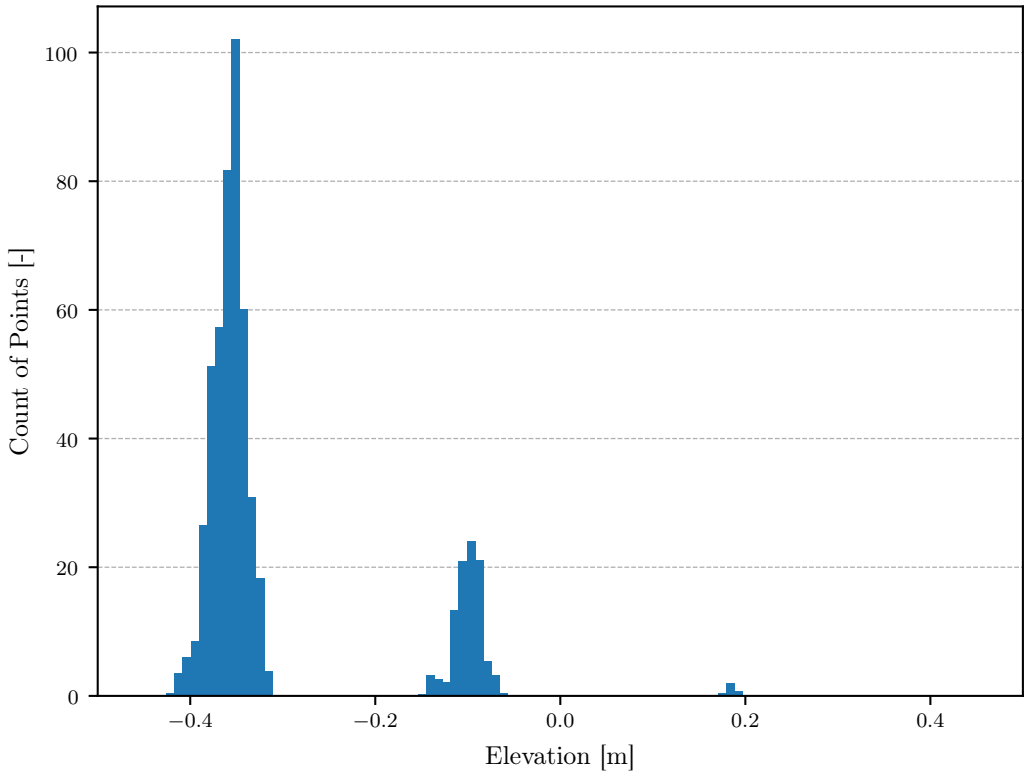
From the plane estimates detected previously we can easily enforce planarity constraints. For the horizontal planes, it is obvious that all candidates extracted from the 1D histogram should be parallel. For the vertical planes, we need instead to explore the dimension of the 2D histogram representing the azimuth. If two detected walls share the same orientation, they can then be constrained as parallel planes.

Orthogonality

On the one hand, the horizontal planes detected from the 1D histogram are by definition orthogonal to the walls detected using the 2D histogram. On the other hand, the vertical planes detected from the 2D histogram might also be orthogonal between them, which we have to check by looking again at the azimuth component of pairs of planes. If the azimuth is offset by 90 degrees, the planes are potentially orthogonal.



(a) Raw histogram.



(b) Smoothed histogram with a Gaussian filter (kernel of size 3).

Figure 3.17: Histogram of landmarks depending on their elevation. All landmarks considered are vertices of faces of the mesh, and we only consider those faces that have a normal parallel to the vertical axis.

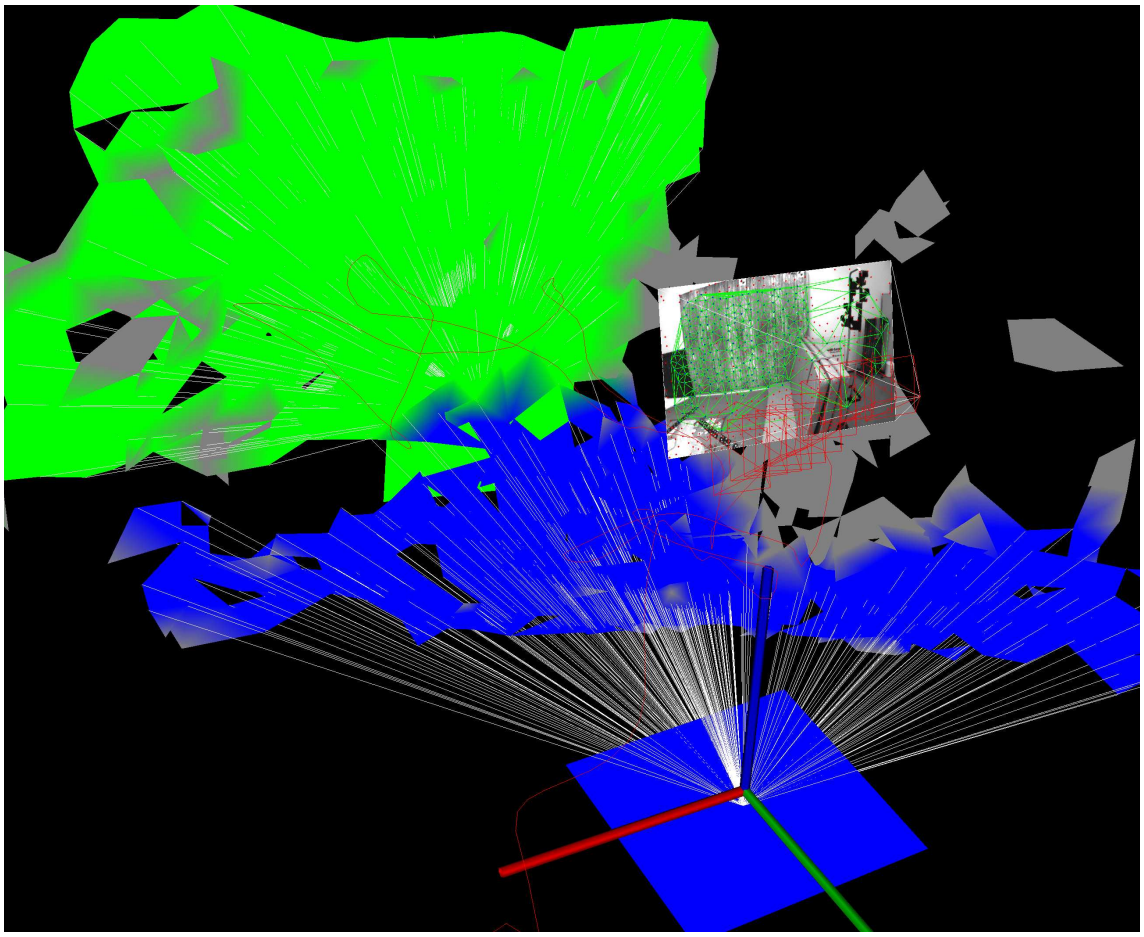


Figure 3.18: Plane detection on 3D mesh. Faces of the mesh that are segmented on vertical walls are colored in green, while the ones segmented on horizontal surfaces are colored in blue. A blue square shows what is the current estimate of the plane parameters for the floor plane. White lines from the plane to the mesh vertices show the constraints between landmarks and plane.

3.2.3 Data Association

Point to Plane Association

With the newly detected planes, we still need to associate which landmarks are on each plane. For this, we use the set of supporting landmarks in the original histogram for the given plane. We use hard associations for simplicity, instead of using a soft association approach within an Expectation Maximization framework [114].

Plane to Plane Association

Once we have a new set of planes detected, we also need to check whether a detected plane is already present in the optimization problem or not, to avoid duplicated plane variables.

For this, we simply compare the normals and distances to the origin of the plane to see if they are close to each other. Other authors might suggest to add another criteria to make this association more robust to false positives; such as ensuring that the plane which is in the optimization accurately fits through the points that are supporting the newly detected plane [74] (if it is the case then the planes are merged together).

Chapter 4

Results

The output of our pipeline is two fold: the pose of the camera at each timestep and a mesh-based representation of the scene.

To quantify the performance of the pose estimate we will be using both absolute and relative error metrics. These metrics will give us insights, respectively, on the global and local consistency of the trajectory estimate.

For the quality of the mesh we will be using a point cloud to point cloud distance as a metric to quantify how well the mesh represents the actual scene.

4.1 Dataset

The EuRoC MAV dataset [115] (for short EuRoC) contains Visual (stereo camera) and Inertial (IMU) data recorded from an MAV flying in different indoor scenes. There are eleven datasets in total, recorded in two different scenarios.

One of the scenarios is a so called *Machine Hall*, denoted by the letters `MH`, which is the interior of an industrial facility. It consists of little planar regularities, as neither the ceiling nor the floor are clearly visible, and there are few or no planar surfaces present.

The other one is a *Vicon Room*, denoted by the letter `V`, which is similar to an office room where walls, floor and ceiling are close together, and other planar surfaces are visible, as some boxes and stacked mattresses fill the room. There are in particular two types of Vicon Room datasets, `V1` and `V2`, which differ only on the position of the objects in the scene.

Moreover, each dataset is labelled by the level of difficulty it represents for Visual-Inertial SLAM algorithms: using the adjectives "`easy`", "`medium`", and "`difficult`". The difficulty is increased by simply increasing the speed of the MAV, which results in motion-blur and drastic illumination changes on the images. For example, dataset `MH_03_medium` corresponds to a dataset in the Machine Hall where the MAV was flying at moderate speeds.

Each dataset provides the ground-truth trajectory of the drone, allowing us to compare our estimated trajectory with the real one. For the Vicon Room datasets, we are also provided with an accurate ground-truth point cloud of the scene, which we will use to evaluate the quality of our mesh.

4.2 State Estimation

In our case, the state of our optimization problem comprises not only the poses of the camera, but also the IMU biases, the plane estimates, as well as the landmarks positions. Nevertheless, the most important state is the actual pose of the camera, since this is ultimately used to control a robot's position and orientation. Also, the plane and landmark estimates will be assessed in the next section, where we evaluate the quality of the mesh itself.

To grade the quality of the pose estimates we must first align both temporally and spatially the ground-truth trajectory with our estimated trajectory, as explained in § 4.2.1. Then, we will use two different metrics to quantify the global accuracy (§ 4.2.2) and the local accuracy (§ 4.2.3) of our trajectory estimate[1].

4.2.1 Data Alignment

Time Synchronization

We need to temporally align ground-truth poses to our estimated poses. First, we select the trajectory with the smallest amount of state estimates. Then, for each entry of this trajectory (which could be either the ground-truth trajectory or the estimated one) we look for the corresponding state estimate with the closest timestamp. Associated timestamps that differ more than a given threshold are discarded. In our evaluation this threshold is held constant at 0.01 seconds.

Trajectory Alignment

Once the state estimates are temporally aligned, we want to spatially align the trajectories. Mathematically speaking, given two sets of 3-dimensional points $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_i\}$, $i = 1, 2, \ldots, n$, we want to find the rotation matrix $\mathtt{R}$, translation vector $\boldsymbol{t}$, and scale factor c that minimize the mean squared error $e^2(\mathtt{R}, \boldsymbol{t}, c) = \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{y}_i - (c\mathtt{R}\boldsymbol{x}_i + \boldsymbol{t})\|^2$.

This problem can be solved in closed form using Umeyama's method[2] [117].

The scale is typically corrected when comparing the trajectories of monocular visual odometry, where the scale is not observable. In our case, the scale is observable because we are using a stereo camera and an IMU (any of these sensors would disambiguate the scale factor by itself). Therefore, we align our estimated trajectory and the ground-truth trajectory in SE(3) (rotation $\mathtt{R}$ and translation $\boldsymbol{t}$) instead of Sim(3)[3], which includes the scale parameter c.

Also, notice that when aligning the trajectories we ignore the actual orientation of the camera at each pose; we just use the translation part of the pose estimates.

4.2.2 Absolute Pose Error

The Absolute Pose Error (APE) is a metric for investigating the global consistency of a SLAM trajectory.

[1] We thank Michael Grupp for developing a great toolbox to evaluate odometry and SLAM: github.com/MichaelGrupp/evo

[2] Umeyama's method always gives the correct transformation parameters even when the data is corrupted, and, in this respect, it is an improved version over Horn's method [116].

[3] $\mathrm{Sim}(3) \doteq \{(\mathtt{R}, \boldsymbol{t}, c) : \mathtt{R} \in \mathrm{SO}(3), \boldsymbol{t} \in \mathbb{R}^3, c \in \mathbb{R}^+\}$

APE is based on the absolute relative pose between two poses $P_{ref,i}$, $P_{est,i} \in \mathrm{SE}(3)$ at timestamp i:

$$E_i = P_{ref,i}^{-1} P_{est,i} = P_{est,i} \ominus P_{ref,i} \in \mathrm{SE}(3)$$

where we introduce the inverse compositional operator $\ominus$, which takes two poses and gives the relative pose, as defined in [118].

We can use different pose relations to calculate the APE:

- Using the translation part of E_i:

$$APE_i = \|\mathrm{trans}(E_i)\| \tag{4.1}$$

- Using the rotation angle of E_i:

$$APE_i = |\mathrm{angle}(\log_{\mathrm{SO}(3)}(\mathrm{rot}(E_i))| \tag{4.2}$$

  $\log_{\mathrm{SO}(3)}(\cdot)$ is the inverse of $\exp_{\mathfrak{so}(3)}(\cdot)$ (Rodrigues' formula)

- Using the rotation part of E_i:

$$APE_i = \|\mathrm{rot}(E_i) - I_{3\times 3}\|_F$$

- Using the full relative pose E_i:

$$APE_i = \|E_i - I_{4\times 4}\|_F,$$

where we use the Frobenius norm $\|A\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$; a_{ij} is the entry at row i and column j of the $m \times n$ matrix A .

Used approach

The absolute pose error of the final points in the trajectory can be a bad indicator of the performance of a state estimation pipeline, since this metric is heavily influenced by the instance where the estimation errors are made. For example, rotational errors happening at the beginning of the trajectory result in larger pose errors at the end poses.

To limit the influence of this bias, we calculate the APEs at all timestamps, and report the statistics of these errors. In particular, we report the minimum, the maximum, the median, as well as the first and third quartile of the APE_i errors in translation eq. (4.1). For a better visualization of these statistics, we use boxplots when presenting the results in fig. 4.1. We also report in table 4.1 the mean, the median and the Root Mean Squared Error (RMSE) of the APE errors in translation eq. (4.1), this time just for the pipeline using Structureless and Projection factors (S + P) and the one using Regularity factors (S + P + R), for an in-depth comparison in the following § 4.2.2.

We refrain from using the APE_i in rotation since the trajectory alignment ignores the orientation component of the pose estimates (as shown in § 4.2.1).

We also report the RMSE to be able to compare the performance of our pipeline against other VIO algorithms; such as OKVIS [4], MSCKF [15], ROVIO [18], VINS-MONO [9], and SVO-GTSAM [2] using the reported values in [1] (see table 4.2). These algorithms currently represent the state-of-the-art in VIO; OKVIS/VINS-MONO (keypoint-based) and SVO-GTSAM

(semi-dense photometric tracking) for optimization-based approaches, and MSCKF and ROVIO for filtering methods. We refer the reader to [1] for details on the particular implementations and set of parameters used for each algorithm. Note that, while VINS-MONO can handle loop closures, we only report the values when the loop-closure module is disabled, in order to be fair in the comparison with the rest of the algorithms.

Finally, we found to be instructive to color-encode the estimated trajectory with the actual APE errors at each pose estimate; which provides insights on how quickly the state estimation degrades (fig. 4.2). To know exactly the point in time when the errors occur we refer the reader to the later § 4.2.3, that explores instead relative errors from pose to pose.

APE results

Figure 4.1 gives us an excellent overview of the performance of the different pipelines considered in this thesis, and makes explicit the different capabilities of our proposed approach using structural regularities.

First, if we look at the performance of the different pipelines for the datasets `MH_03_medium`, `MH_04_difficult` and `MH_05_difficult` in fig. 4.1, we observe that all pipelines perform equally. This is because in these datasets no structural regularities were detected. Hence, our proposed pipeline using Strutureless, Projection and Regularity factors (S + P + R) gracefully downgrades to a pipeline using solely Structureless factors (S) when no regularities are detected.

Second, looking at the performance of the pipelines for dataset `V2_03_difficult`, we can observe that both the pipeline using Structureless and Projection factors (S + P) and the S + P + R pipeline perform equally, and better than the S pipeline. In this case, structural regularities are detected, leading to a conversion of Structureless factors to Projection factors. Nevertheless, since the number of regularities detected is not sufficient to spawn a new plane estimate and enforce its corresponding constraints with the landmarks, no structural regularities are actually used in the factor graph. This leads to the S + P + R pipeline having the same exact performance as the S + P pipeline, while achieving different results than the S pipeline.

Finally, we can observe in fig. 4.1 that the S + P + R pipeline consistently achieves better results over the rest of datasets where structural regularities are detected and enforced.

For the remaining of the state estimation analysis, we will just compare the S + P pipeline against our proposed S + P + R pipeline as this will best show the impact of the regularity factors on the state estimation performance.

Table 4.1 shows an in-depth comparison between S + P and the S + P + R pipelines. It shows that the S + P + R pipeline consistently achieves better translational APE results over all datasets exhibiting structural regularities than the S + P pipeline, while performing equally for the datasets where no structural regularities were enforced. In particular, the performance increases up to 28% on the median APE, 24% on the mean, and 26% on the RMSE for datasets with multiple planes such as `V1_01_easy`, `V1_02_medium` and `V2_02_medium`.

This performance increase is clearly visible when plotting the APE errors on the trajectory, as done in fig. 4.2 for dataset `V1_01_easy`. Figure 4.3 shows in detail the effects of using structural regularities on the APE. We provide these plots for all datasets as well in appendix A.

Finally, we compare our results with the state-of-the-art in table 4.2.

We can extract two main conclusions from this comparison:

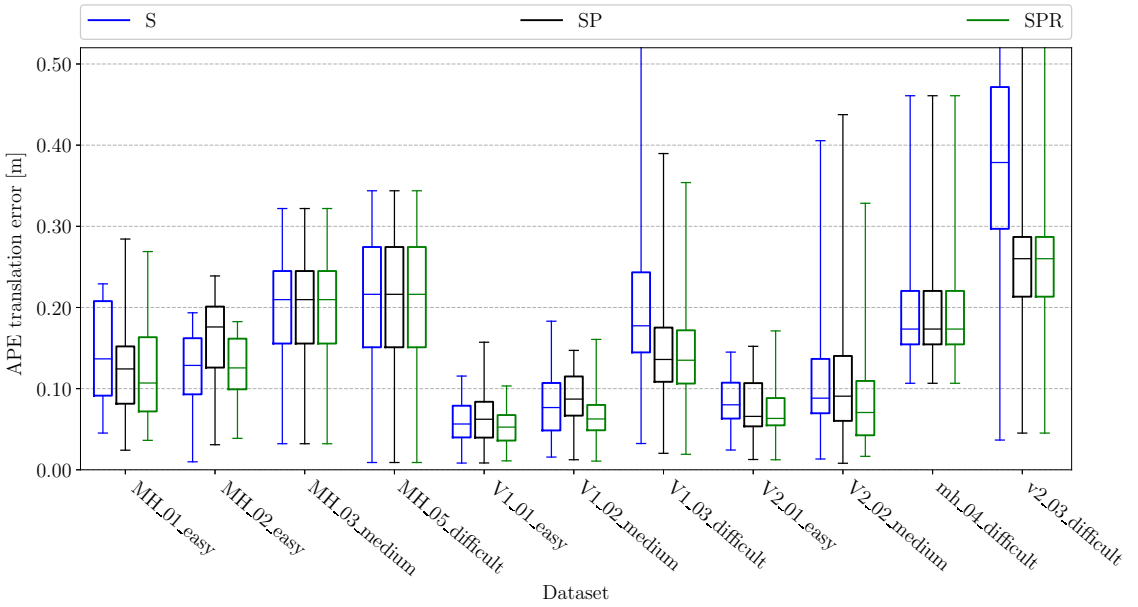
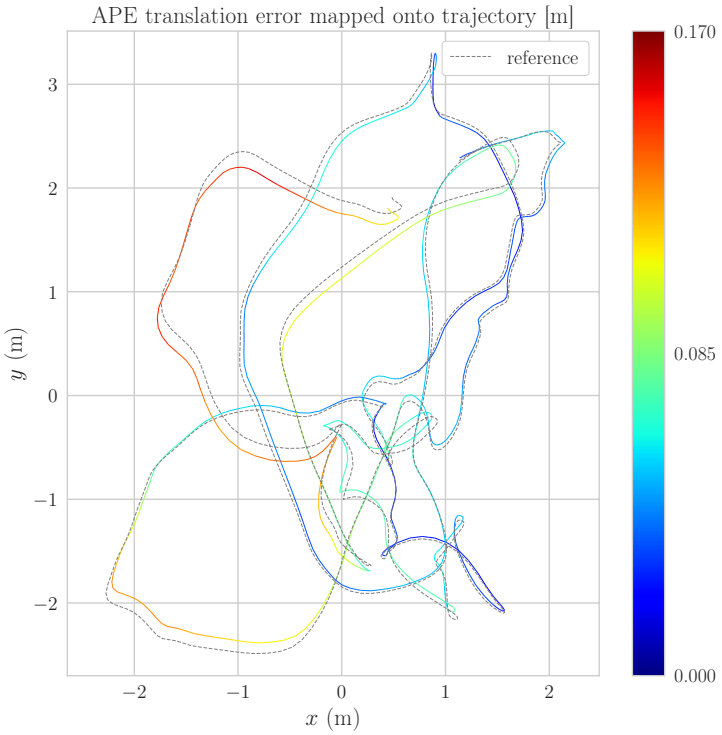


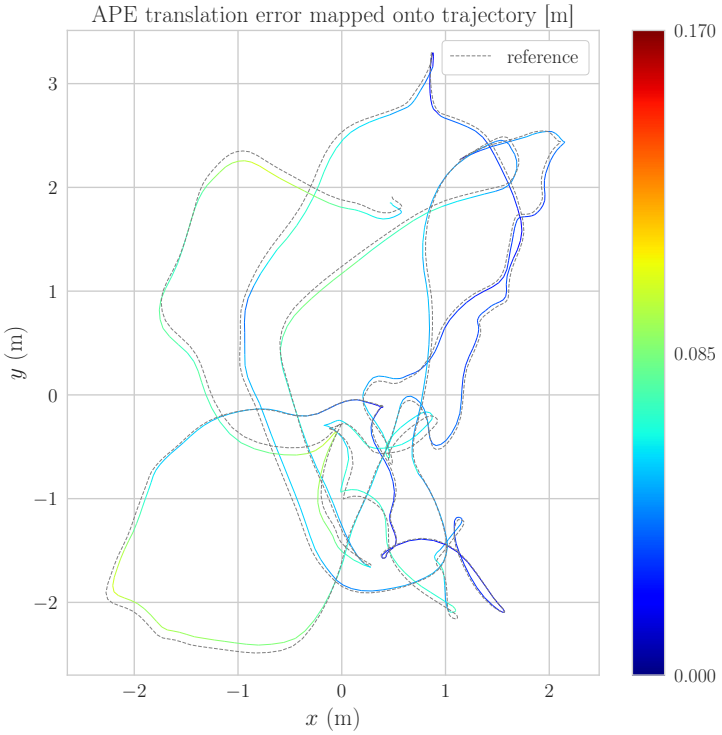
Figure 4.1: Comparison of the Absolute Pose Error (APE) on the EuRoC datasets while using Structureless factors (S), structureless and Projection factors (S + P), and our proposed approach using Structureless, Projection and Regularity factors (S + P + R).

Table 4.1: Accuracy of the state estimation when using Structureless factors (S), Structureless and Projection factors (P), and our proposed approach using Structureless, Projection and Regularity factors (R).

	APE Translation					
	S + P			**S + P + R** (Proposed)		
Sequence	Median [cm]	Mean [cm]	RMSE [cm]	Median [cm]	Mean [cm]	RMSE [cm]
MH_01_easy	12.4	13.3	15.0	**10.7**	**12.7**	**14.5**
MH_02_easy	17.6	15.8	16.7	**12.6**	**12.4**	**13.0**
MH_03_medium	21.0	20.3	21.2	21.0	20.3	21.2
MH_04_difficult	17.3	20.4	21.7	17.3	20.4	21.7
MH_05_difficult	21.6	21.0	22.6	21.6	21.0	22.6
V1_01_easy	6.2	6.9	7.7	**5.3**	**5.3**	**5.7**
V1_02_medium	8.7	8.9	9.4	**6.3**	**6.8**	**7.4**
V1_03_difficult	13.6	15.6	17.6	**13.5**	**15.1**	**16.7**
V2_01_easy	6.6	7.5	8.2	**6.3**	**7.5**	**8.1**
V2_02_medium	9.1	11.2	13.5	**7.1**	**8.5**	**10.3**
V2_03_difficult	26.0	25.7	27.2	26.0	25.7	27.2

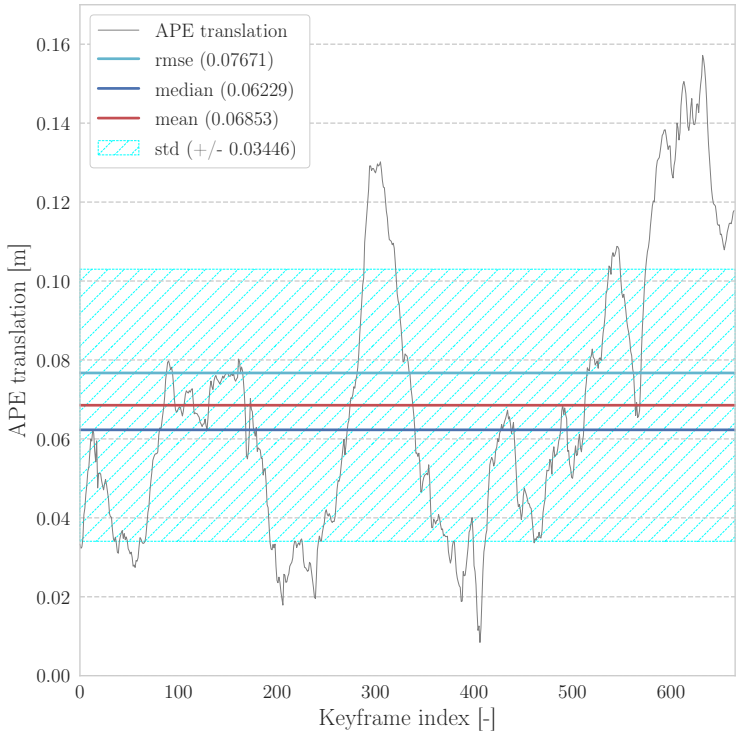


(a) APE translation for S+P

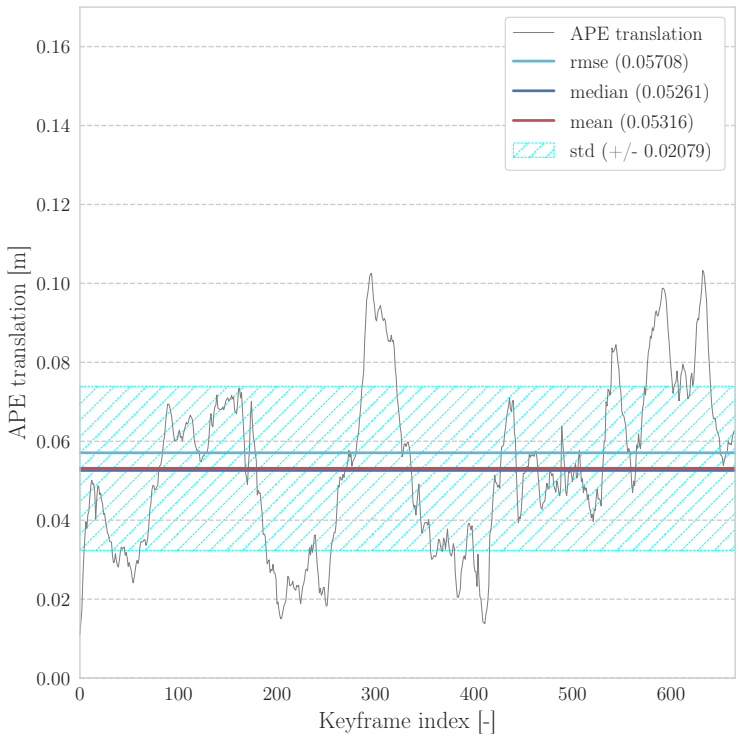


(b) APE translation for S+P+R

Figure 4.2: Dataset `V1_01_easy`: APE translation error plotted on the trajectory estimated by VIO using Strutureless and Projection factors (S + P), against our proposed approach using also Regularity factoris (S + P + R).



(a) S+P



(b) S+P+R

Figure 4.3: Dataset `V1_01_easy`: APE translation error of VIO using Strutureless and Projection factors (S + P), against our proposed approach using Structureless, Projection and Regularity factors (S +P + R).

(i) Our approach using structural regularities (S + P + R) achieves the best results when compared with the state-of-the-art on datasets which have structural regularities; such as in datasets `V1_01_easy` and `V1_02_medium`, where multiple planes are present (walls, floor, etc.). We observe a 19% improvement compared to the next best performing algorithm (SVO-GTSAM) in dataset `V1_01_easy`, and a 26% improvement in dataset `V1_02_medium` compared to ROVIO and VINS-MONO, which achieve the next best results. Moreover, as explained previously, we used the same parameters for all datasets (no per dataset fine-tuning), and we are still not enforcing higher level regularities (such as parallelism/orthogonality) between plane estimates. Therefore, the improvements could potentially be larger.

(ii) Our implementation of the S and S + P pipelines (and implicitly S + P + R) still need to be improved. We arrive at this conclusion by looking at the performance of SVO-GTSAM. SVO-GTSAM is in essence very similar to the pipeline using just Structureless factors (S) – at least in terms of the backend used – yet SVO-GTSAM performs much better than the S pipeline in all `MH` datasets. Indeed, we both use Structureless (Smart) and Pre-integrated IMU factors for the factor-graph formulation. Moreover, we both use GTSAM to build the factor-graph, as well as ISAM2 to solve the optimization problem. The main difference being that SVO-GTSAM's frontend is semi-dense and based on photometric errors, while ours is sparse and based on reprojection errors. While the frontends are quite different, it seems unlikely that this alone can explain why the performance is so different. Also, the S + P pipeline should be more accurate than the SVO-GTSAM pipeline, since we are using Projection factors which are known to be more accurate than Structureless factors [119]. Note that, if we ignore SVO-GTSAM's results, our pipeline is still doing very good in the `MH` datasets.

We can also see that SVO-GTSAM fails for datasets `V1_03_difficult`, `V2_02_medium`, and `V2_03_difficult`, while ours keeps providing accurate state estimates. SVO-GTSAM fails in these datasets because of motion-blur and fast changes in brightness on the images, which make state estimation using direct photometric tracking difficult (although new event-cameras have proved to minimize this problem [120]).

Table 4.2: Comparison of the RMSE of the APE in translation for OKVIS, MSCKF, ROVIO, VINS-MONO and SVO-GTSAM (reported values from [1]), against our proposed S + P + R pipeline. A cross (×) states that the pipeline failed.

Sequence	RMSE APE translation [cm]					
	OKVIS	MSCKF	ROVIO	VINS-MONO	SVO-GTSAM	S+P+R (Proposed)
MH_01_easy	16	42	21	27	**5**	14.5
MH_02_easy	22	45	25	12	**3**	13.0
MH_03_medium	24	23	25	13	**12**	21.2
MH_04_difficult	34	37	49	23	**13**	21.7
MH_05_difficult	47	48	52	35	**16**	22.6
V1_01_easy	9	34	10	7	7	**5.7**
V1_02_medium	20	20	10	10	11	**7.4**
V1_03_difficult	24	67	14	**13**	×	16.7
V2_01_easy	13	10	12	8	**7**	8.1
V2_02_medium	16	16	14	**8**	×	10.3
V2_03_difficult	29	113	**14**	21	×	27.2

While the APE can give meaningful insights on the global consistency of the trajectory estimate, it does not provide insights on the moment in time when the erroneous estimates happen.

To obtain this information we calculate instead the Relative Pose Error as explained in the next § 4.2.3.

4.2.3 Relative Pose Error

The Relative Pose Error (RPE) is a metric for investigating the local consistency of a SLAM trajectory.

RPE compares the relative poses along the estimated and the reference trajectory. This is based on the delta pose difference $\delta_{est_{i,j}} = P_{est,i} \ominus P_{est,j}$:

$$E_{i,j} = (P_{ref,i}^{-1} P_{ref,j})^{-1} (P_{est,i}^{-1} P_{est,j}) = (P_{est,j} \ominus P_{est,i}) \ominus (P_{ref,j} \ominus P_{ref,i}) = \delta_{est_{i,j}} \ominus \delta_{ref_{i,j}} \in \mathrm{SE}(3)$$

We can use different pose relations to calculate the RPE from timestamp i to j:

- Using the translation part of $E_{i,j}$:

$$RPE_{i,j} = \|\mathrm{trans}(E_{i,j})\| \tag{4.3}$$

- Using the absolute angular error of $E_{i,j}$:

$$RPE_{i,j} = |\mathrm{angle}(\log_{\mathrm{SO}(3)}(\mathrm{rot}(E_{i,j}))| \tag{4.4}$$

  where $\log_{\mathrm{SO}(3)}(\cdot)$ is the inverse of $\exp_{\mathfrak{so}(3)}(\cdot)$ (Rodrigues' formula)

- Using the rotation part of $E_{i,j}$:

$$RPE_{i,j} = \|\mathrm{rot}(E_{i,j}) - I_{3\times 3}\|_F$$

- Using the full delta pose difference $E_{i,j}$:

$$RPE_{i,j} = \|E_{i,j} - I_{4\times 4}\|_F$$

The above RPE metrics can be combined in multiple ways to offer different assessments of the performance of the state estimate. For example, [121] suggested to use the sum of the squared translational and rotational parts of $E_{i,j}$, resulting in the following metric:

$$RPE(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} trans(E_{i,j})^2 + rot(E_{i,j})^2 \tag{4.5}$$

where $\mathcal{F}$ is the set of pairs of frames considered, $|\mathcal{F}|$ the number of relative relations, and $trans(\cdot)$ and $rot(\cdot)$ are used to separate and weight the translational and rotational components; and differ from our definitions of trans$(\cdot)$ and rot$(\cdot)$ in that they have a weighting component encoded.

We will be nevertheless using the metric suggested in [122]. Which we believe is a more transparent metric to evaluate the performance of the state estimation, since it decouples translational and rotational errors. It consists on taking the mean of the $RPE_{i,j}$ defined in (4.3) and (4.4):

$$RPE_{trans}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} \|\mathrm{trans}(E_{i,j})\|_2 \tag{4.6}$$

$$RPE_{rot}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} |\text{angle}(\log_{\text{SO}(3)}(\text{rot}(E_{i,j}))| \tag{4.7}$$

Unfortunately, this metric, (4.6) and (4.7), is not as convenient as (4.5), since we now have two different numbers to compare a pair of algorithms; and while one might be achieving good results in terms of translational errors, it might perform poorly in terms of rotational errors. Justifying which algorithm is best therefore rests in the reader's hands.

Notice that none of the metrics suggested explicitly states how do we choose which relative displacements $\delta_{i,j}$ are included in the set $\mathcal{F}$. There seem to be at least three reasonable ways to choose those:

(i) Calculating the error between all possible pairs of measurements in the trajectory, as it is done by default in the RPE's calculation used in TUM's RGB-D SLAM dataset [123]. A caveat of this approach is that the number of pair of measurements in the trajectory is quadratic in the length of the trajectory, therefore downsampling is usually considered. How we downsample the trajectory must be specified when comparing the output of different algorithms, since merely using a uniform random sampling would lead to different results even when using the same algorithm.

(ii) Another alternative is to compare each pose to a later pose according to a window size defined in a convenient unit (e.g. number of frames, distance travelled, time difference). Therefore, we are comparing overlapping windows of estimates.

(iii) If we do not want overlapping windows to be considered, instead of calculating the RPE for each pose, we can use consecutive poses at a fixed window size. For example, if we consider a window size of two frames, we would calculate the RPE for the first pose against the third, and consecutively the third one to the fifth. This method, is ideal to visualize relative errors plotted on the trajectory itself, but might not give a complete picture of the performance of the algorithm as we are deliberately ignoring estimates.

Also note that the RPE is independent of the spatial alignment of the trajectories, and in this respect it provides a metric that just depends on the accuracy of the state estimate and the temporal alignment of the trajectories.

Used approach

To evaluate our pipeline we will be using the last two approaches, (ii) and (iii), for the metrics defined in (4.3) and (4.4).

For the approach (ii), we plot the statistics of all the calculated $RPE_{i,j}$ at different window sizes defined in terms of the distance travelled. These statistics include the maximum and the minimum $RPE_{i,j}$ achieved, the mean (as in (4.6) and (4.7)), as well as the values of the first and third quartile. Instead of using a table to report these values, we use boxplots for an eye-friendly comparison, as in fig. 4.5.

For the approach (iii), we calculate $RPE_{i,j}$ for a window size also defined in terms of length travelled, but instead of computing the statistics as before, we directly plot the error on the 3D trajectory by color-encoding the segments of the trajectory, as in fig. 4.6.

RPE results: approach (ii)

First, the boxplots in fig. 4.4 confirm one of the conclusions that we draw when looking at the APE results: our pipeline gracefully downgrades to a S pipeline when there are no regularities in the scene (like in dataset `MH_03_medium`, fig. 4.4a), and downgrades to a S + P pipeline when regularities are detected but not enforced (like in dataset `V2_03_difficult`, fig. 4.4b).

Second, we can evaluate the accuracy increase when using regularities. For this, we collect the medians $\widetilde{RPE}$ of the RPE errors for the different segment lenghts that we used to evaluate both the S + P and S + P + R pipelines, and compute the percentage improvement as:

$$\Delta\widetilde{RPE}_n = \frac{\widetilde{RPE}_n^{S+P} - \widetilde{RPE}_n^{S+P+R}}{\widetilde{RPE}_n^{S+P}} \times 100, \tag{4.8}$$

where $\widetilde{RPE}_n^{S+P}$ is the RPE of the n^{th} segment length used to evaluate the pipeline S + P. For example, in fig. 4.5a, we used 5 different segment lengths to evaluate the RPE, therefore $\widetilde{RPE}_1^{S+P}$ refers to the median of the RPE errors calculated for the first segment length (which is 10m long), for the S + P pipeline.

We then calculate the mean of these improvements as:

$$\Delta\widetilde{RPE} = \frac{1}{N}\sum_{n=1}^{N}\Delta\widetilde{RPE}_n, \tag{4.9}$$

where N is the number of segment lenghts used ($N = 5$ for fig. 4.5a).

Calculating the different $\Delta\widetilde{RPE}_n$, for datasets `V1_02_medium` (fig. 4.5a) and `V2_02_medium`(fig. 4.5b), we observe that using our proposed pipeline S + P + R leads to:

- An accuracy improvement of up to 50% in translation and 30% in rotation (maximum values of $\Delta\widetilde{RPE}_n$, achieved on dataset `V2_02_medium` fig. 4.5b), and,
- An average improvement of $\Delta\widetilde{RPE} = 20\%$ in translation and 15% in rotation with respect to S + P pipeline.

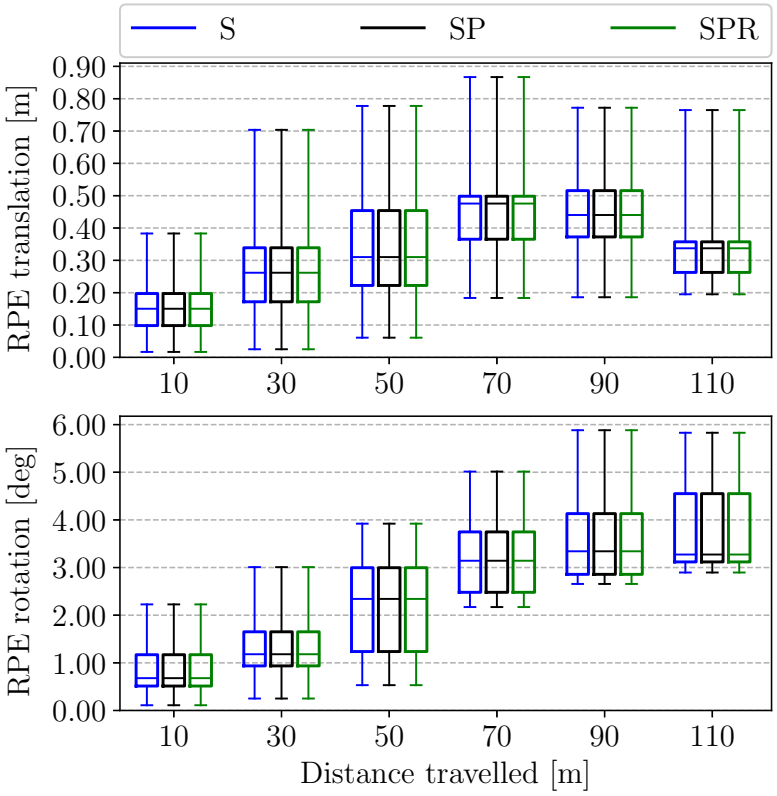
While these improvements are significant, we are not yet enforcing higher level regularities (such as parallelism or orthogonality) between planes. Therefore, these improvements could be even larger, potentially rivaling pipelines enforcing loop-closures.

We provide boxplots for the RPE in translation and rotation for all datasets in the appendix B.

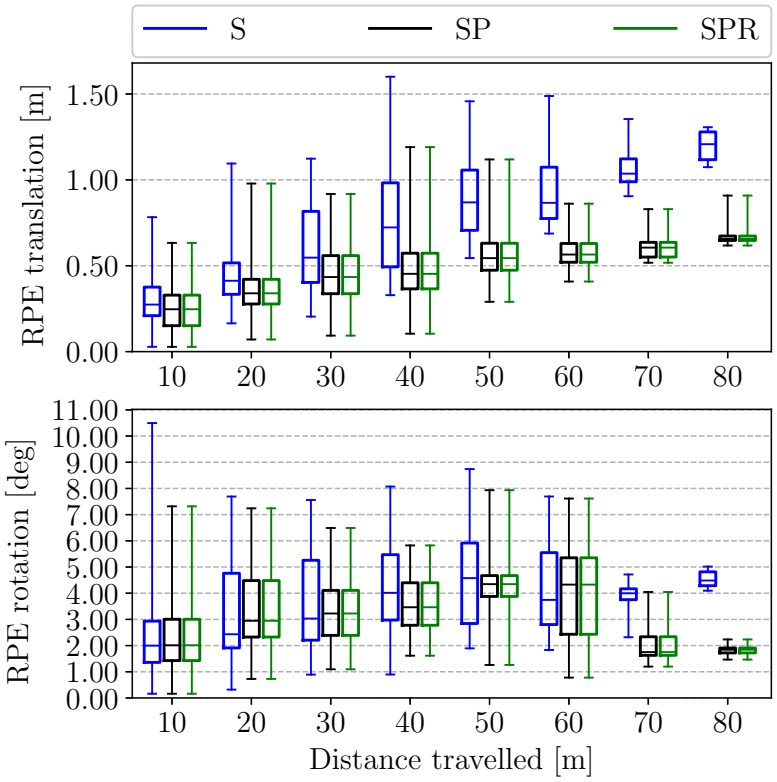
RPE results: approach (iii)

Finally, fig. 4.6 provides an example of the visualization of the RPE errors in translation plotted on the trajectory, which gives us precise insights on where the errors have occurred, and it is more meaningful in this sense than the APE errors plotted on the trajectory, which do not provide information on the exact point where the errors originated. Alternatively, we can spot which keyframe generated the largest errors by looking at fig. 4.7, which simply plots the RPE errors per keyframe.

Figures 4.6 and 4.7 must nevertheless be treated with caution, since they provide a narrow view of the actual performance of the pipeline. Indeed, we are using consecutive frames to compute

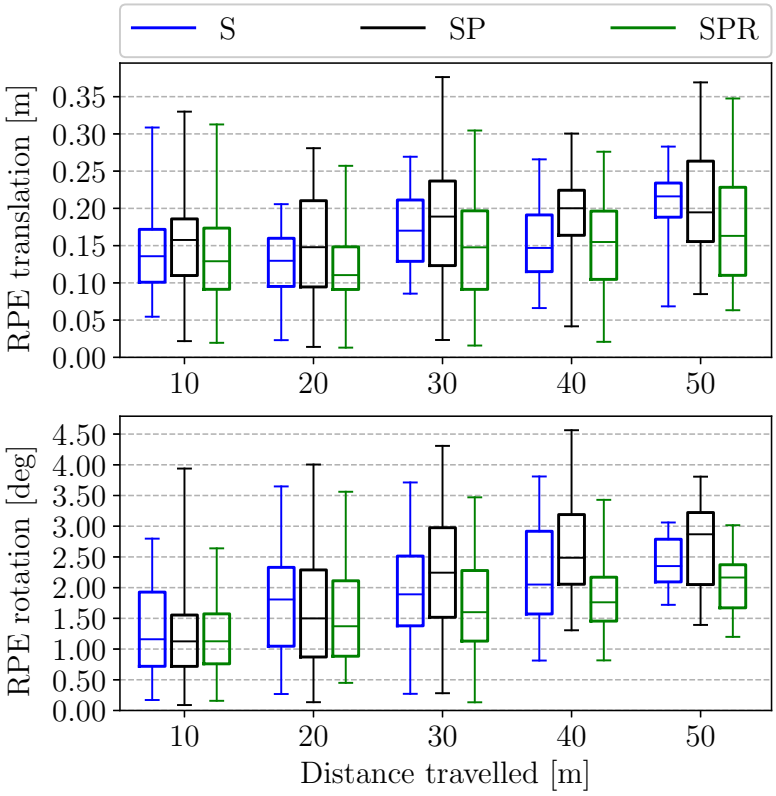


(a) `MH_03_medium`

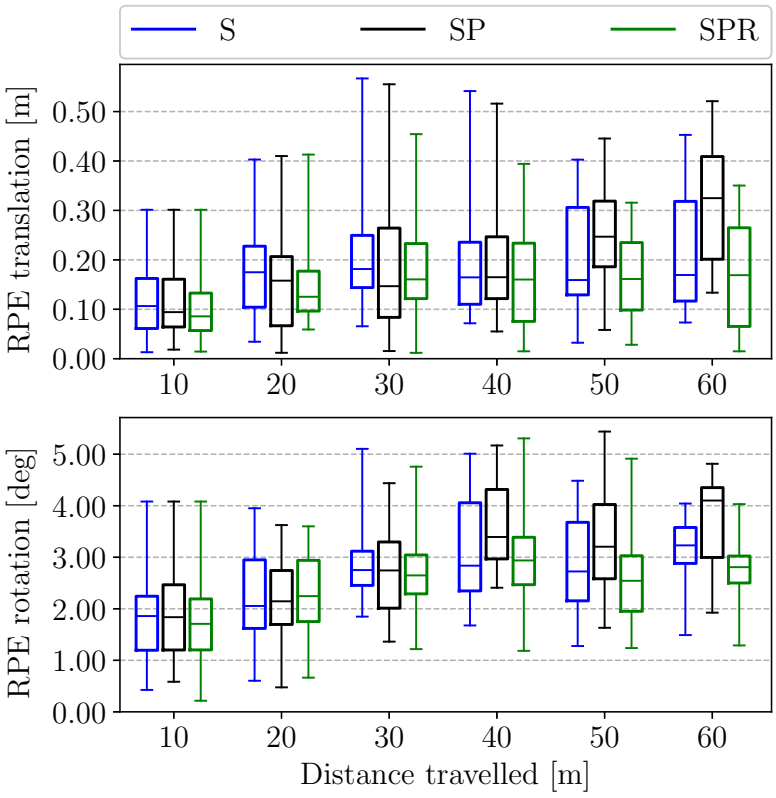


(b) `V2_03_difficult`

Figure 4.4: Detailed comparison of the state estimation accuracy while using Structureless factors (S), Structureless and Projection factors (SP), and our proposed approach using Structureless, Projection and Regularity factors (SPR) on EuRoC's V1_02_medium and V2_02_medium datasets.



(a) `V1_02_medium`



(b) `V2_02_medium`

Figure 4.5: Detailed comparison of the state estimation accuracy while using Structureless factors (S), Structureless and Projection factors (SP), and our proposed approach using Structureless, Projection and Regularity factors (SPR) on different EuRoC datasets.

the RPE, which is susceptible to a particular set of errors, such as errors in feature tracking due to brightness changes, instead of capturing the overall performance of the pipeline, and in this respect the RPE boxplots provide a clearer perspective.

We provide as well plots of all trajectories color-encoded with the RPE in translation C and rotation D in the appendix.

4.3 Mapping quality

We use the ground-truth point clouds provided on the EuRoC dataset (fig. 4.8) to assess the quality of the mesh by calculating its *accuracy* (§ 4.3.2) and *completeness* (§ 4.3.3), as defined in [124]. The evaluation is done using the freely available open-source software Cloud Compare [125]. Before evaluating these metrics, we must pre-process the mesh for a suitable comparison with the ground-truth point cloud (§ 4.3.1).

4.3.1 Pre-processing

Due to their different nature, in order to compare the estimated mesh against the ground-truth point cloud, we must perform some preliminary steps that we detail below:

Mesh Sampling

Comparing a mesh with sparse vertices with a dense point cloud can be achieved by generating a point cloud from the mesh itself, and then comparing both point clouds. In our case, we compute a point cloud by sampling the mesh with a uniform density of 1000 points per square meter.

Point Cloud Registration

We register the resulting point cloud to the ground-truth point cloud using an Iterative Closest Point (ICP) algorithm.

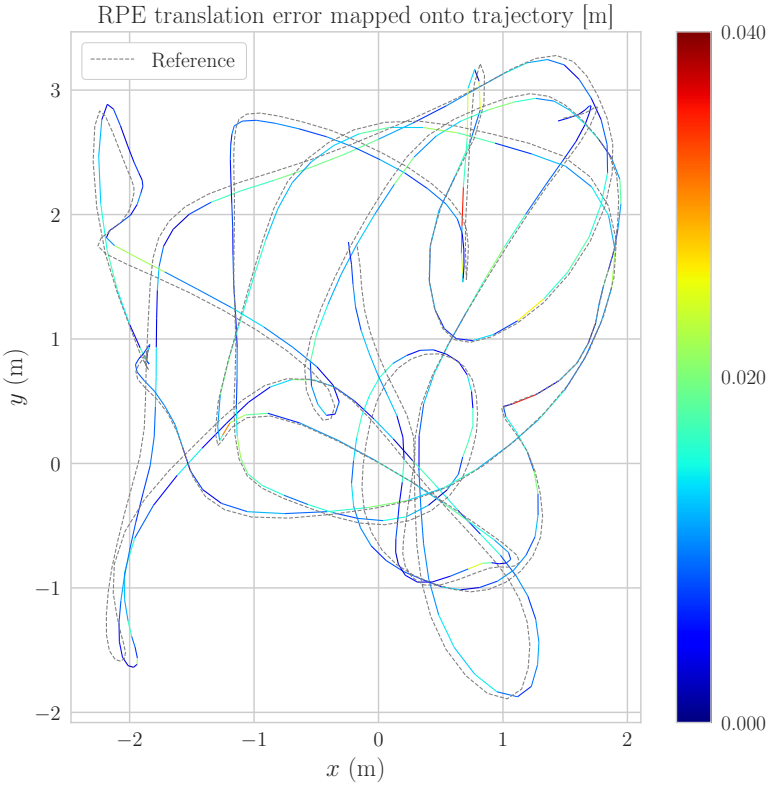
4.3.2 Map Accuracy

With the newly registered point cloud, we can compute a cloud to cloud distance to assess the accuracy of the mesh relative to the ground-truth point cloud.

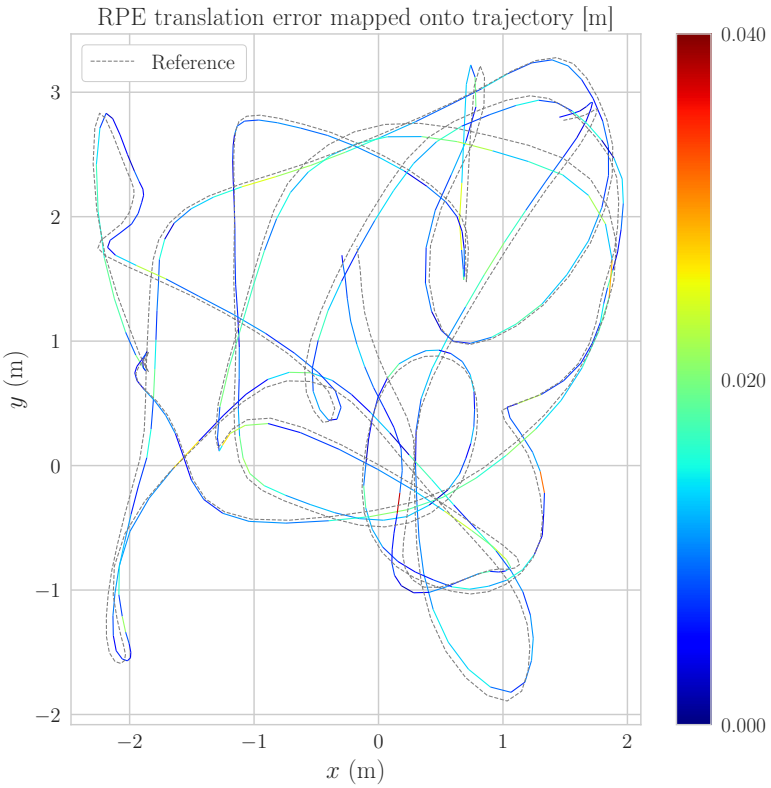
Used Approach

We compute the cloud to cloud absolute distance using the nearest neighbour distance. For each point of the estimated cloud from the mesh, we search the nearest point in the reference cloud and compute their Euclidean distance.

Mathematically, if we define by $\mathcal{R}$ our estimated point cloud, and by $\mathcal{G}$ our ground-truth point cloud, the nearest neighbour distance from any point in $\mathcal{R}$ to the ground-truth point cloud

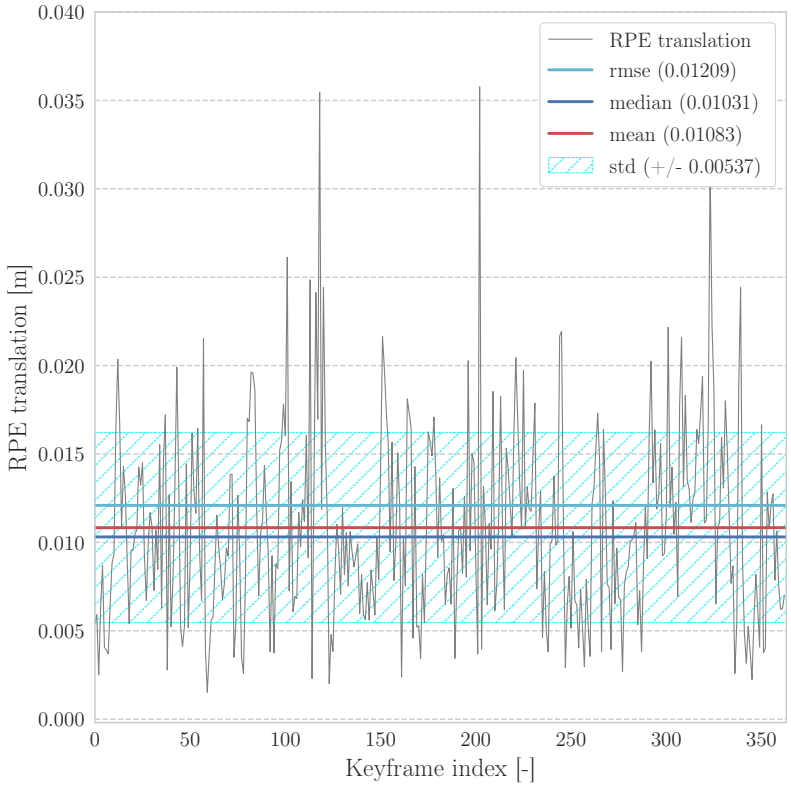


(a) RPE translation for S+P

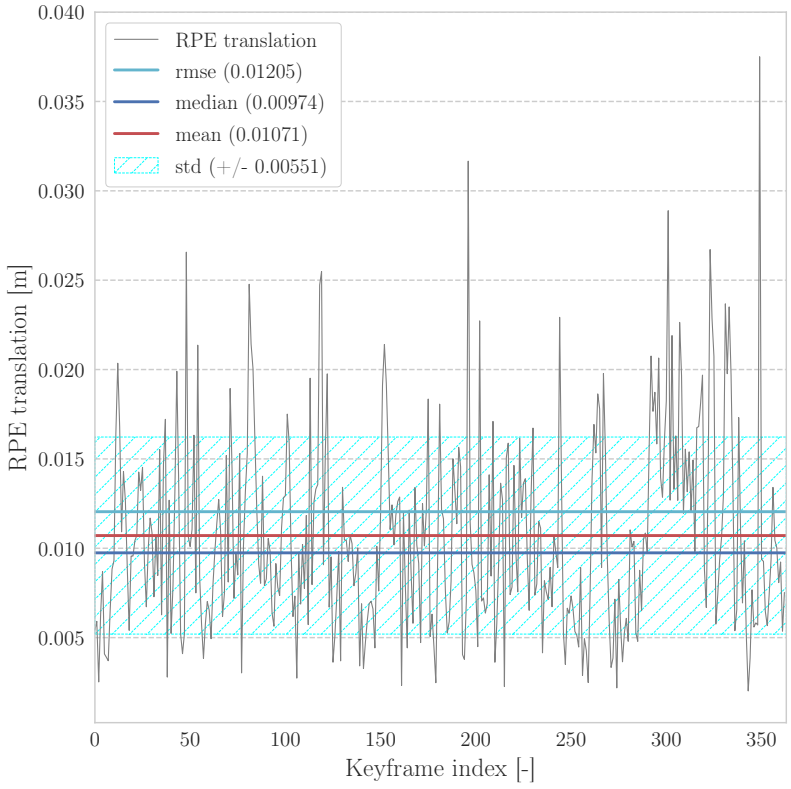


(b) RPE translation for S+P+R

Figure 4.6: Dataset `V1_02_medium`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.



(a) S+P



(b) S+P+R

Figure 4.7: Dataset `V1_02_medium`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

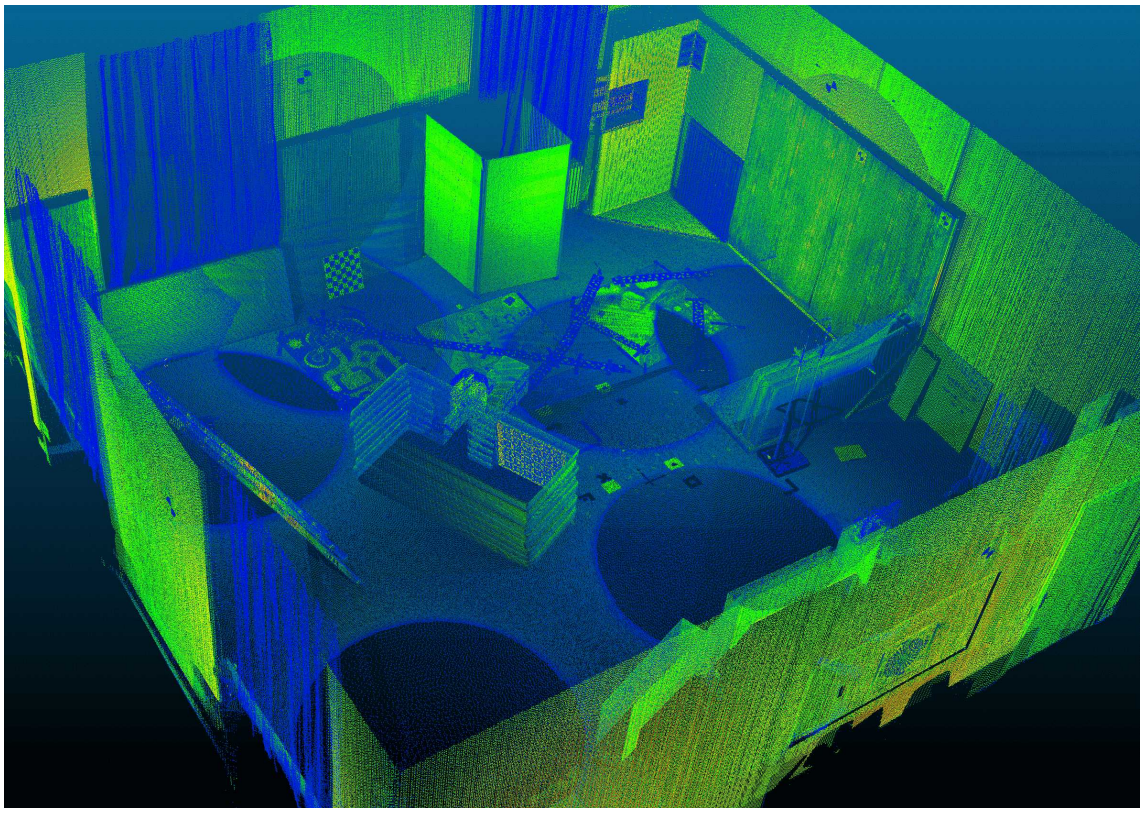


Figure 4.8: Ground-truth point cloud for dataset `V1_01_easy`.

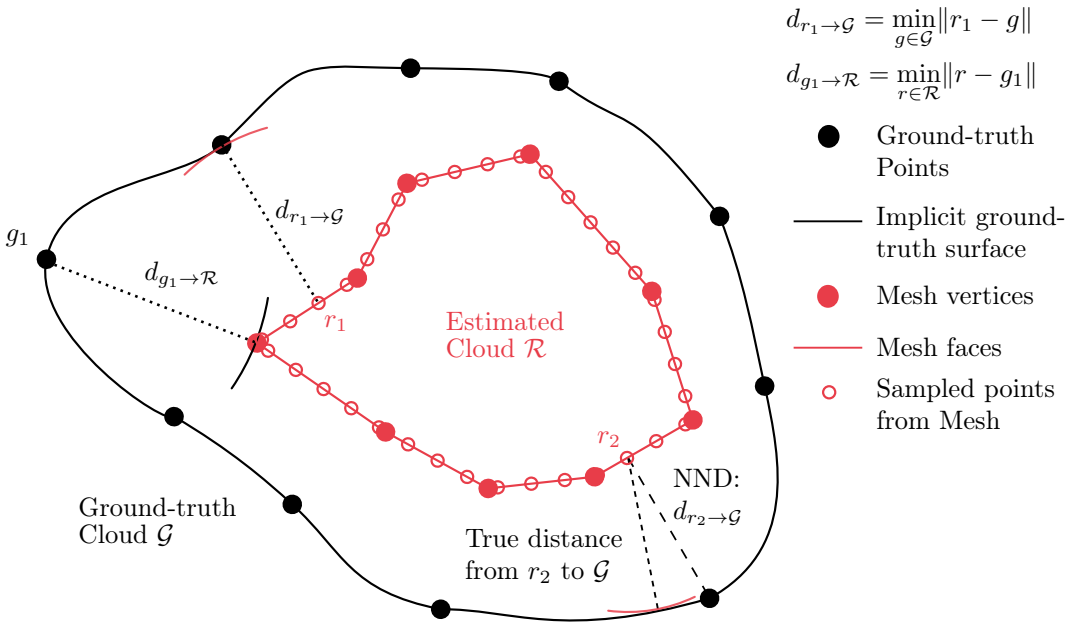


Figure 4.9: Graphical comparison of the Nearest Neighbour Distance (NND) used to compute the cloud to cloud distance ($d_{r_2 \to \mathcal{G}}$), against the true distance between a point on the mesh and the real surface (bottom-right corner). Also depicted are the distances from the sampled point cloud of the mesh to the ground-truth point cloud $d_{r_1 \to \mathcal{G}}$, used to calculate the accuracy of the mesh (eq. (4.12)), and viceversa $d_{g_1 \to \mathcal{R}}$, used to calculate the completeness of the mesh (eq. (4.13)).

$\mathcal{G}$ corresponds to:

$$d_{r\to\mathcal{G}} = \min_{g\in\mathcal{G}} \|r-g\|_2 \quad \text{for} \quad r\in\mathcal{R} \tag{4.10}$$

The nearest neighbour distance is susceptible to the density of the reference point cloud. The denser the cloud the best approximation of the accuracy error. In fig. 4.9 we can see graphically the difference between the nearest neighbour distance and the actual real distance. In our case, since the ground-truth point cloud is fairly dense, we can assume that our approximation converges to the true distance.

More specifically, we report the mean and standard deviation of eq. (4.10):

$$\bar{d}_{\mathcal{R}} = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} d_{r\to\mathcal{G}} \quad \text{and} \quad \sigma_{\mathcal{R}} = \sqrt{\frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \left|d_{r\to\mathcal{G}} - \bar{d}_{\mathcal{R}}\right|^2} \tag{4.11}$$

where $|\mathcal{R}|$ is the cardinality of $\mathcal{R}$, $\bar{d}_{\mathcal{R}}$ is the mean, and $\sigma_{\mathcal{R}}$ the standard deviation. These values give a good estimate of what is the accuracy (mean) and precision (standard deviation) of the mesh.

Nevertheless, it is more common to define the accuracy $\mathcal{A}$ as the fraction (in percentage) of estimated points which are within a distance threshold τ of the ground-truth point cloud [39, 124]:

$$\mathcal{A}(\tau) = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \Big[d_{r\to\mathcal{G}} < \tau\Big]_I \times 100 \tag{4.12}$$

where $[P]_I$ is the Iverson bracket, defined as:

$$[P]_I = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{otherwise,} \end{cases}$$

Table 4.3 reports the mean and standard deviation of the cloud to cloud distance from the mesh to the ground-truth point cloud (eq. (4.11)) for both the S + P and the S + P + R pipelines. Table 4.4 reports instead the accuracy $\mathcal{A}(\tau)$, as defined in eq. (4.12), for different τ.

As a reminder, the S + P pipeline still uses the mesh to detect regularities and transform Structureless factors to Projection factors, but, contrary to the S + P + R pipeline, it does not enforce the structural regularities on the landmarks.

Mapping Accuracy Results

Table 4.3 shows that both the mean and the standard deviation of the distance from the mesh to the ground-truth point cloud decreases when enforcing structural regularities, as done in the S + P + R pipeline. On average, each point sampled on the mesh generated by the S + P + R pipeline is 0.5 cm closer to the ground-truth point cloud than the points sampled on the mesh generated by the S + P pipeline. Therefore, enforcing structural regularities makes the estimated mesh closer to the real scene. In terms of accuracy values $\mathcal{A}(\tau)$, table 4.4 shows that the S + P + R pipeline consistently achieves more accurate mesh estimates (between 3%-7% better) for distance thresholds $\tau < 10cm$. Figures 4.10 and 4.11 show the actual error distributions for $d_{r\to\mathcal{G}}$, and the mesh accuracy $\mathcal{A}(\tau)$ for distance thresholds τ going from 0 to 0.3 meters, for both the S + P and the S + P + R pipelines respectively.

Note that, the results in absolute values are not near the quality of dense 3D reconstruction algorithms such as COLMAP [126], that easily achieves more than 80% in accuracy and 50% in terms of completeness for thresholds as small as $2cm$ [124]. Indeed, we do not pretend to compete against offline dense 3D reconstruction algorithms, but rather sparse keypoint-based visual pipelines. Compared with the sparse point cloud generated by these visual pipelines, our mesh provides a denser estimate of the scene.

We can also visualize the improvement achieved when enforcing structural regularities by color-encoding the estimated point cloud with the errors. Figure 4.14 shows the ground-truth point cloud (a), next to the estimated point cloud (b), from the same point of view. The error distances $d_{r\to\mathcal{G}}$ for each point are color-encoded on the estimated point cloud, while the colors on the ground-truth point cloud are only set for visualization purposes. We can observe that significant errors are actually present on the planar surfaces, especially on the walls, when we do not enforce structural regularities. A closer view on the wall itself in fig. 4.20 shows that it is visually clear that adding co-planarity constraints results in smoother walls.

Figure 4.15 shows instead that the estimated point cloud has significantly less errors when structural regularities are enforced (b).

Finally, on fig. 4.16 we compare the point cloud estimated by the S + P pipeline against the point cloud estimated by the S + P + R pipeline. We can observe that planar surfaces such as the walls and the floor have significantly less errors for the mesh generated by the S + P + R pipeline than for the mesh of the S + P pipeline. On figs. 4.17 to 4.19, we compare the same point clouds from another viewpoint.

Table 4.3: Statistics for the cloud to cloud absolute distance from the mesh to the ground-truth point cloud $d_{r\to\mathcal{G}}$, as defined in eq. (4.11). Results are reported for the pipeline using Structureless and Projection factors (S + P), and our proposed approach using Regularity factors (S + P + R), for dataset `V1_01_easy`.

	VIO Type	
$d_{r\to\mathcal{G}}$ statistics (eq. (4.11))	**S + P**	**S + P + R** (Proposed)
Mean $\bar{d}_{\mathcal{R}}$ [cm]	4.9	**4.4**
Standard Deviation $\sigma_{\mathcal{R}}$ [cm]	5.0	**4.6**

Table 4.4: Mesh accuracy $\mathcal{A}(\tau)$ as defined in eq. (4.12) for pipeline using Structureless and Projection factors (S + P), and our proposed approach using Regularity factors (S + P + R), for dataset `V1_01_easy`. The distance threshold τ is set to different values to better assess the accuracy of the mesh.

	Mesh accuracy $\mathcal{A}(\tau)$ [%] (eq. (4.12))	
Distance threshold τ [cm]	**S + P**	**S + P + R** (Proposed)
$\tau = 1$	14	**17**
$\tau = 4$	57	**64**
$\tau = 10$	90	90

4.3.3 Map Completeness

Similarly to the accuracy, we define the *completeness* of the mesh as the percentage of points within completeness

a threshold of the ground truth:

$$\mathcal{C}(\tau) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \Big[d_{g \to \mathcal{R}} < \tau \Big]_I \times 100 \tag{4.13}$$

where $[P]_I$ is the Iverson bracket. The completeness explains to what extent all the ground-truth points are covered.

Map Completeness Results

Table 4.5 shows that, in terms of completeness, both S + P and S + P + R pipelines achieve similar results. On figs. 4.12 and 4.13, we can see that the distributions of the distance errors $d_{g \to \mathcal{R}}$ are similar for both pipelines; leading to similar completeness $\mathcal{C}(\tau)$ values for distance thresholds τ going from 0 to 0.3 meters. Figure 4.21 color-encodes each ground-truth point with its corresponding distance to the sampled point cloud from the mesh ($d_{g \to \mathcal{R}}$) for both the S + P and the S + P + R pipelines; we can see that the walls have a slightly better completeness for the S + P + R mesh, but there is no significant difference against the S + P mesh.

As expected, enforcing structural regularities does not necessarily improve the completeness metric since the amount of faces of the mesh used by both pipelines is similar. Both pipelines extract and use the same keypoints from the image, therefore they work both with a similar amount of 3D landmarks, and implicitly the same number of mesh faces. Of course, due to changes in accuracy caused by the regularity factors, it can happen that some faces of the mesh are filtered out when propagating the mesh from 2D to 3D, as explained in § 3.2.1. Nevertheless, the filtering of mesh faces is not substantial and does not lead to significant differences on the completeness of the mesh.

Table 4.5: Mesh completeness $\mathcal{C}(\tau)$ as defined in eq. (4.13) for pipeline using Structureless and Projection factors (S + P), and our proposed approach using Regularity factors (S + P + R), for dataset `V1_01_easy`. The distance threshold τ is set to different values to better assess the level of completeness achieved.

	Mesh completeness $\mathcal{C}(\tau)$ [%] (eq. (4.13))	
Distance threshold τ [cm]	**S + P**	**S + P + R** (Proposed)
$\tau = 1$	**18**	17
$\tau = 4$	52	**53**
$\tau = 10$	72	**74**

Note that the trajectory of the camera in the EuRoC dataset is not intended for a complete 3D reconstruction (as it only observes a subset of the scene); assessing the completeness using this dataset is therefore misleading. To avoid the issue of having parts of the scene unobserved, we explicitly ignore the points in the ground-truth point cloud which have a completeness value higher than $0.3m$, since these points are most likely unobserved.

4.3.4 F-score

As noted by [126], precision alone can be maximized by creating multiple copies of a sparse set of precisely localized landmarks, while completeness can be maximized by uniformly covering the scene with points. To rank dense reconstructions it is therefore common to use the *F-score* [39,126]:

F-score

$$\mathcal{F}(\tau) = \frac{2\mathcal{A}(\tau)\mathcal{C}(\tau)}{\mathcal{A}(\tau) + \mathcal{C}(\tau)}. \tag{4.14}$$

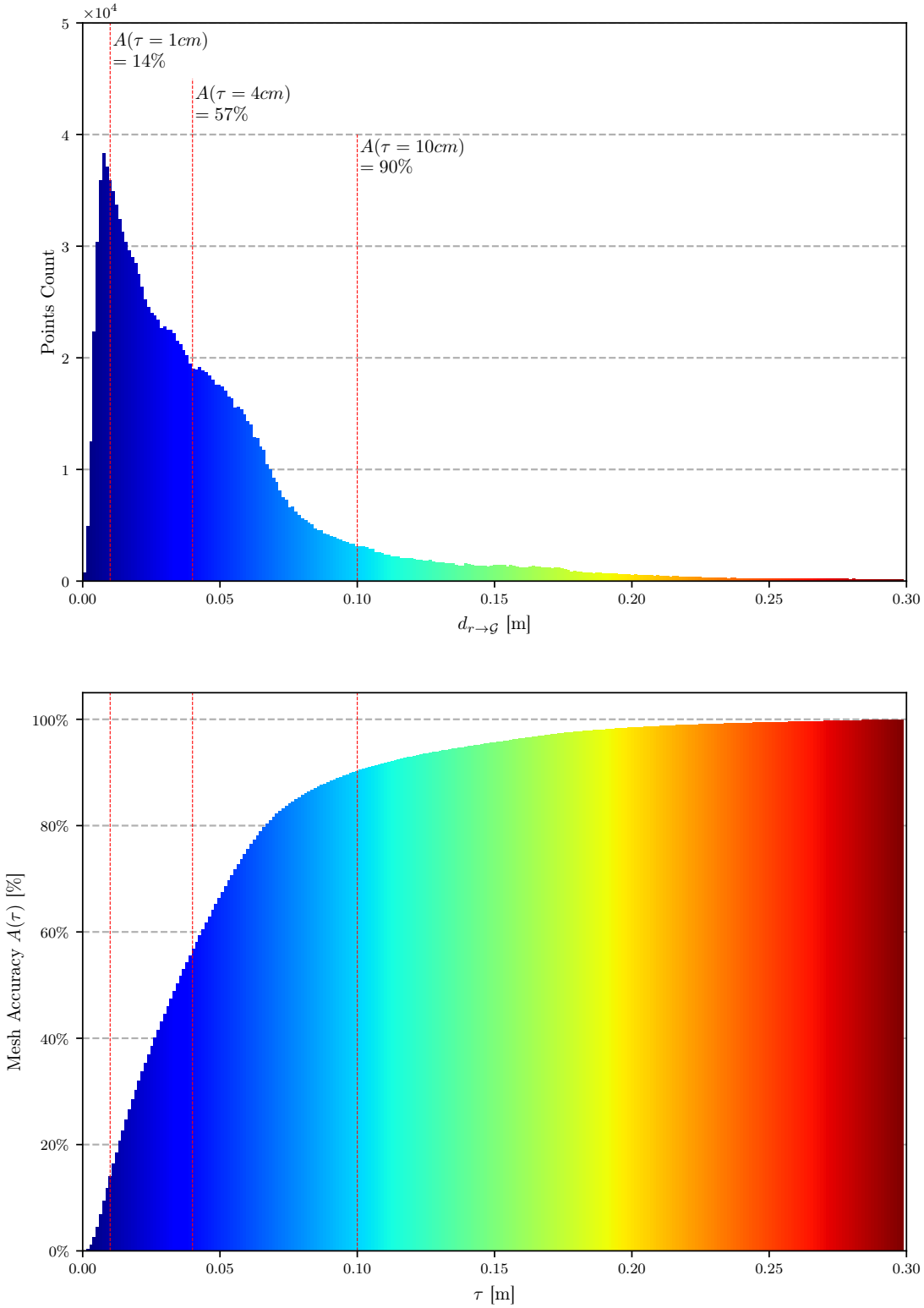


Figure 4.10: Dataset `V1_01_easy`, S + P pipeline. Top figure: Histogram of points sampled on the mesh depending on their distance to the ground-truth point cloud ($d_{r\to\mathcal{G}}$). Bottom figure: Accuracy $\mathcal{A}(\tau)$, corresponding to the cumulative histogram of $d_{r\to\mathcal{G}}$. Colormap matches the one used in figs. 4.14 to 4.19 for better visualization.

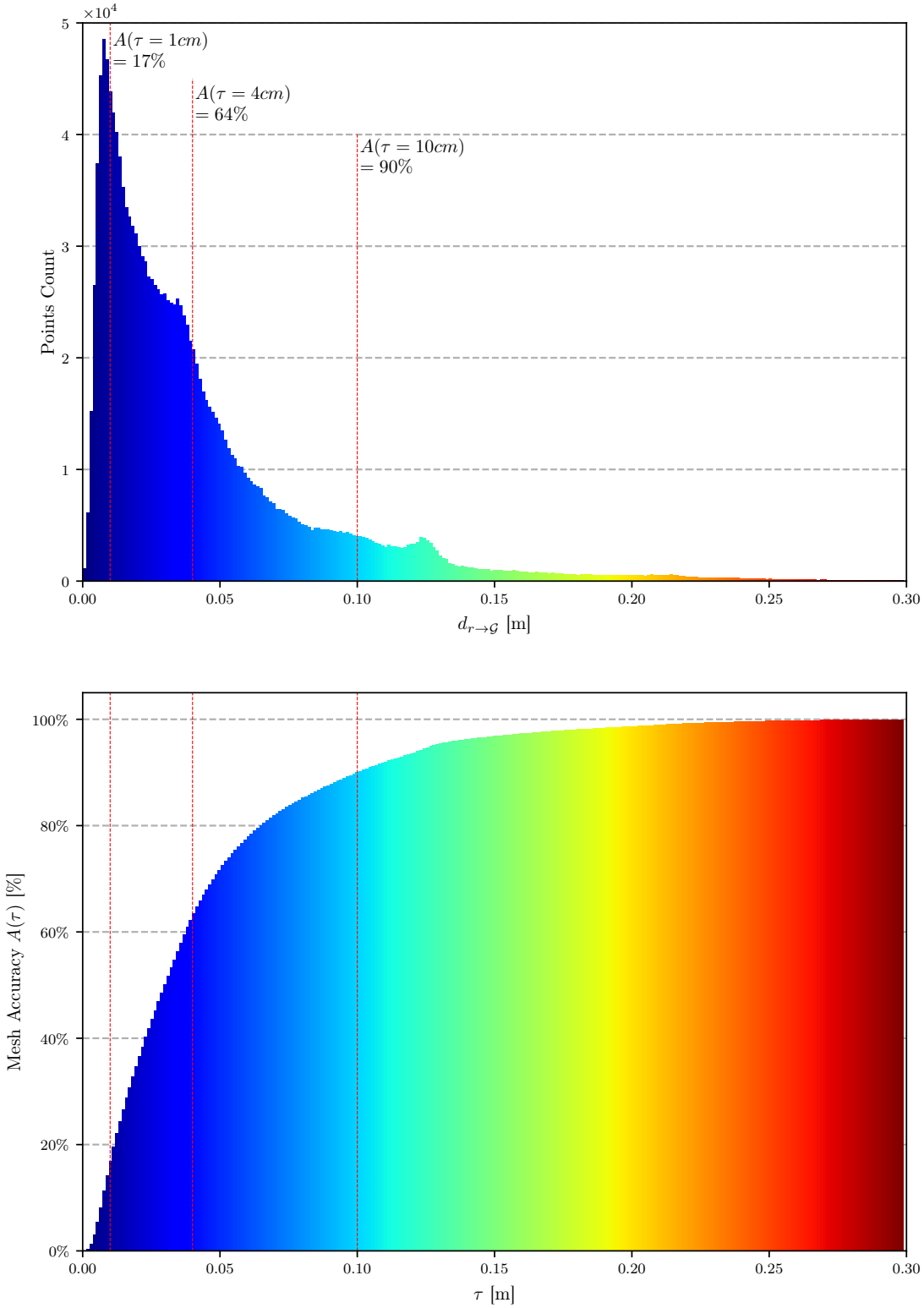


Figure 4.11: Dataset `V1_01_easy`, S + P + R pipeline. Top figure: Histogram of points sampled on the mesh depending on their distance to the ground-truth point cloud ($d_{r\to\mathcal{G}}$). Bottom figure: Accuracy $\mathcal{A}(\tau)$, corresponding to the cumulative histogram of $d_{r\to\mathcal{G}}$. Colormap matches the one used in figs. 4.14 to 4.19 for better visualization.

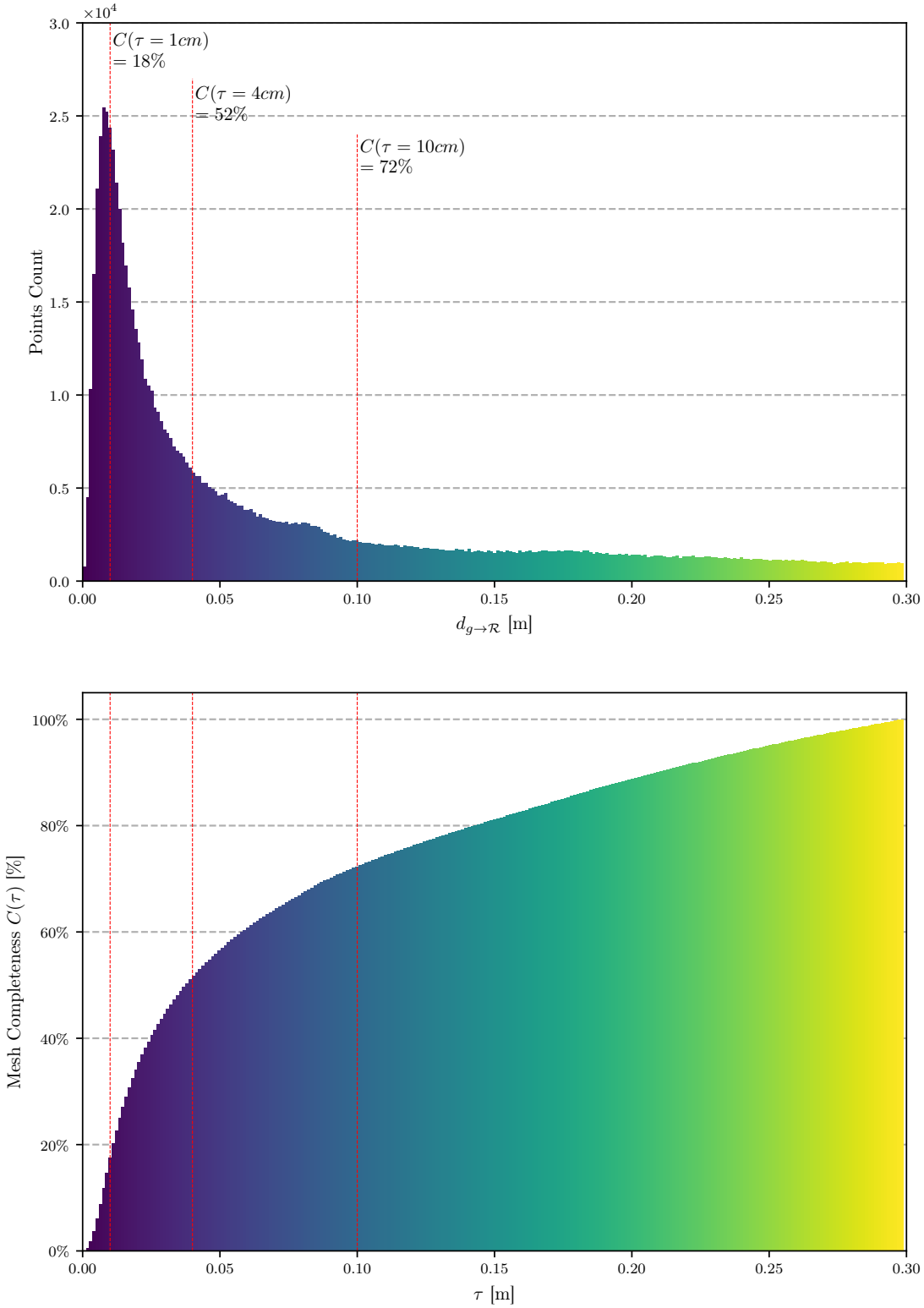


Figure 4.12: Dataset `V1_01_easy`, S + P pipeline. Top figure: Histogram of ground-truth points depending on their distance to the point cloud sampled from the mesh ($d_{g\to\mathcal{R}}$). Bottom figure: Completeness $\mathcal{C}(\tau)$, corresponding to the cumulative histogram of $d_{g\to\mathcal{R}}$. Colormap matches the one used in fig. 4.21 for better visualization.

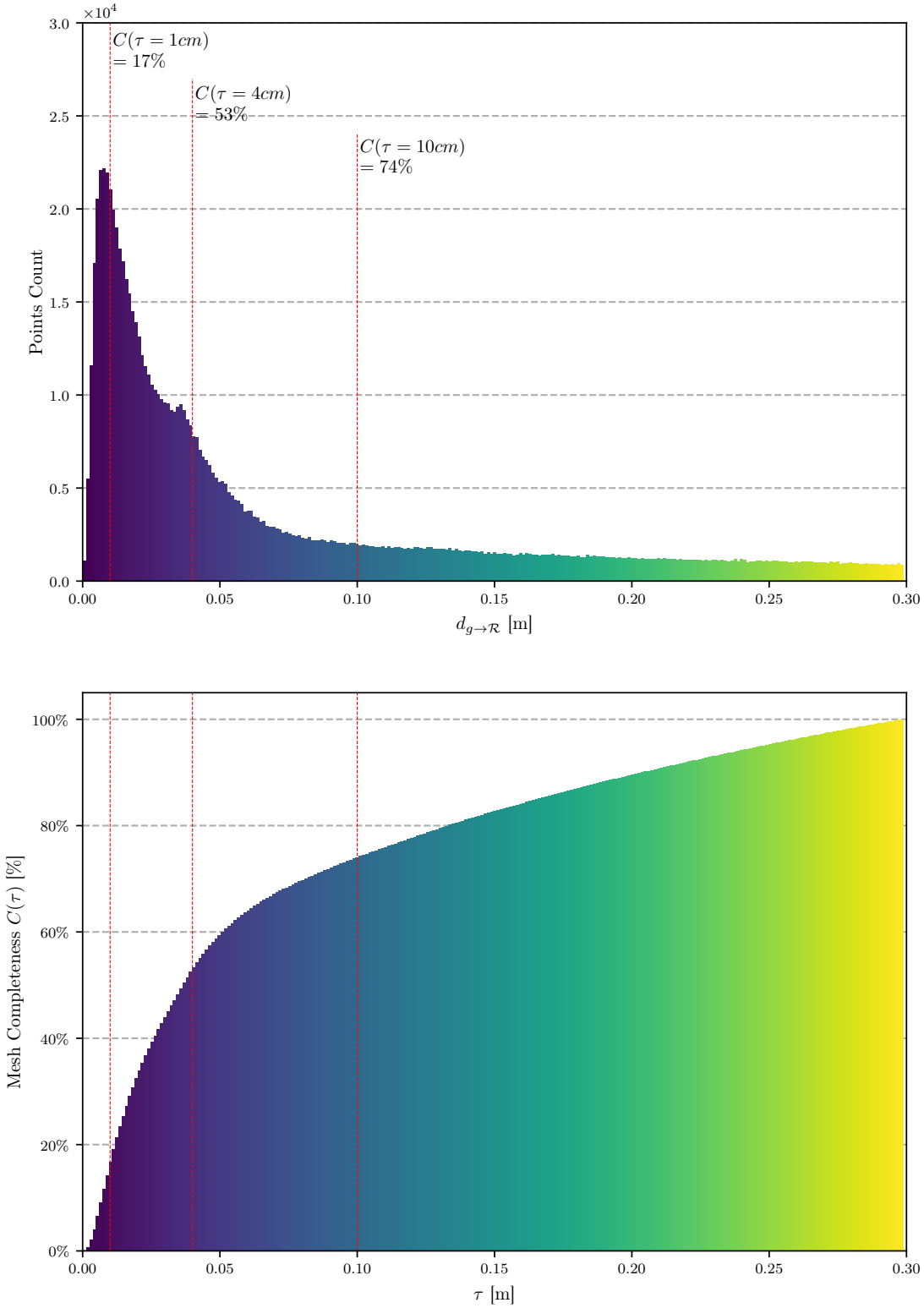


Figure 4.13: Dataset `V1_01_easy`, S + P + R pipeline. Top figure: Histogram of ground-truth points depending on their distance to the point cloud sampled from the mesh ($d_{g\to\mathcal{R}}$). Bottom figure: Completeness $\mathcal{C}(\tau)$, corresponding to the cumulative histogram of $d_{g\to\mathcal{R}}$. Colormap matches the one used in fig. 4.21 for better visualization.

Contrary to the arithmetic mean, the F-score has the property that if either $\mathcal{A}(\tau) \to 0$ or $\mathcal{C}(\tau) \to 0$, then $\mathcal{F}(\tau) \to 0$, and in this sense it is a more suitable summary measure than the mean.

F-score results

It is clear from table 4.6 that the mesh estimated by our proposed approach, using structural regularities (S + P + R), consistently outperforms the mesh generated by the S + P pipeline. In this respect, we can conclude that using structural regularities on the optimization problem improves the overall quality of the mesh.

Table 4.6: Mesh F-score $\mathcal{F}(\tau)$ as defined in eq. (4.14) for pipeline using Structureless and Projection factors (S + P), and our proposed approach using Regularity factors (S + P + R), for dataset `V1_01_easy`. The distance threshold τ is set to different values to better assess the F-factor achieved by each pipeline.

	Mesh F-score $\mathcal{F}(\tau)$ [%] (eq. (4.14))	
Distance threshold τ [cm]	**S + P**	**S + P + R** (Proposed)
$\tau = 1$	15.7	**17.0**
$\tau = 5$	54.4	**58.0**
$\tau = 10$	80.0	**81.2**

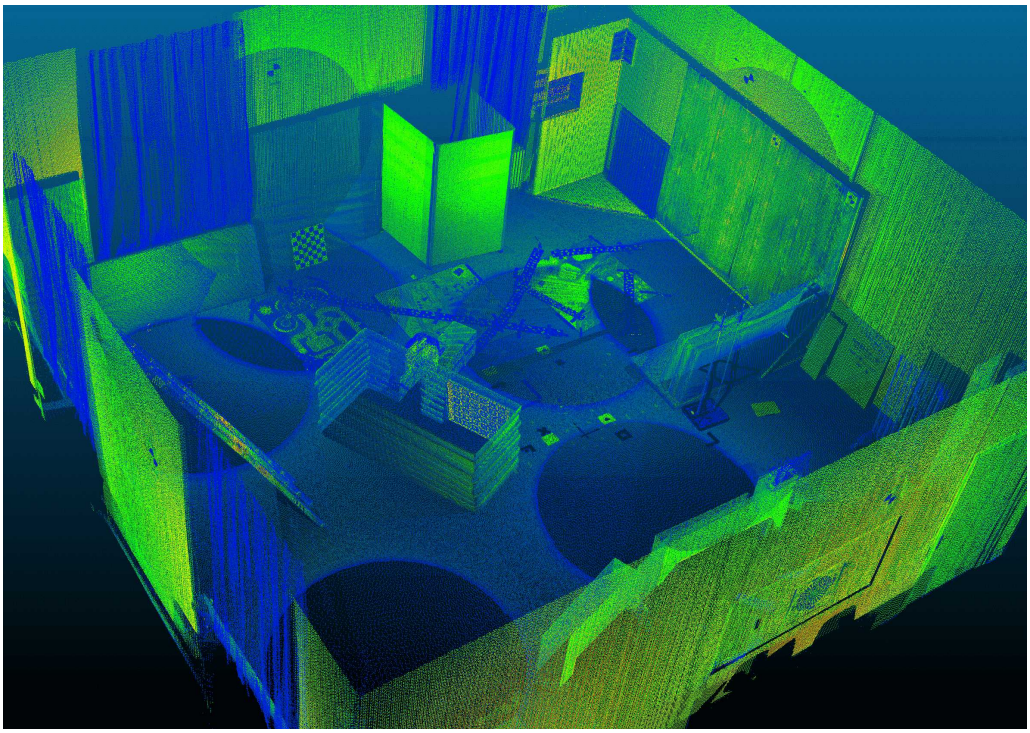
4.4 Timing

The different pipelines covered in this thesis differ in that they try to solve an increasingly complicated problem. While the Structureless pipeline (S) does not include neither the 3D landmarks nor the planes as variables in the optimization problem, the pipeline using Structureless and Projection factors (S + P) includes 3D landmarks, and the pipeline using regularities (S + P + R) further includes planes as variables. Moreover, the S + P has significantly less constraints between the variables than the S + P + R pipeline. We refer the reader to the concise book of F. Dellaert and M. Kaess [14] for an explanation on how the optimization time depends on the number of variables involved and the number of constraints. For the scope of this thesis, it is sufficitent to know that the larger the number of variables and constraints between them, the slower it takes to solve the optimization problem.

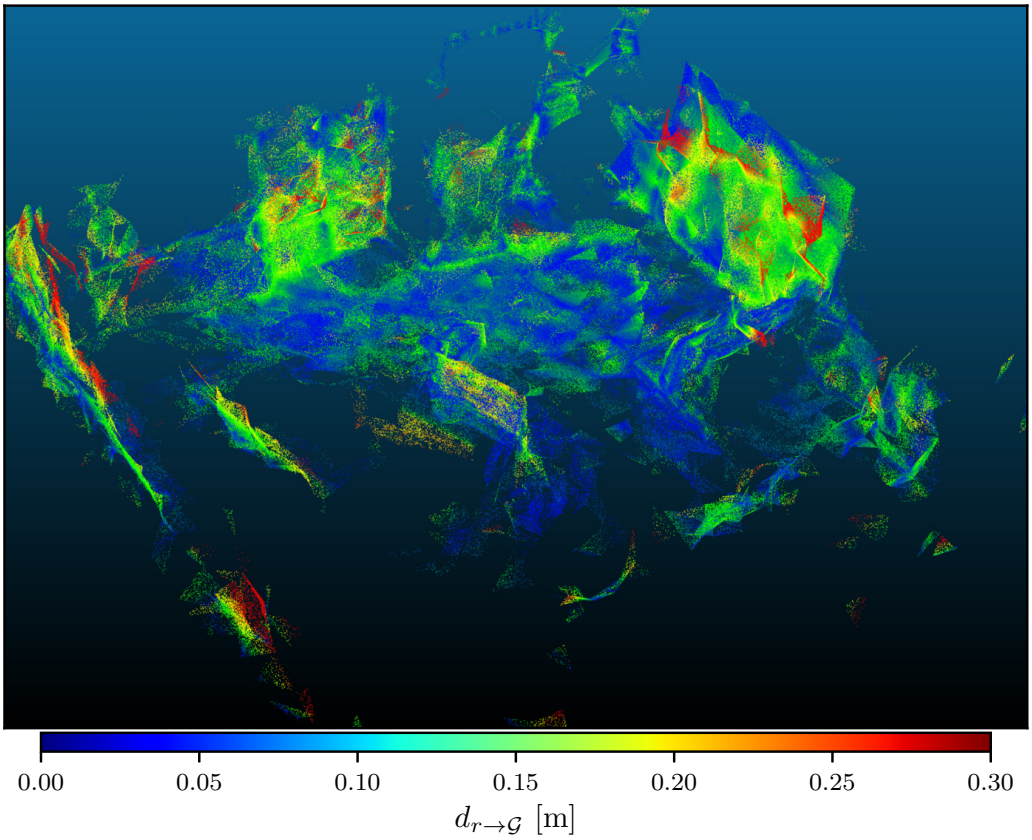
Hence, we can expect that the optimization times for the different pipelines will be each bounded by the other as $t_S^{opt} < t_{S+P}^{opt} < t_{S+P+R}^{opt}$, where t_X^{opt} is the time taken to solve the optimization problem of pipeline X.

4.4.1 Experimental Results

Experimentally, we observe that the optimization time follows indeed the expected distribution $t_S^{opt} < t_{S+P}^{opt} < t_{S+P+R}^{opt}$. Figure 4.22 shows the time taken to solve the optimization problem for each type of pipeline. We can also see that, while the S + P pipeline takes slightly longer than the S pipeline per keyframe (an average of 11ms longer, with a maximum difference of 138ms), the S + P + R seems to show larger delays with respect to the S + P pipeline (an average of 28ms longer, with a maximum of 262ms). This larger delay between S + P and S + P + R than between the S and S + P pipelines can be explained by the considerable increase of constraints added when

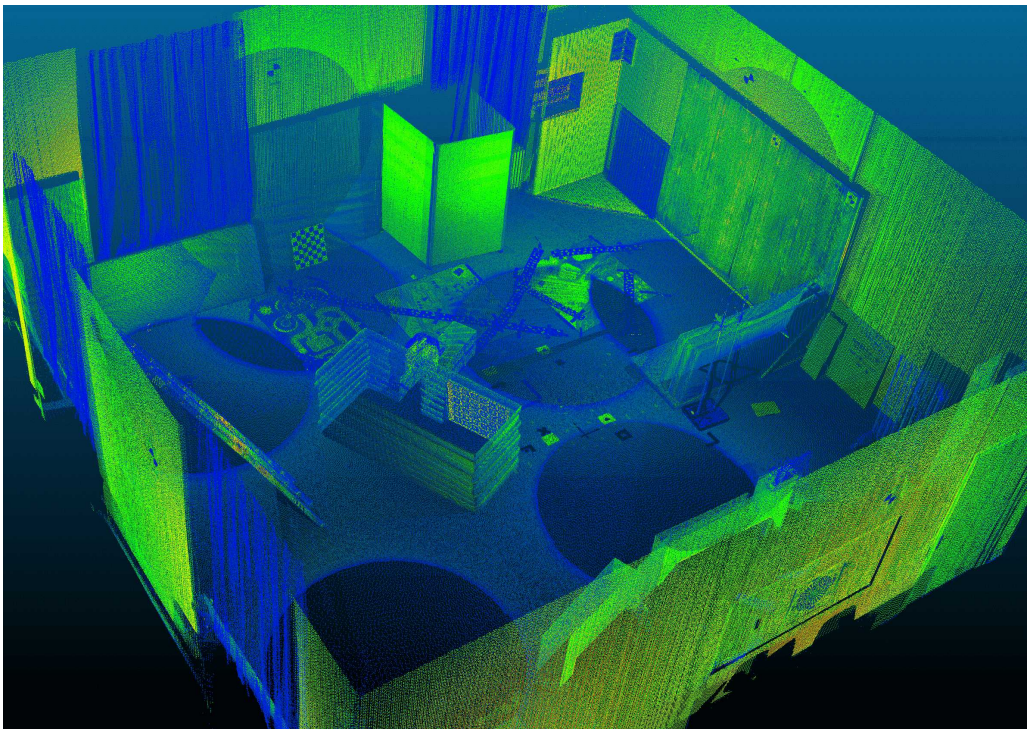


(a) Ground-truth point cloud. Color-encoded for better visualization.

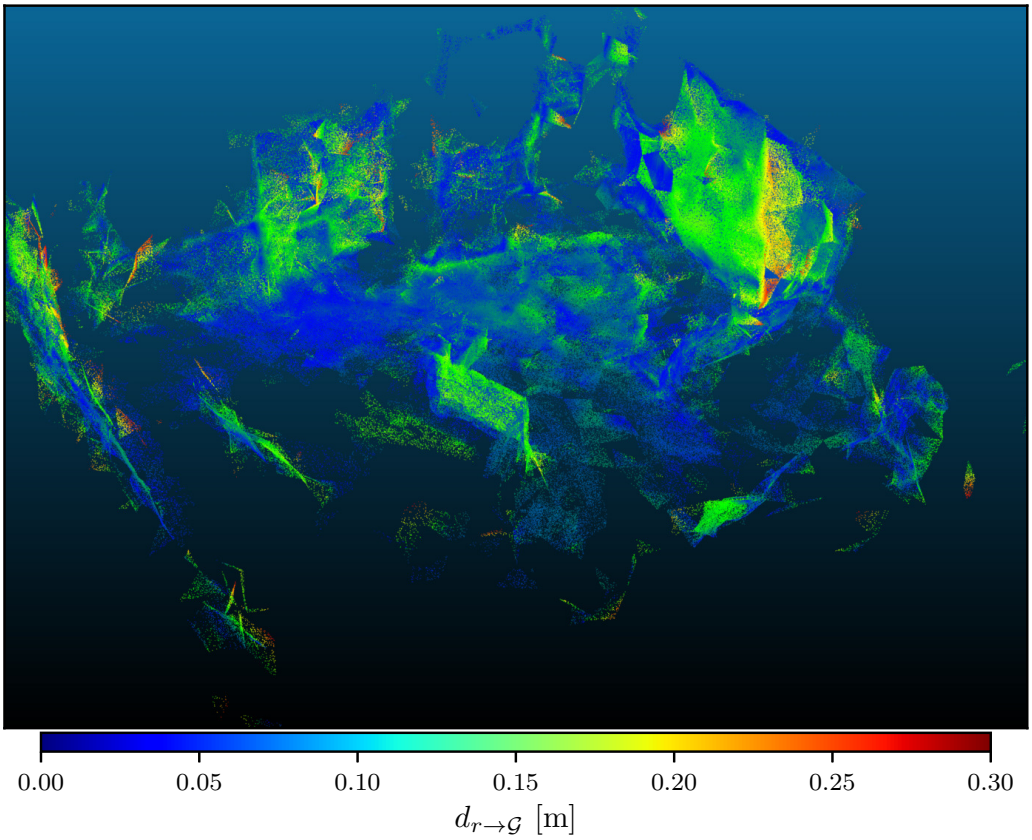


(b) Sampled mesh with color-encoded cloud to cloud distances against ground-truth point cloud.

Figure 4.14: Ground-truth point cloud and estimated point cloud (sampled from the mesh) for pipeline S + P and dataset `V1_01_easy`. Viewpoint is the same for both images.

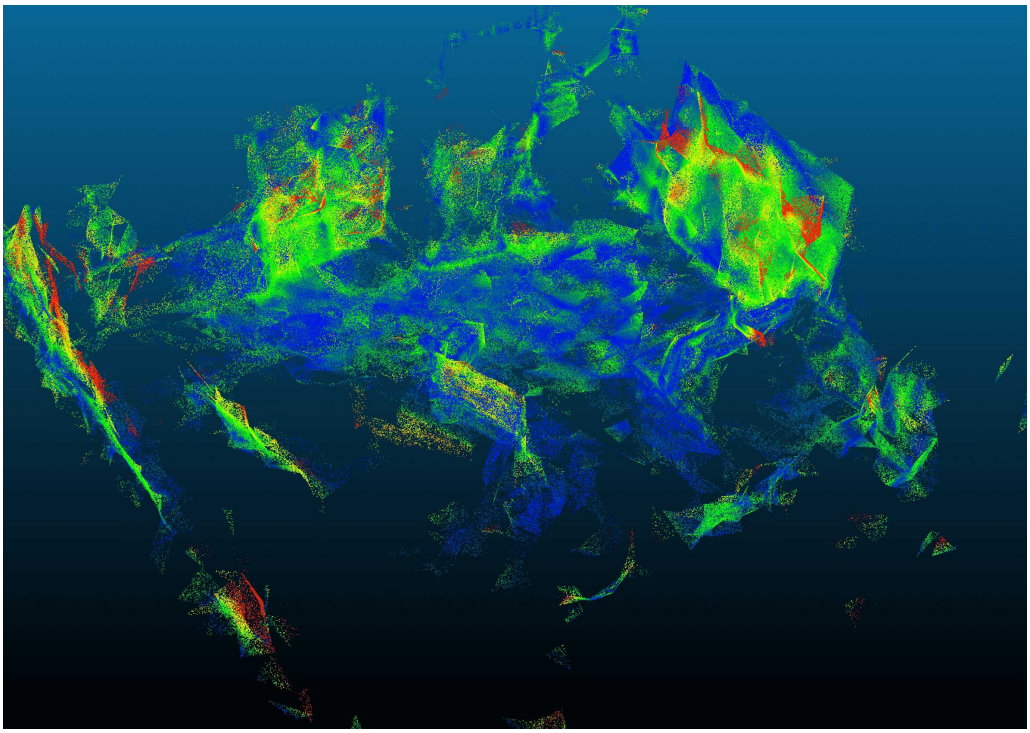


(a) Ground-truth point cloud. Color-encoded for better visualization.

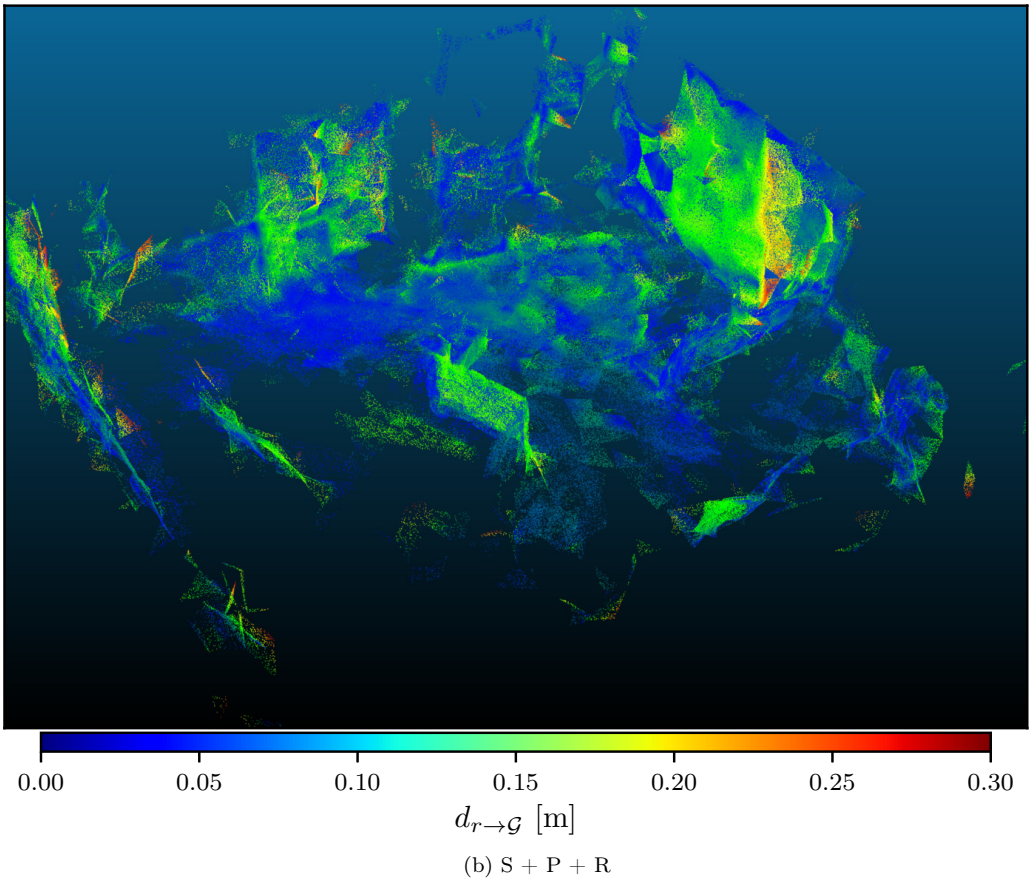


(b) Sampled mesh with color-encoded cloud to cloud distances against ground-truth point cloud.

Figure 4.15: Ground-truth point cloud and estimated point cloud (sampled from the mesh) for pipeline S + P + R and dataset `V1_01_easy`. Viewpoint is the same for both images.

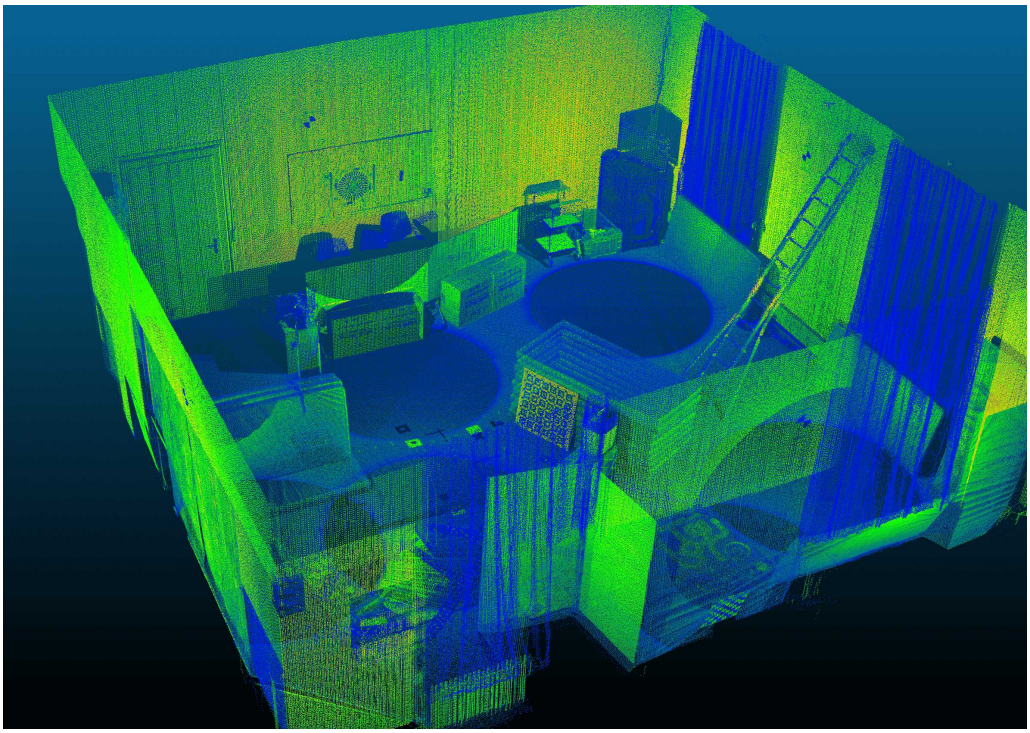


(a) S + P

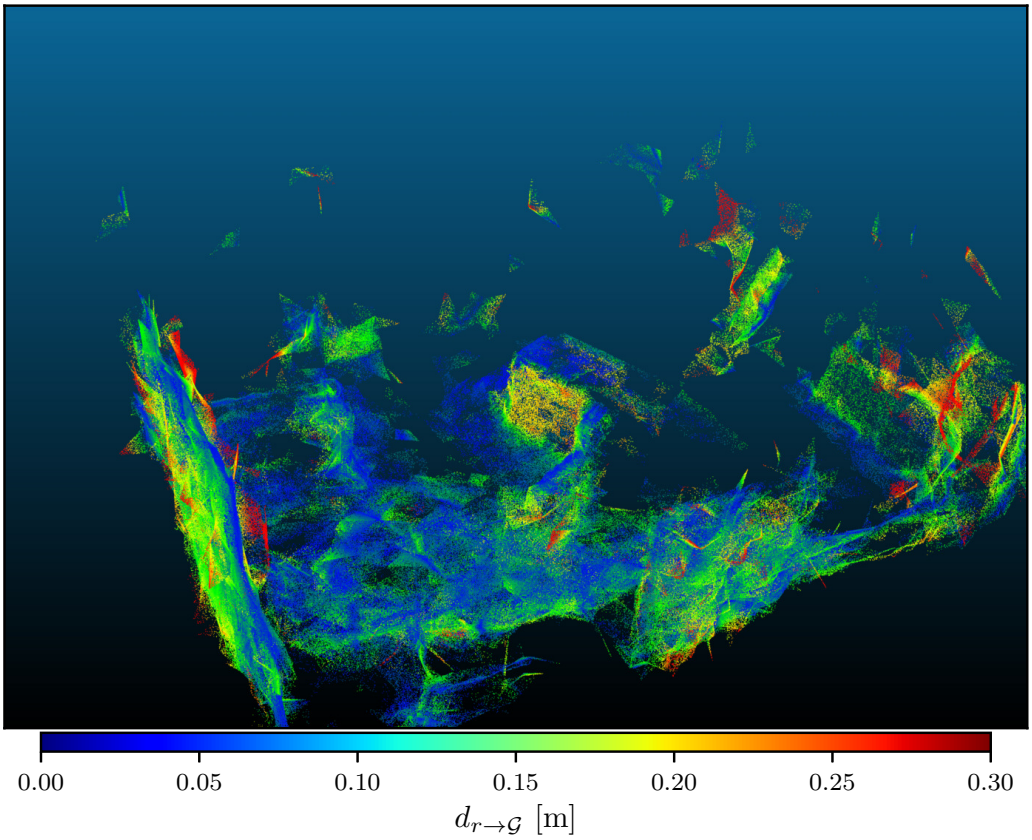


(b) S + P + R

Figure 4.16: Estimated point cloud (sampled from the mesh) for pipeline S + P and S + P + R color-encoded with cloud to cloud errors with respect to ground-truth point cloud for dataset `V1_01_easy`. Viewpoint is the same for both images.

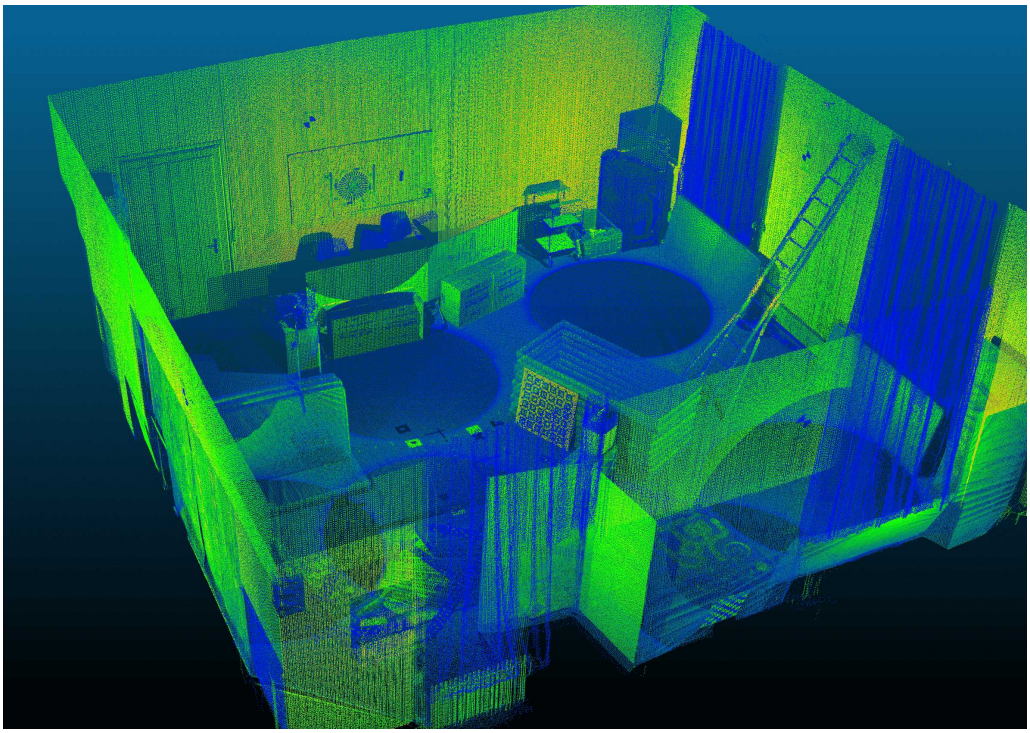


(a) Ground-truth point cloud. Color-encoded for better visualization.

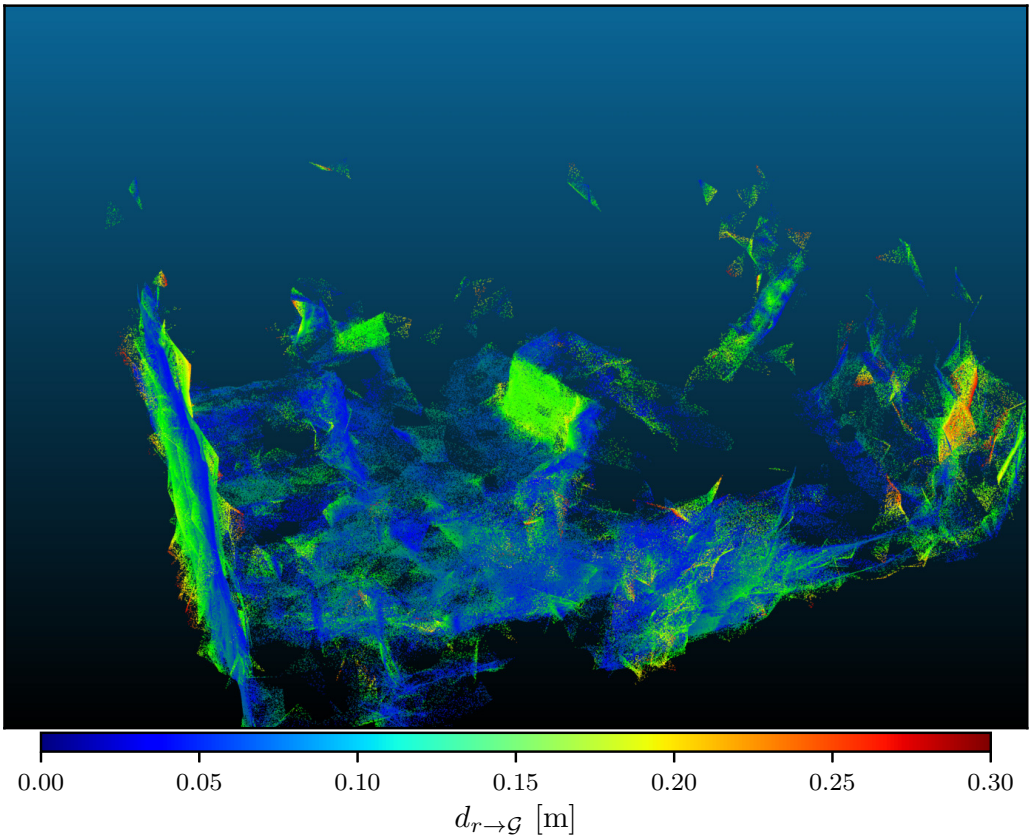


(b) Sampled mesh with color-encoded cloud to cloud distances against ground-truth point cloud.

Figure 4.17: Ground-truth point cloud and estimated point cloud (sampled from the mesh) for pipeline S + P and dataset `V1_01_easy`. Viewpoint is the same for both images.

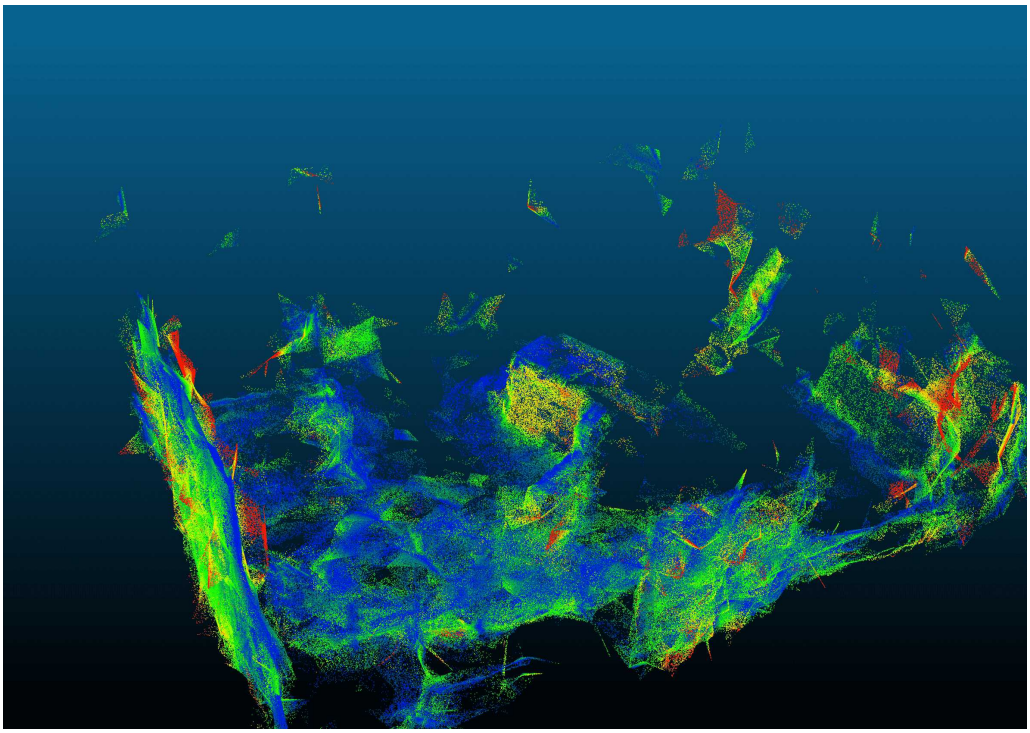


(a) Ground-truth point cloud. Color-encoded for better visualization.

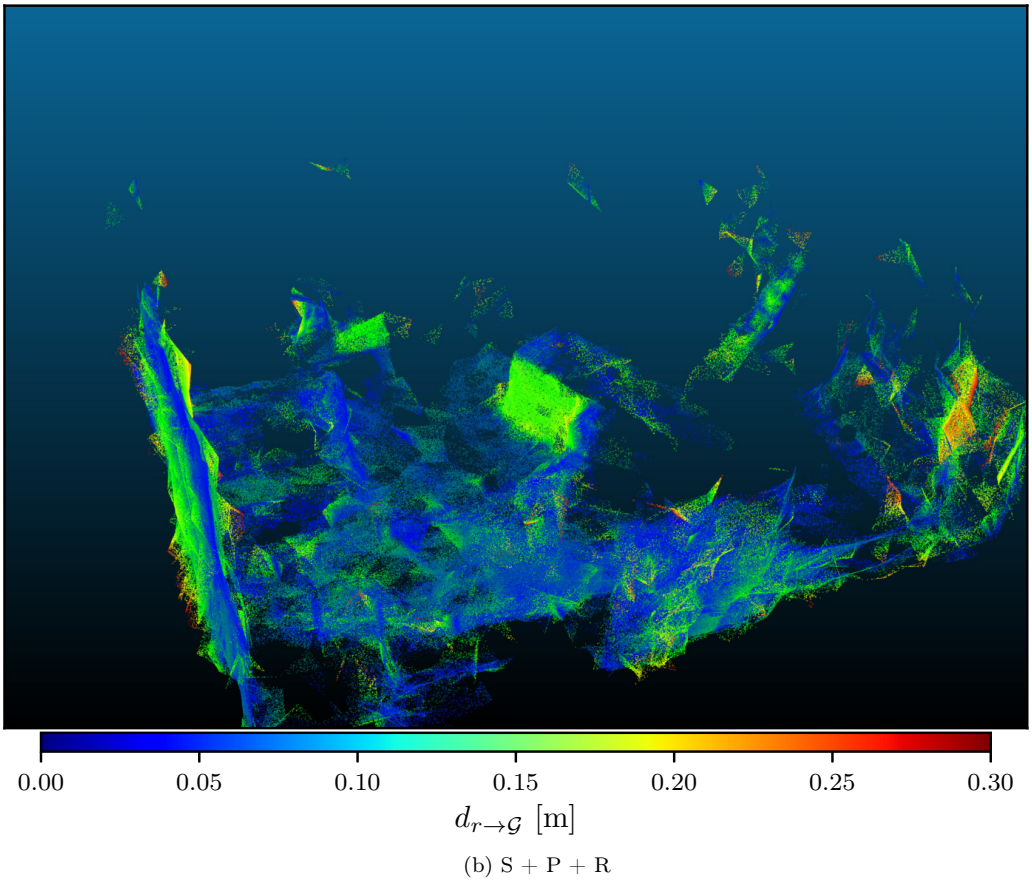


(b) Sampled mesh with color-encoded cloud to cloud distances against ground-truth point cloud.

Figure 4.18: Ground-truth point cloud and estimated point cloud (sampled from the mesh) for pipeline S + P + R and dataset `V1_01_easy`. Viewpoint is the same for both images.

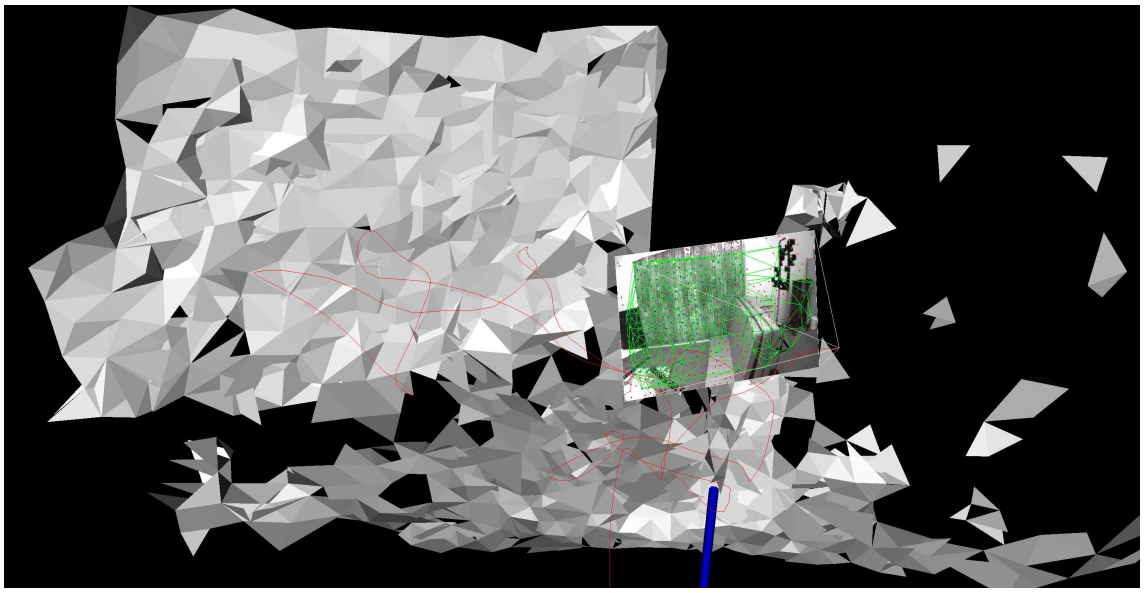


(a) S + P

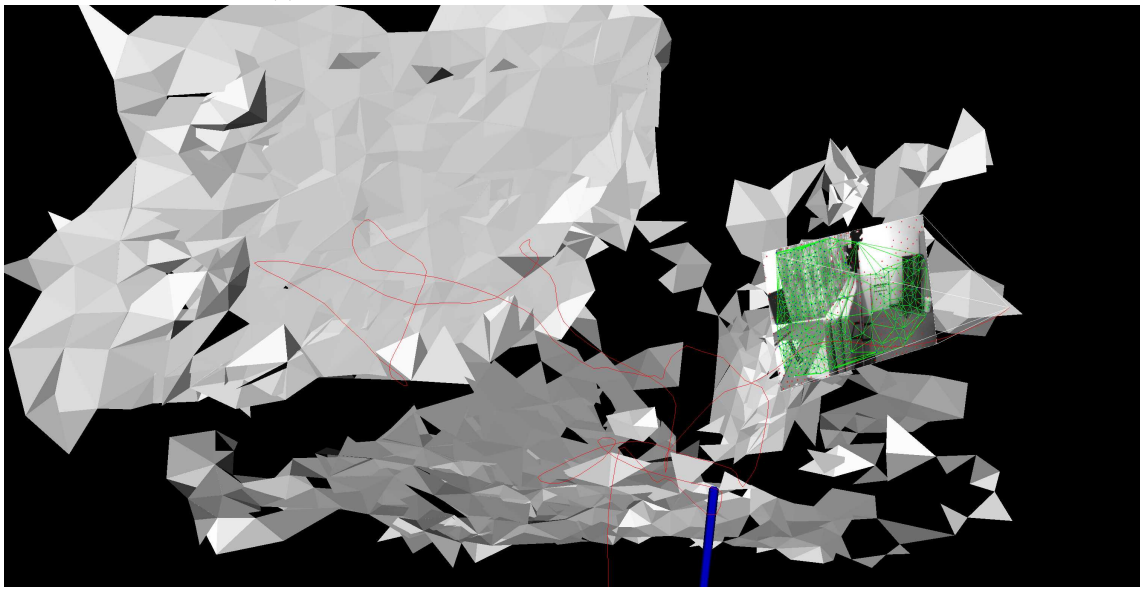


(b) S + P + R

Figure 4.19: Estimated point cloud (sampled from the mesh) for pipeline S + P and S + P + R color-encoded with cloud to cloud errors with respect to ground-truth point cloud for dataset `V1_01_easy`.

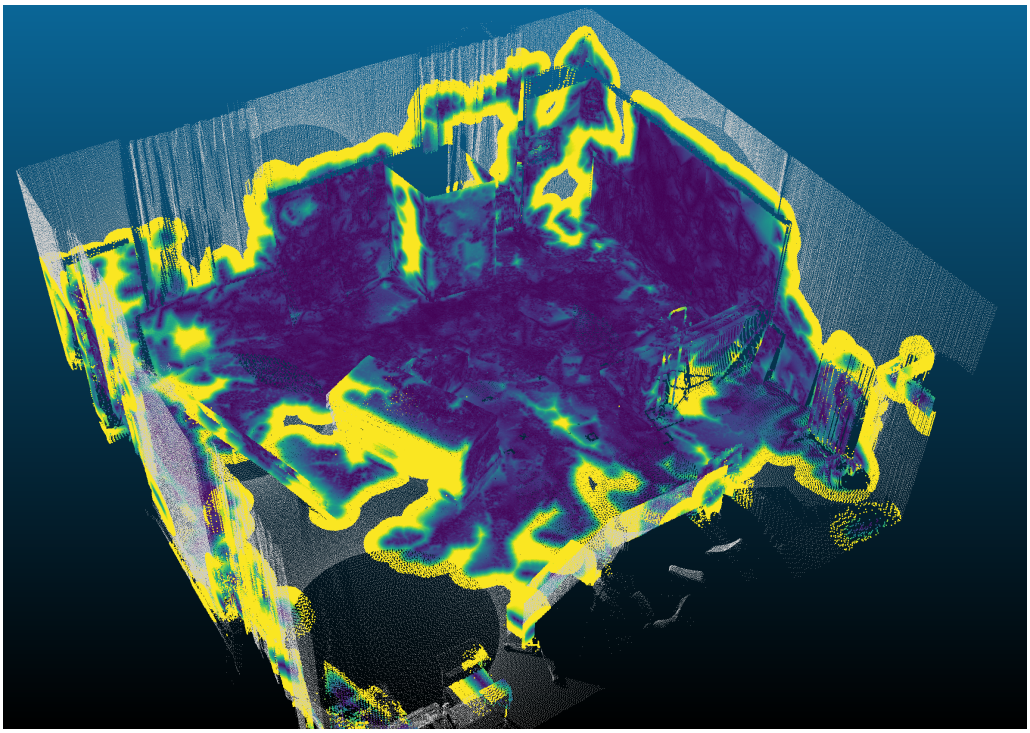


(a) Mesh estimate without enforcing co-planarity constraints.

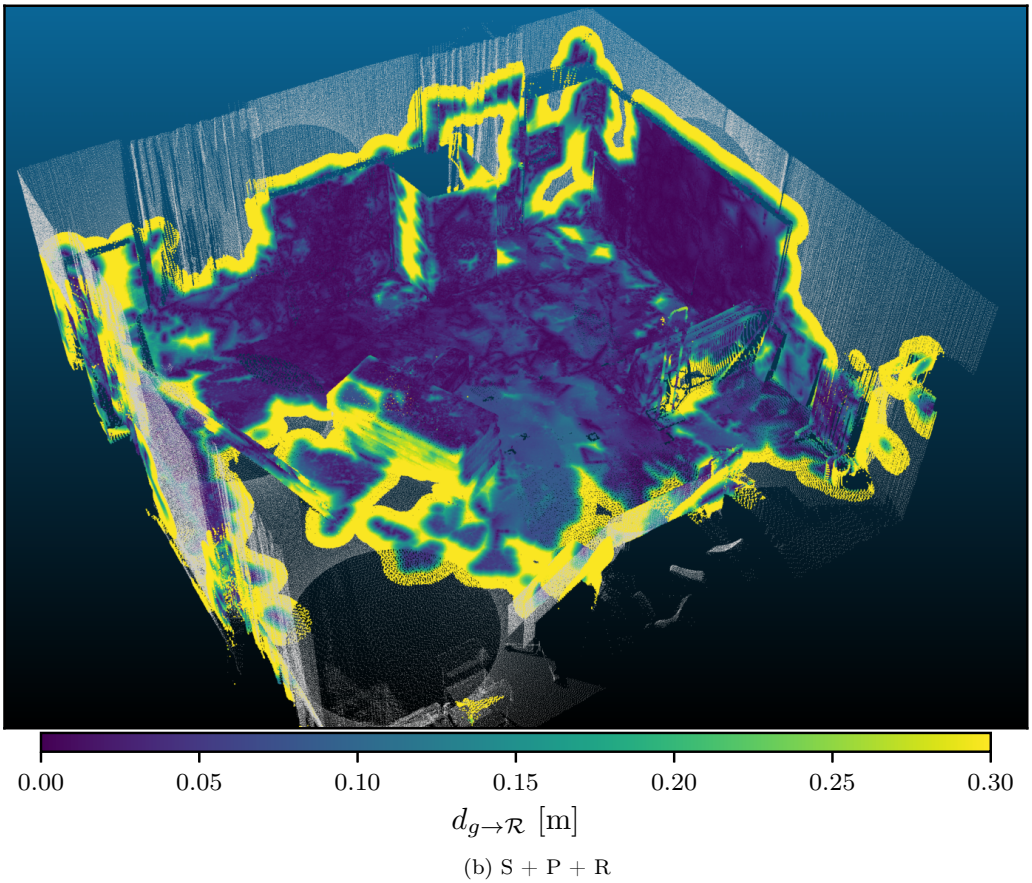


(b) Mesh estimate when enforcing co-planarity constraints.

Figure 4.20: Visual comparison of the mesh with and without co-planarity constraints enforced.



(a) S + P

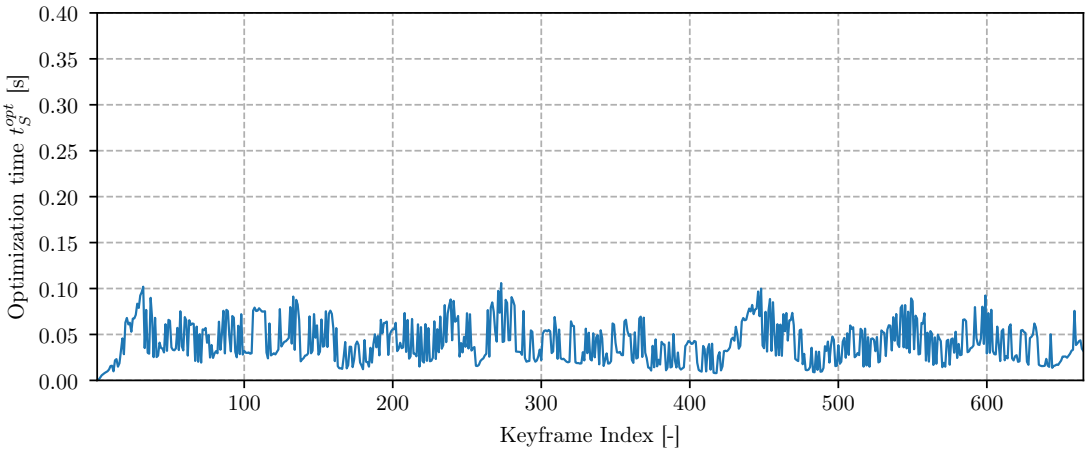


(b) S + P + R

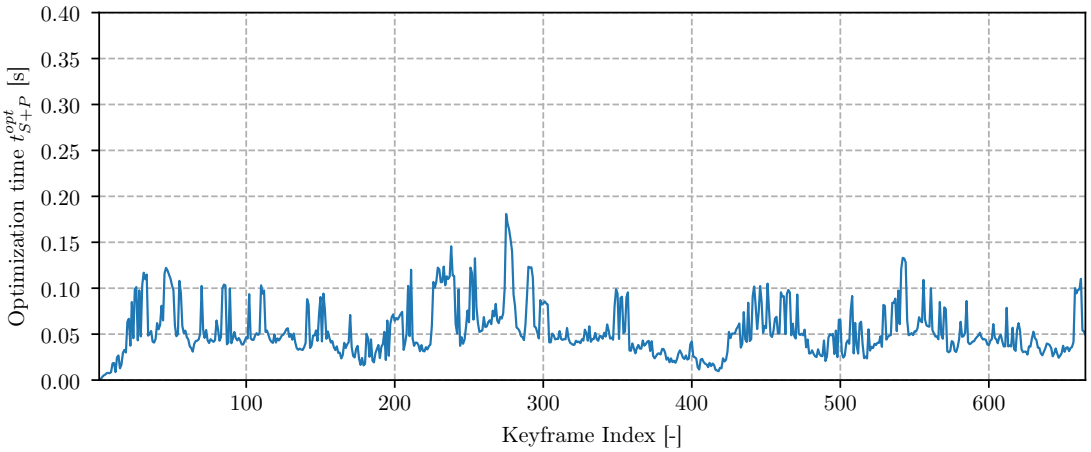
Figure 4.21: Estimated point cloud (sampled from the mesh) for pipeline S + P and S + P + R color-encoded with cloud to cloud errors with respect to ground-truth point cloud for dataset `V1_01_easy`. Viewpoint is the same for both images.

enforcing regularities. Indeed, each time a plane is added it is attached to potentially hundreds of landmarks in the factor graph, therefore increasing significantly the number of constraints.

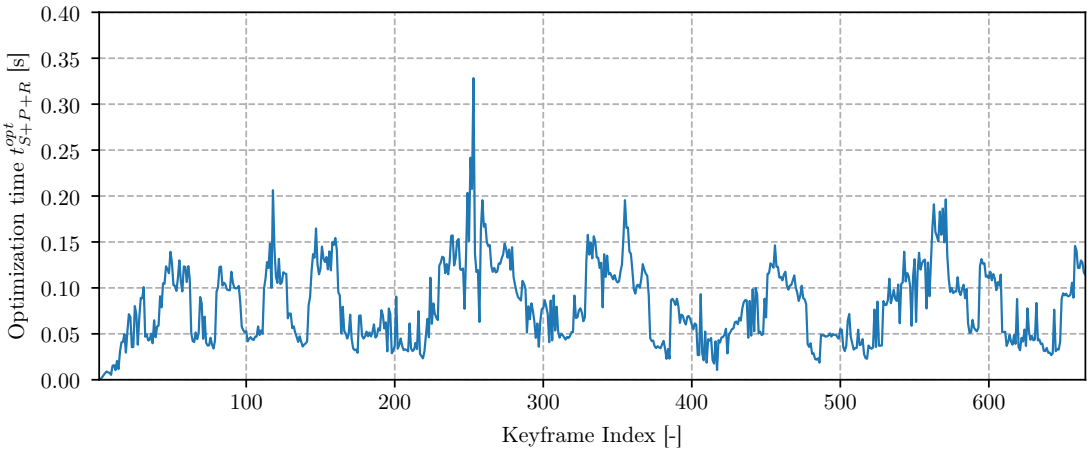
Similarly to the failure modes of landmark-based SLAM methods, which become intractable when too many landmarks are present in the optimization at any time, we observe that if the number of plane variables is large ($\sim 10^1$), and consequently the number of constraints between landmarks and planes also gets large ($\sim 10^2$), the optimization problem cannot be solved in real-time.



(a) Structureless factors. Dataset `V1_01_easy`



(b) Structureless and Projection factors. Dataset `V1_01_easy`



(c) Structureless, Projection and Regularity factors. Dataset `V1_01_easy`

Figure 4.22: Comparison of the time to solve the optimization problem for pipeline using Structureless factors (S), Structureless and Projection factors (S + P), and our proposed approach using Structureless, Projection and Regularity factors (S + P + R).

Chapter 5

Conclusion

5.1 Key insights

Overall, we have seen that enforcing structural regularities, in particular co-planarity constraints between landmarks, provides more accurate state and map estimates than simply ignoring these structural regularities. In particular, the state estimation improves its global consistency by 26% (Absolute Position Error), while it improves its local consistency by up to 50% (Relative Position Error) in scenes with structural regularities. The accuracy and completeness of the mesh also benefit from enforcing structural constraints by a significant degree.

We have also shown a way to incrementally build a 3D mesh using only the sparse keypoints detected, while also presenting a way to reduce the mesh to the time-horizon of the optimization problem. Therefore, the mesh spans multiple viewpoints, thereby covering a larger area; yet the size of the mesh remains bounded, allowing for real-time operation.

Finally, we have shown that the proposed VIO algorithm surpasses in accuracy the state-of-the-art in scenes exhibiting structural regularities.

We have hence confirmed, in real experiments, the conclusions drawn by previous authors [73] about the benefits of using structural regularities. We have also implemented, to the best of our knowledge, the first VIO pipeline to use Structureless [119] and Projection factors (both monocular and stereo) simultaneously. And hence the first to also show the benefits of using Structureless, Projection and Regularity factors in the same optimization problem.

5.2 Future Work

Our current implementation would greatly benefit from improving the current frontend of the pipeline. As it is now, the types of structural regularities that we can detect are rather limited. Indeed, we can only hope to recover horizontal or vertical planes in the scene. Ideally, the frontend would be capable of extracting all planar surfaces in the scene, and encode these constraints in the optimization problem. To this end, one can imagine to use a convolutional neural network to detect potentially planar surfaces in the image. The landmarks associated to the keypoints falling in planar regions could then be used to infer initial plane parameters by using a simple consensus approach such as RANSAC.

Another straigthforward extension would be to use an RGB-D camera that easily extracts depth information of textureless planes, which are the main source of problems for RGB cameras.

Yet another aspect from the frontend that we wish to improve is the quality of the 3D mesh. An ambitious approach that would substantially increase the quality of the mesh would be to enforce that the incremental updates of the mesh maintain the manifold property. Recent works have managed to do so in a reasonable frequency, which could be amenable for use in real-time applications [59]. Alternatively, it would be interesting to use a multi-resolution quad-tree based mesh such as in [41], that would make the handling of triangles covering depth disparities more amenable.

An increased quality of the mesh could in turn allow us to simplify the optimization problem and the complexity of the mesh, reducing the number of vertices (landmarks) of the mesh; and in turn the number of variables in the optimization. One approach would be to merge triangles along the plane estimates [127] to simplify the mesh. Reduce the state vector by replacing points by the plane as in [88] or maybe using a multi-resolution mesh as in [41].

The plane to plane association is currently not very robust since it relies simply on the difference between normals and distances to the origin of the planes. The approach of Hsiao et al. [91] is able to reduce the number of false positive associations by ensuring that the associated planes both describe the combined set of landmarks represented by each plane.

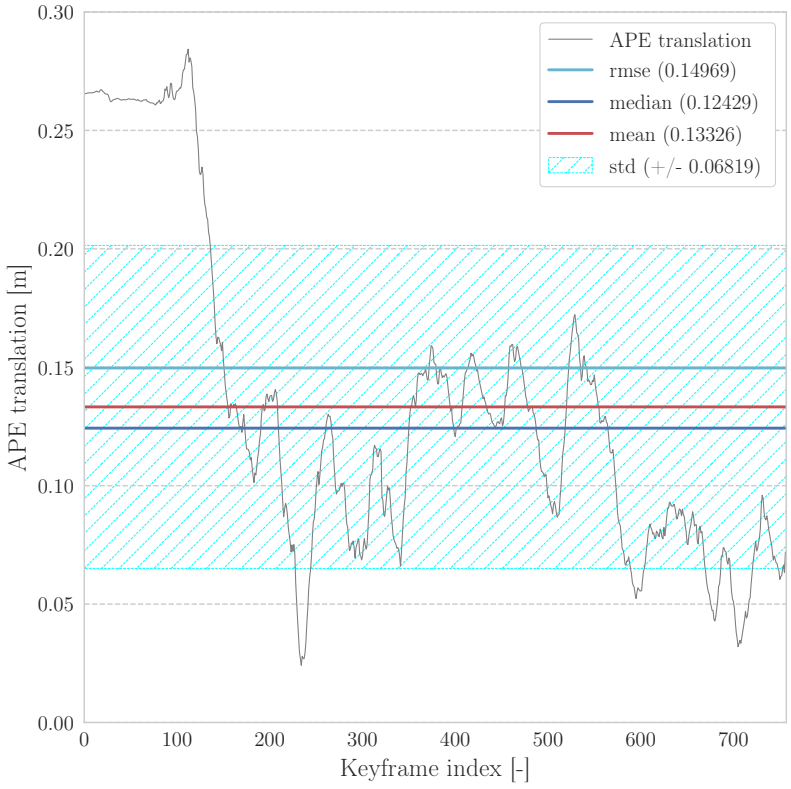
Also, in terms of data association, we are now using hard associations, instead of using a probabilistic approach with soft labels using an Expectation Maximization framework as in [114].

On the performance assessement side, we could use a synthetic dataset to more effectively assess the performance changes that noise levels have. An example of dataset we could use is a synthetic Blender-generated city by Zhang et al. [128], from which we can even generate RGB-D images.

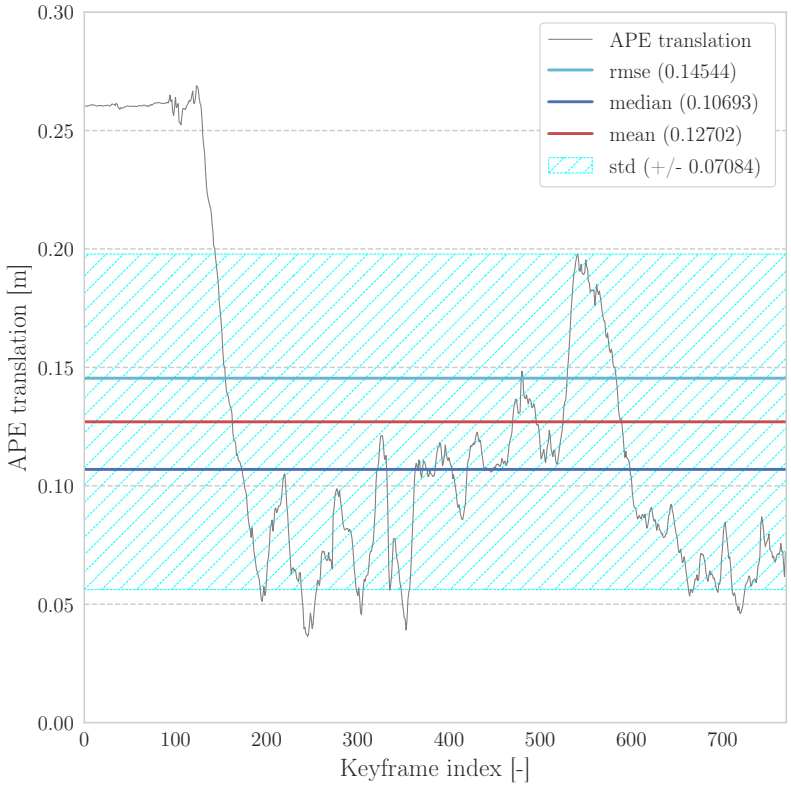
On the optimization's side it would be also interesting to use a regularity constraint between landmarks and planes defined in image space as in [99], instead of geometrically, as we do in § 3.1.5.

Appendix A

APE translation errors

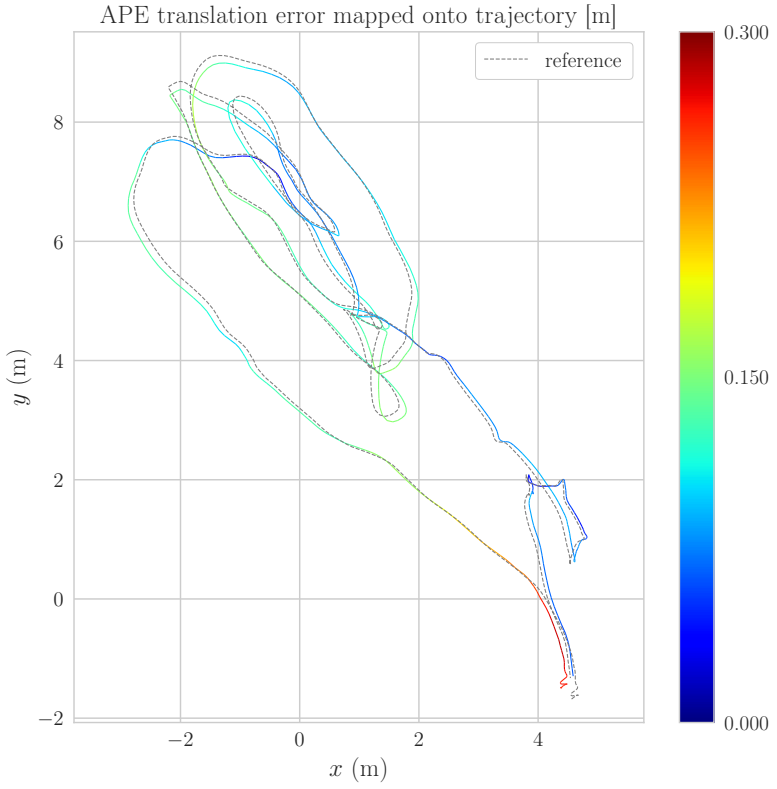


(a) S+P

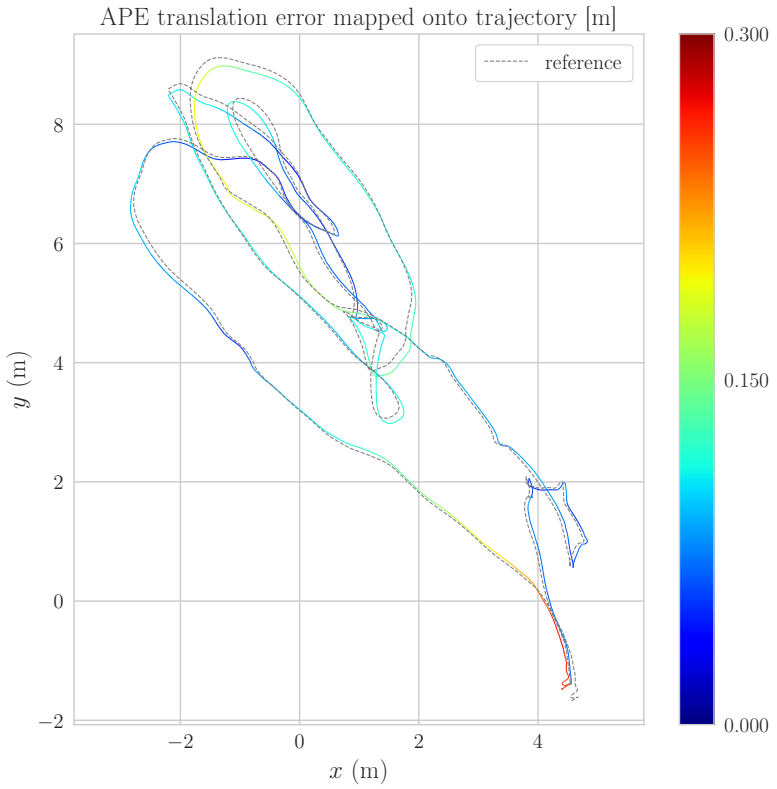


(b) S+P+R

Figure A.1: Dataset `MH_01_easy`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

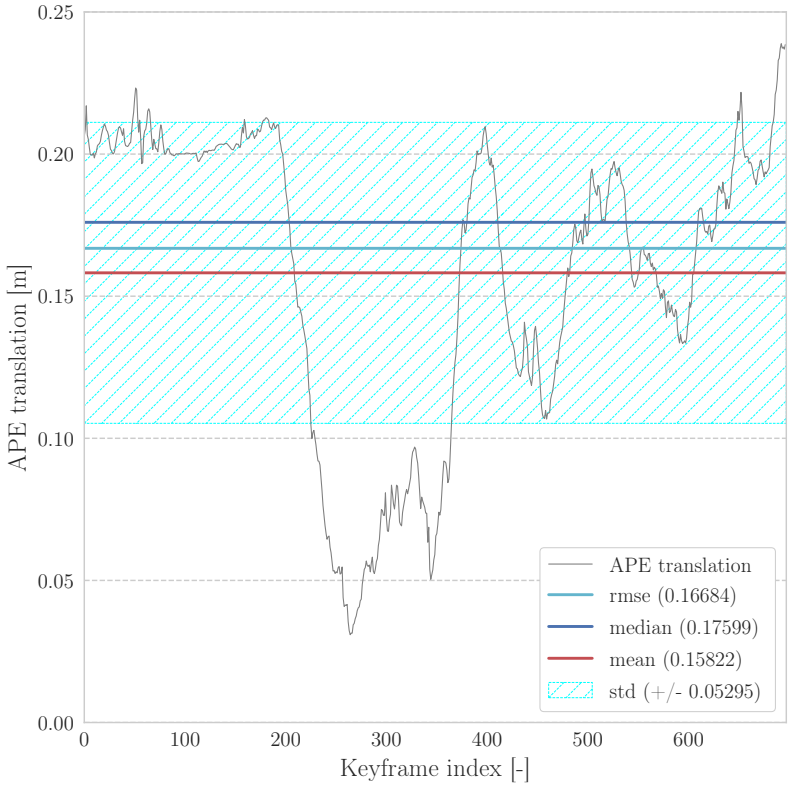


(a) APE translation for S+P

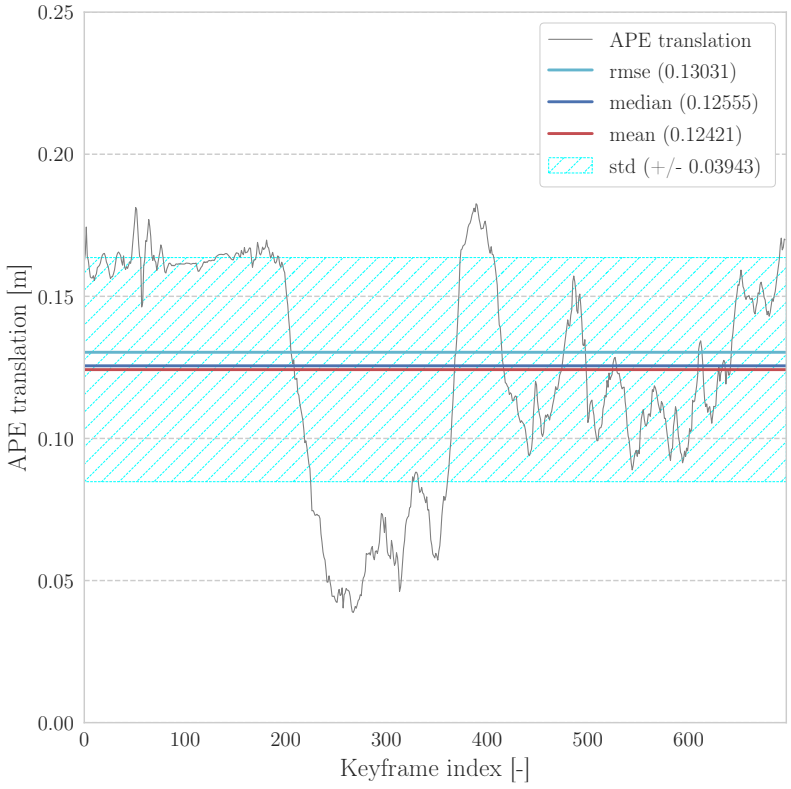


(b) APE translation for S+P+R

Figure A.2: Dataset `MH_01_easy`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

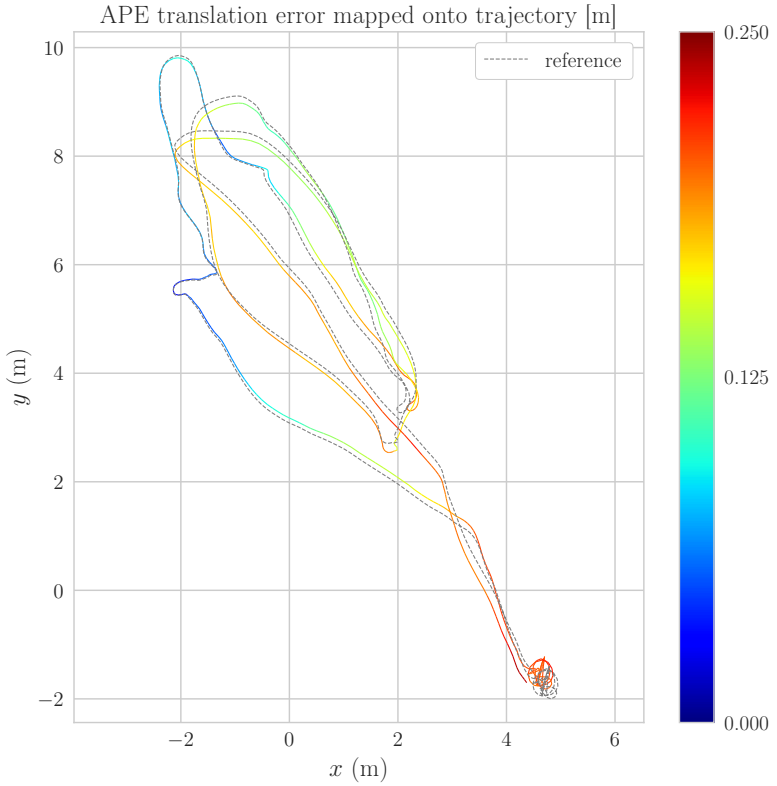


(a) S+P

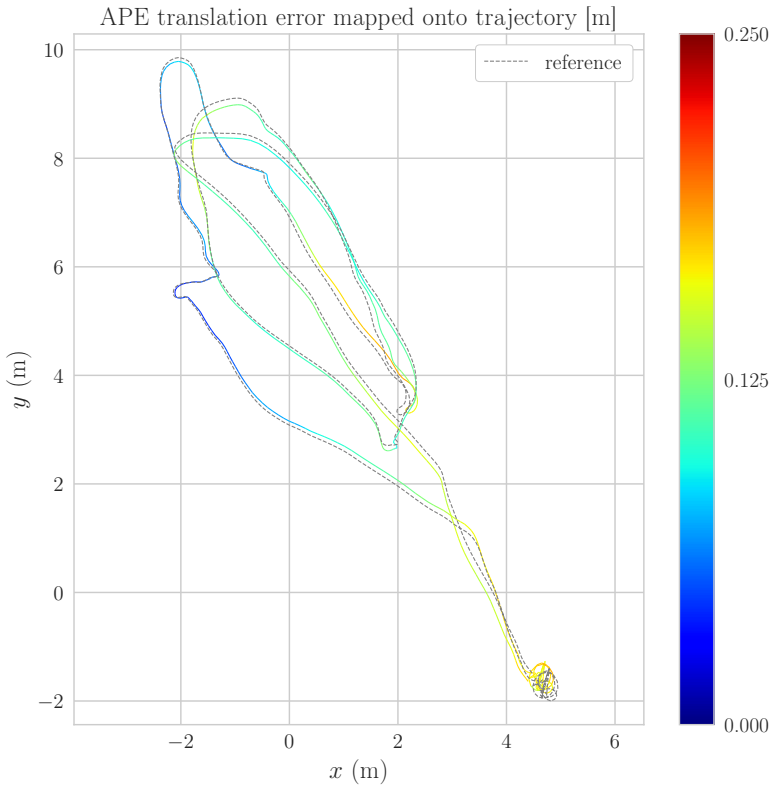


(b) S+P+R

Figure A.3: Dataset `MH_02_easy`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

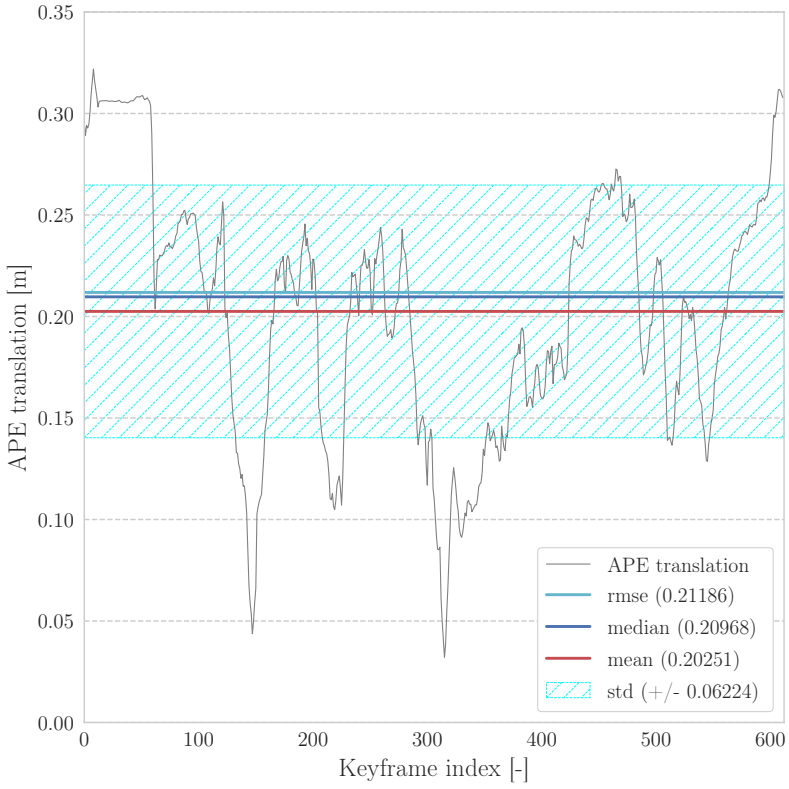


(a) APE translation for S+P

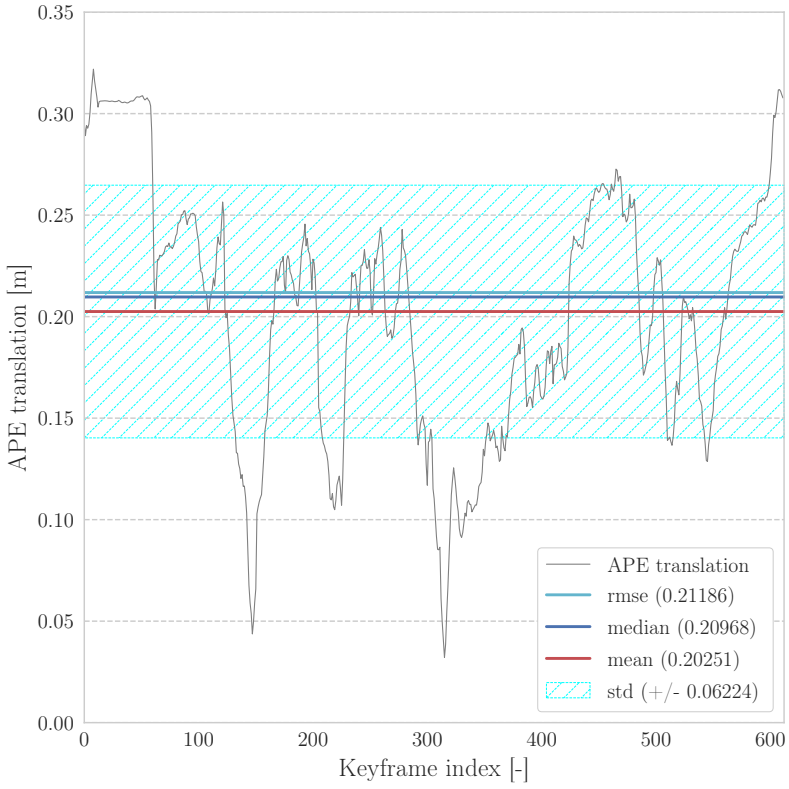


(b) APE translation for S+P+R

Figure A.4: Dataset `MH_02_easy`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

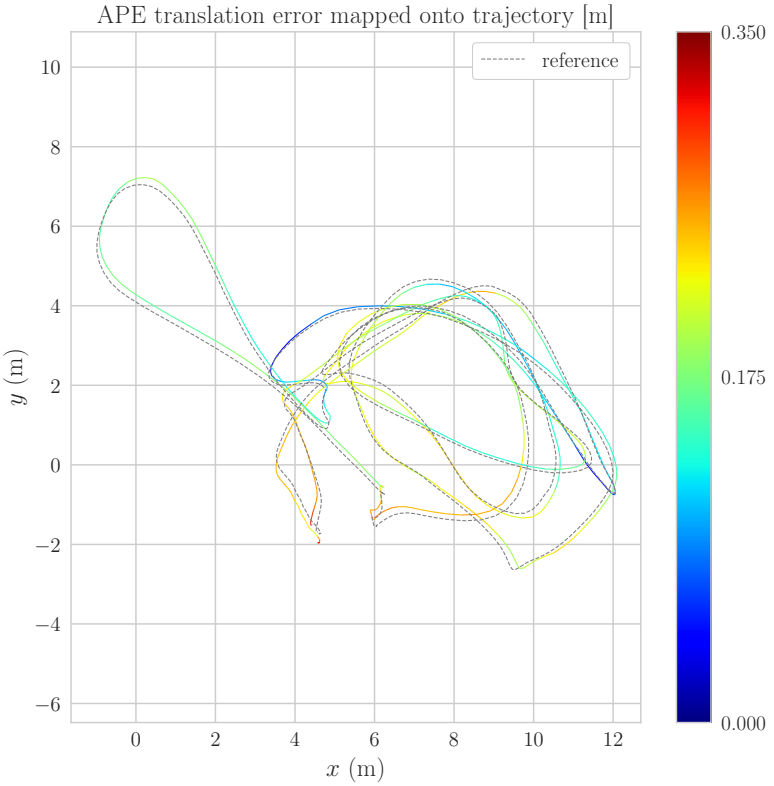


(a) S+P

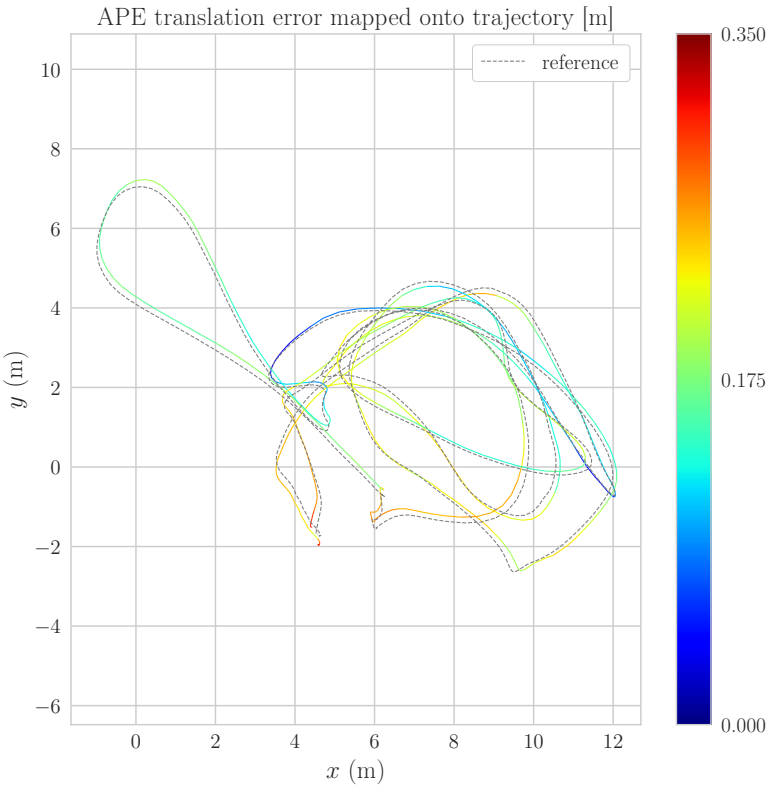


(b) S+P+R

Figure A.5: Dataset `MH_03_medium`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

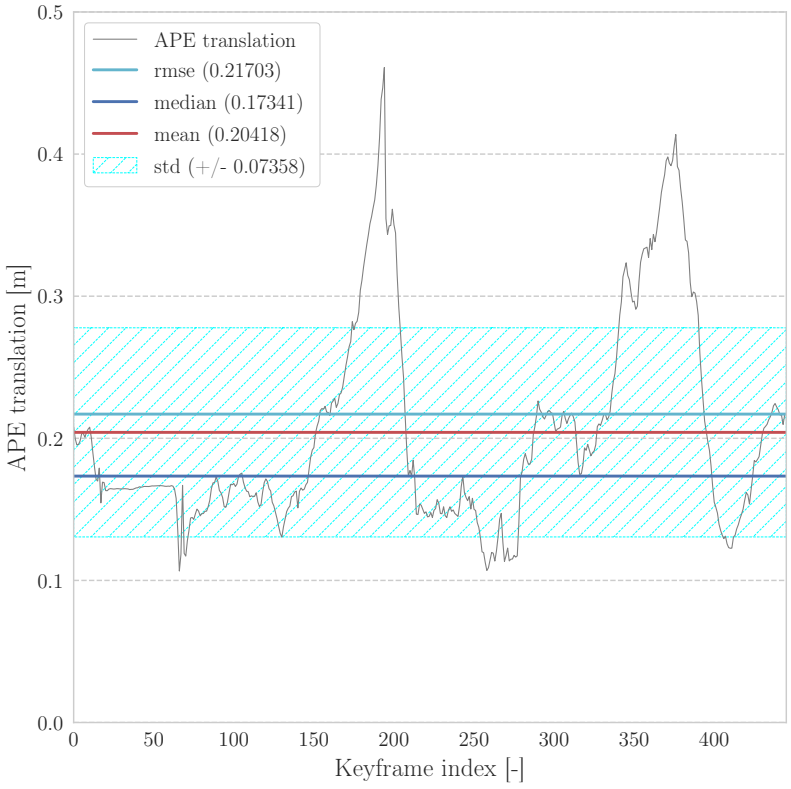


(a) APE translation for S+P

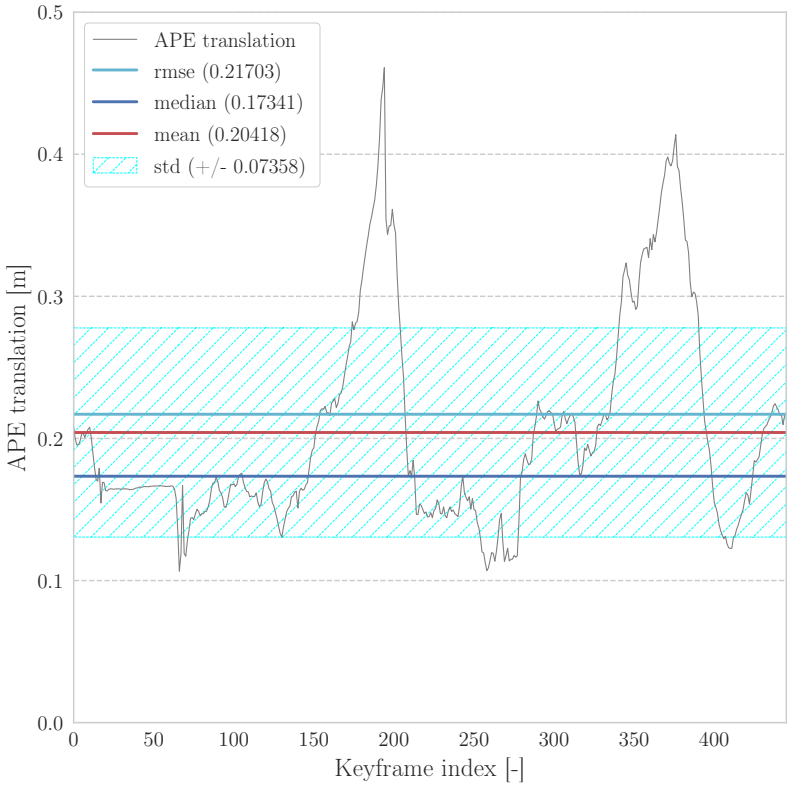


(b) APE translation for S+P+R

Figure A.6: Dataset `MH_03_medium`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

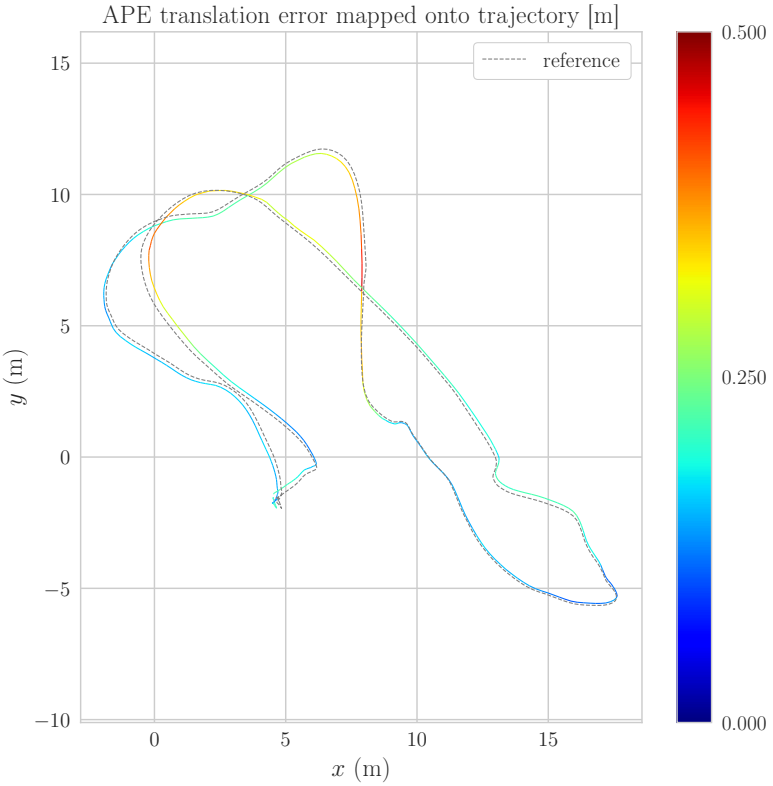


(a) S+P

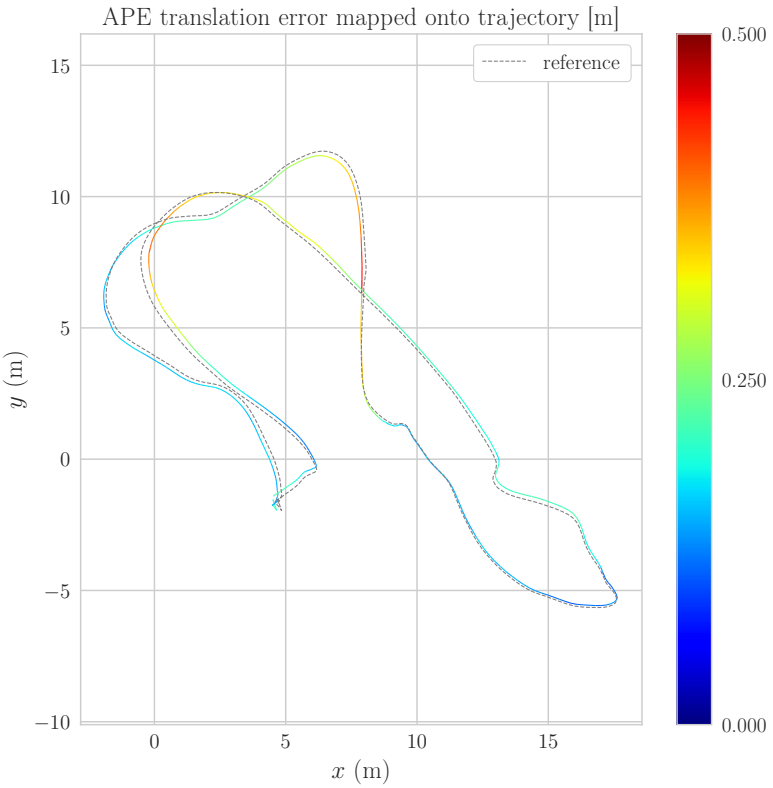


(b) S+P+R

Figure A.7: Dataset `MH_04_difficult`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

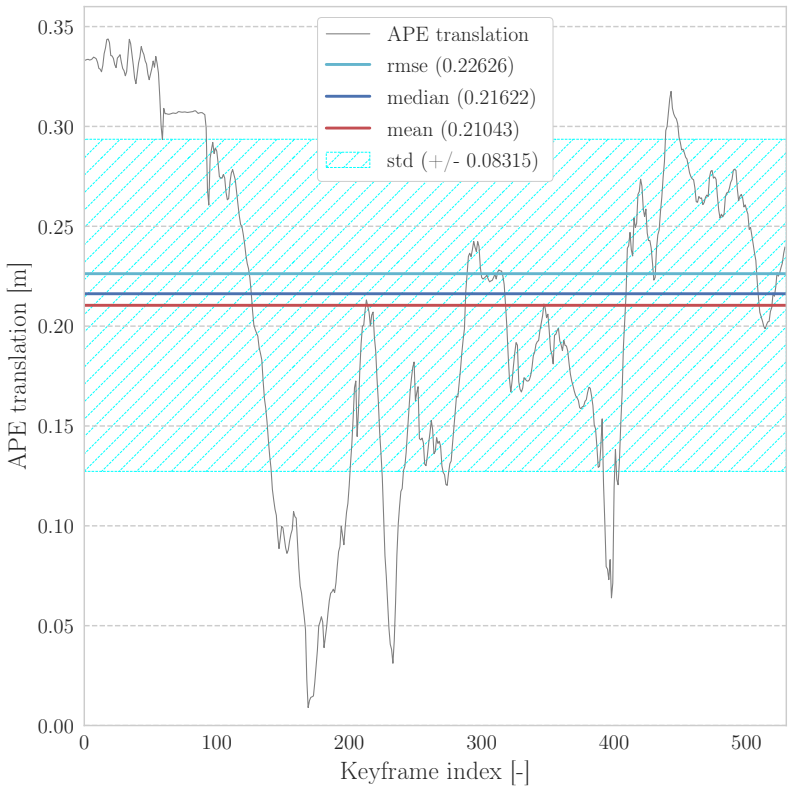


(a) APE translation for S+P

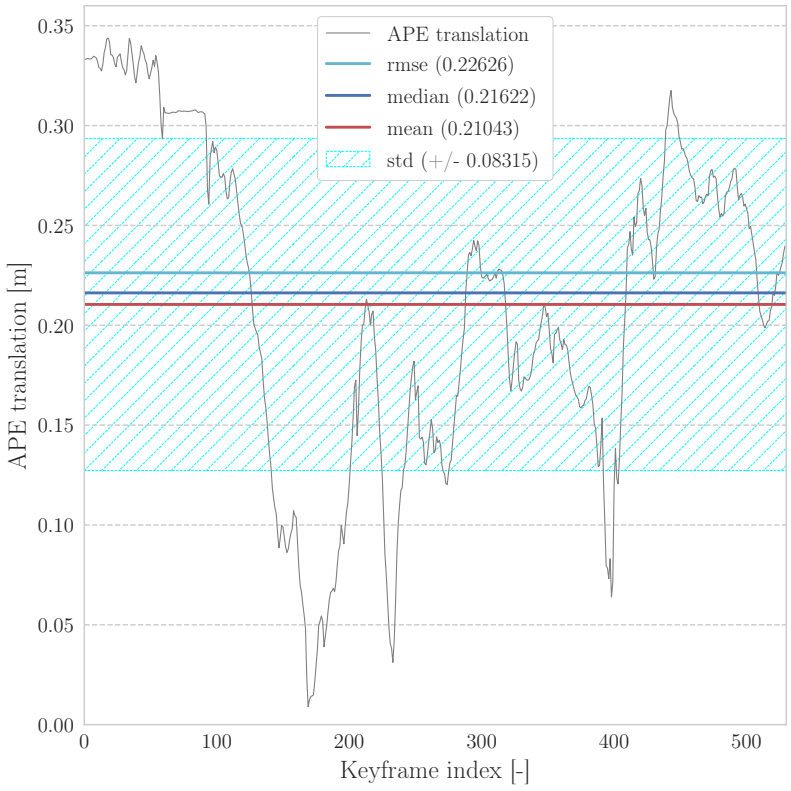


(b) APE translation for S+P+R

Figure A.8: Dataset `MH_04_difficult`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

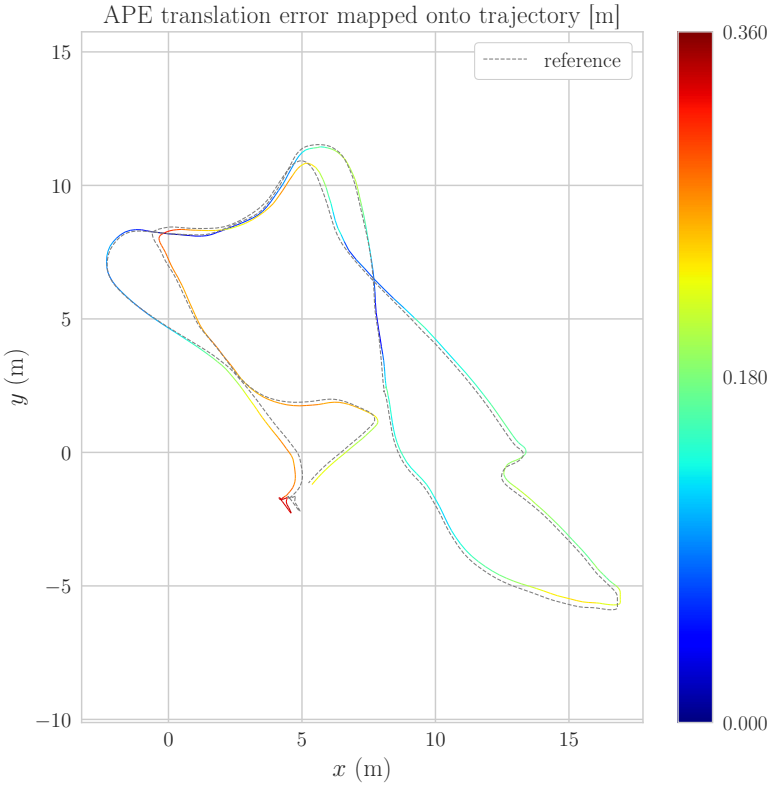


(a) S+P

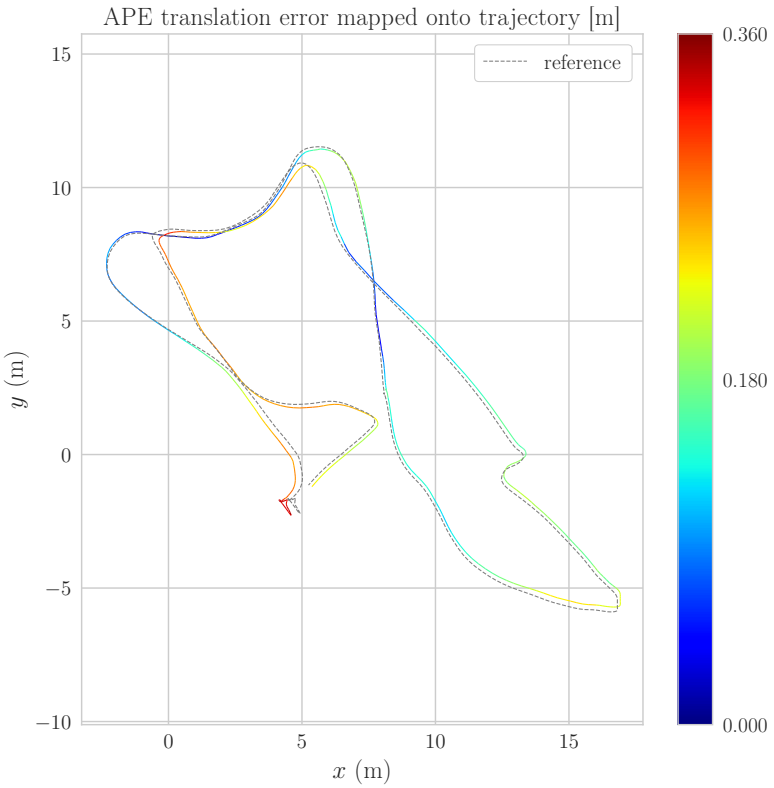


(b) S+P+R

Figure A.9: Dataset `MH_05_difficult`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

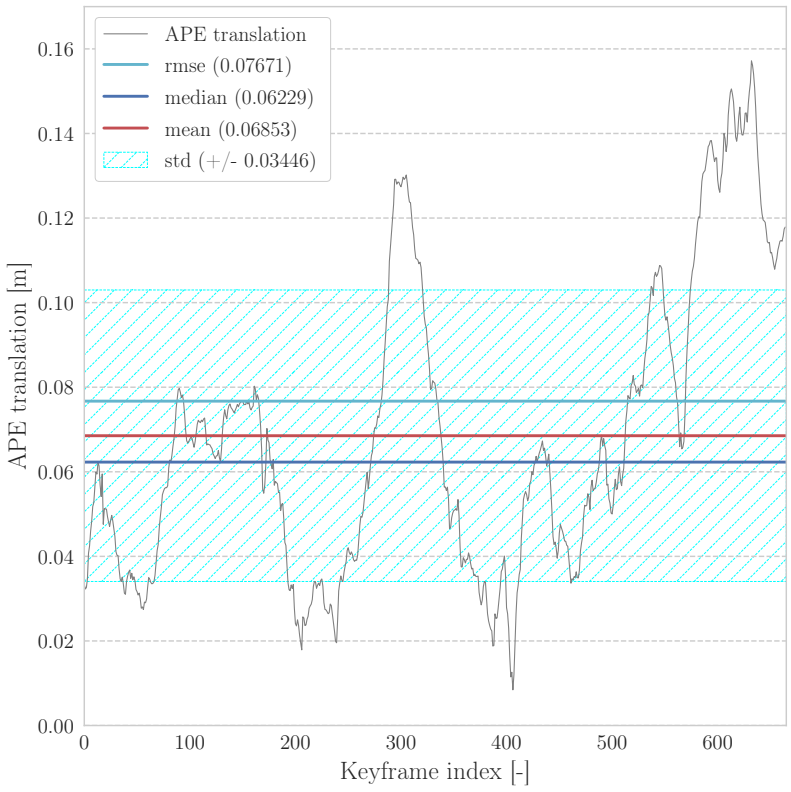


(a) APE translation for S+P

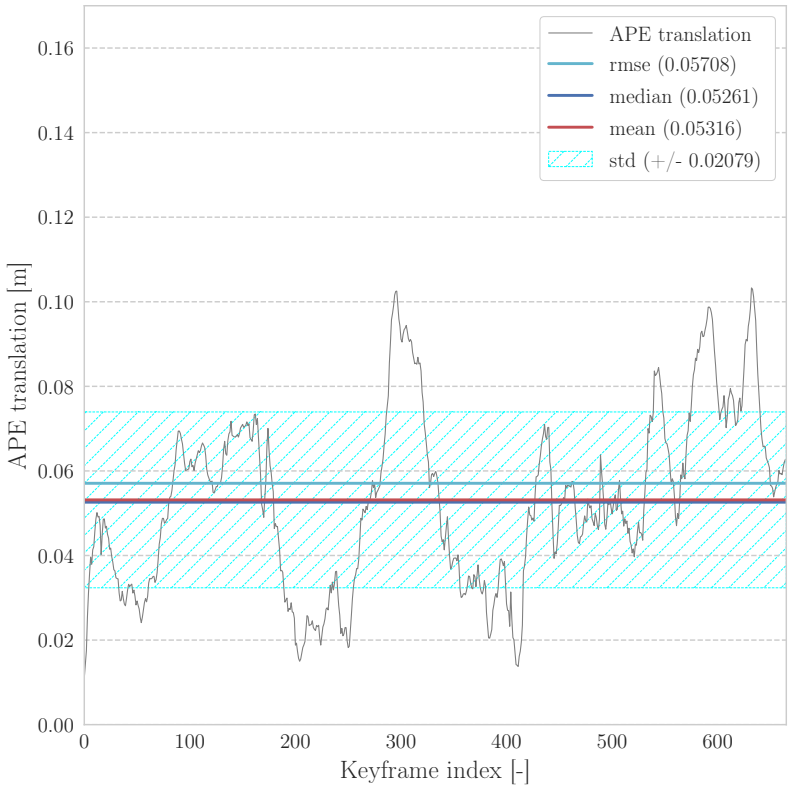


(b) APE translation for S+P+R

Figure A.10: Dataset `MH_05_difficult`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

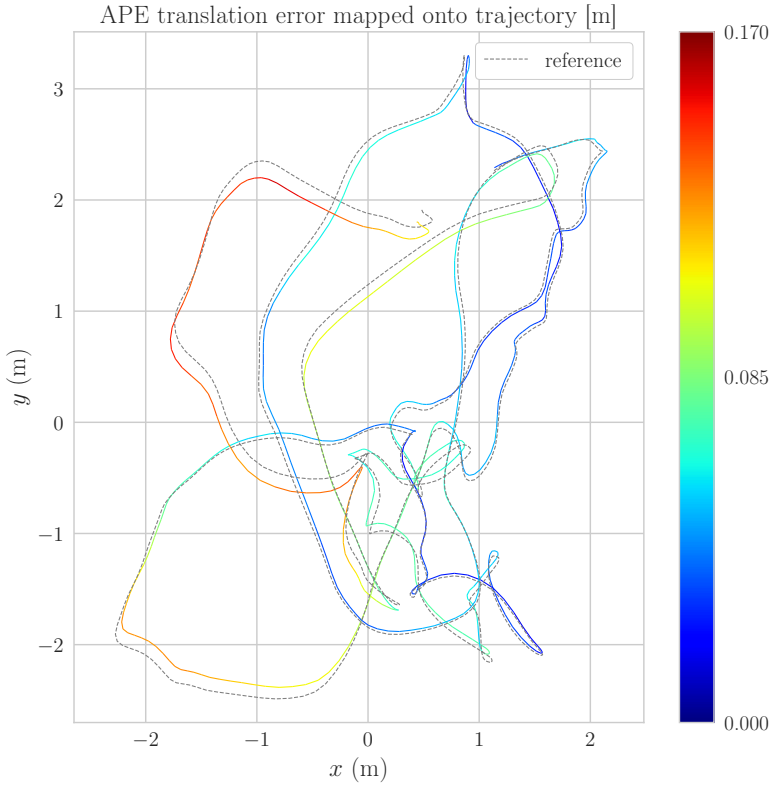


(a) S+P

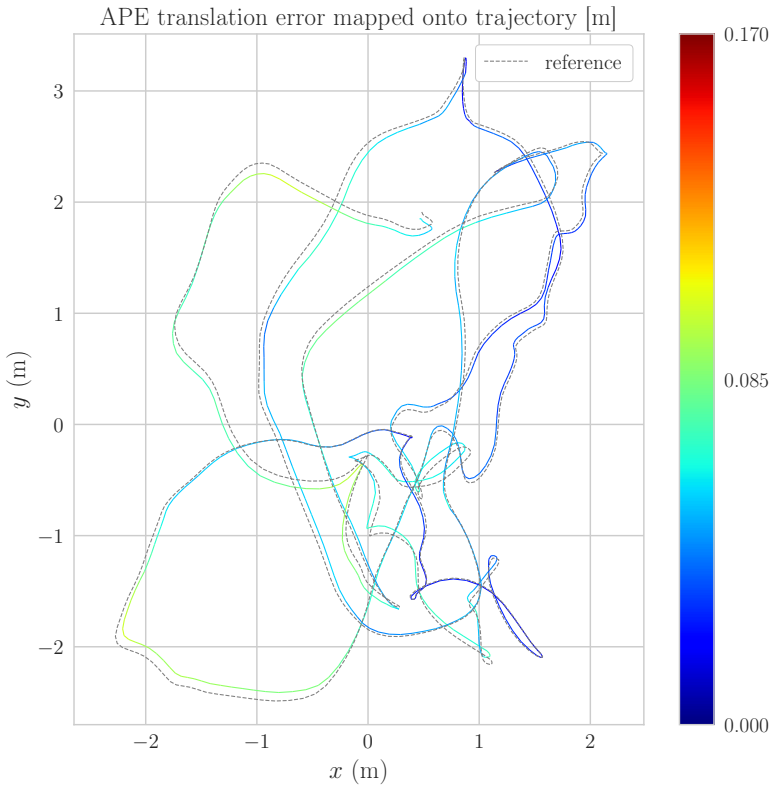


(b) S+P+R

Figure A.11: Dataset `V1_01_easy`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

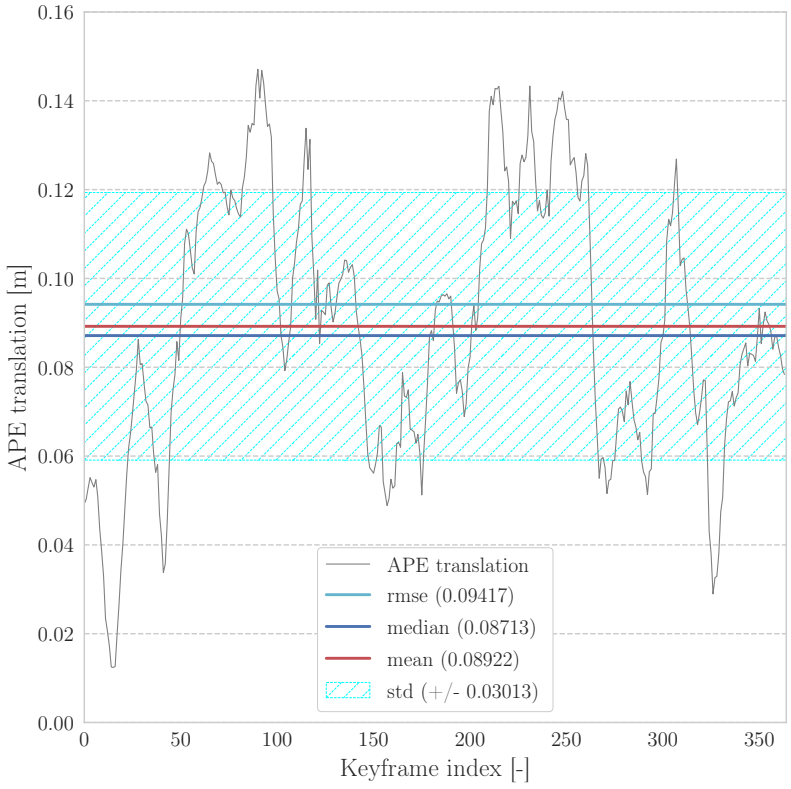


(a) APE translation for S+P

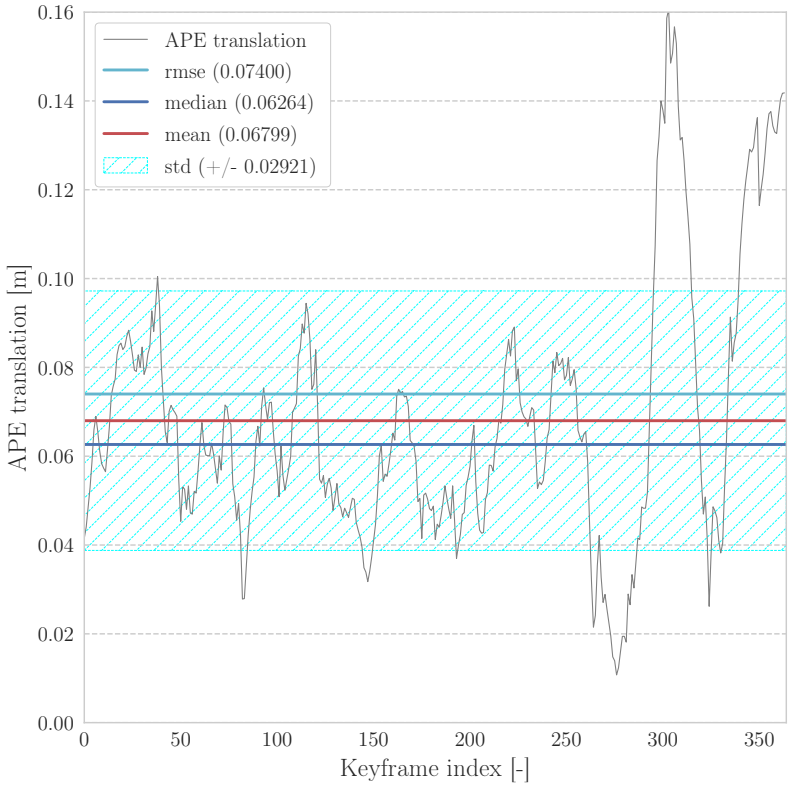


(b) APE translation for S+P+R

Figure A.12: Dataset `V1_01_easy`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

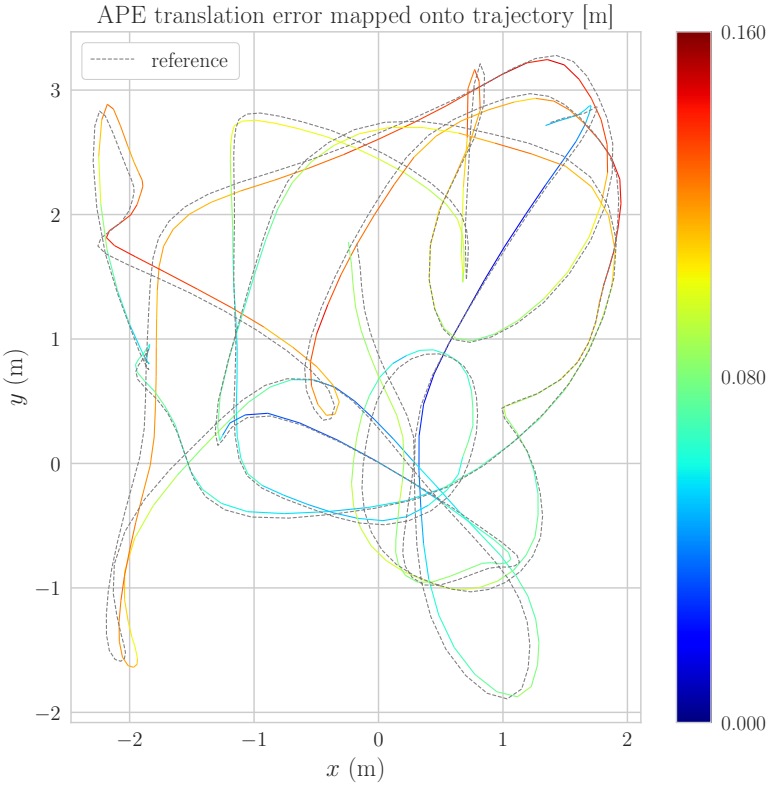


(a) S+P

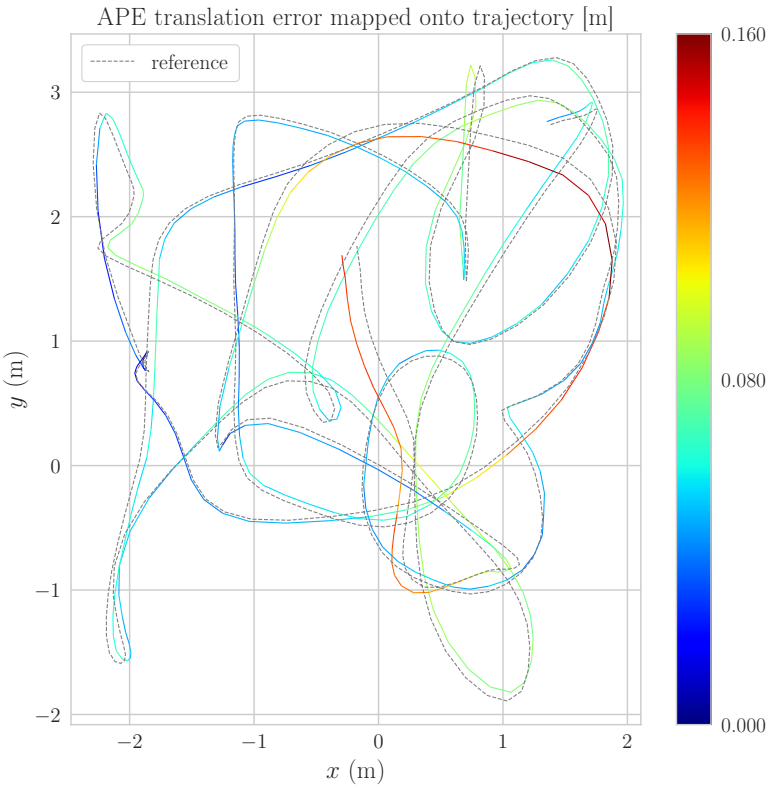


(b) S+P+R

Figure A.13: Dataset `V1_02_medium`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

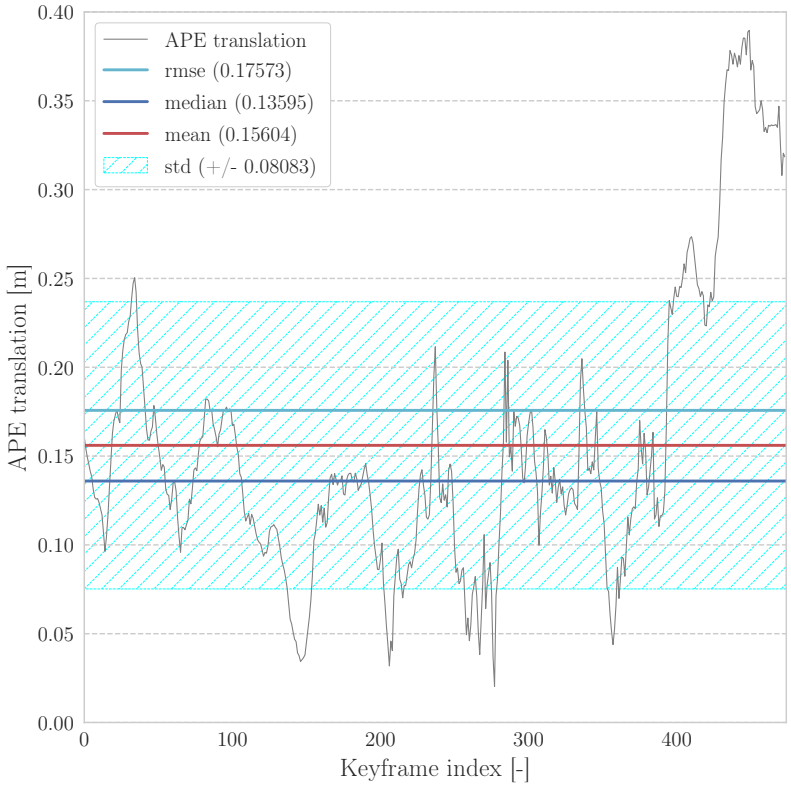


(a) APE translation for S+P

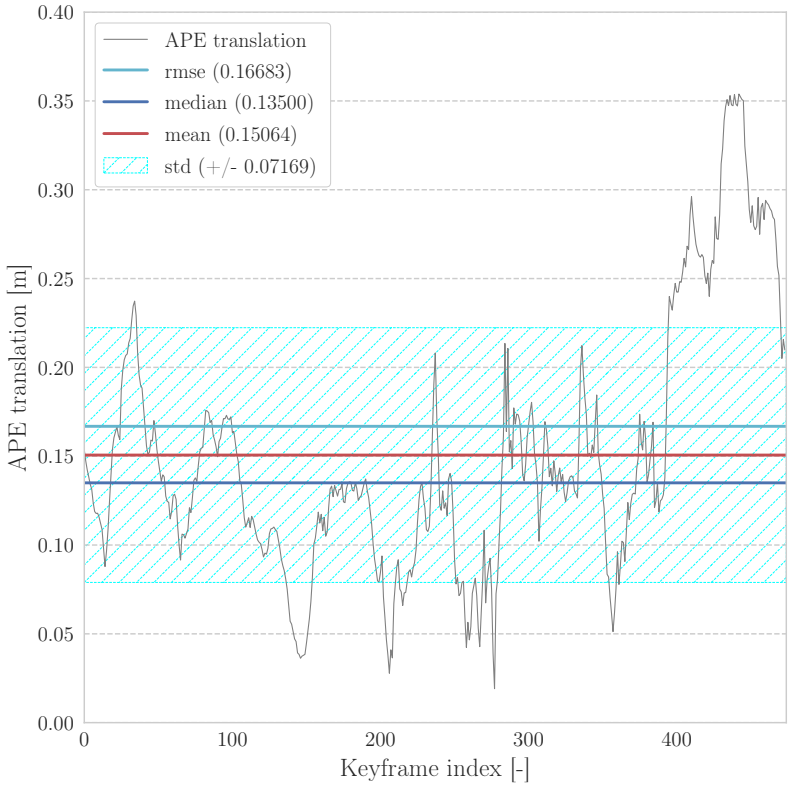


(b) APE translation for S+P+R

Figure A.14: Dataset `V1_02_medium`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

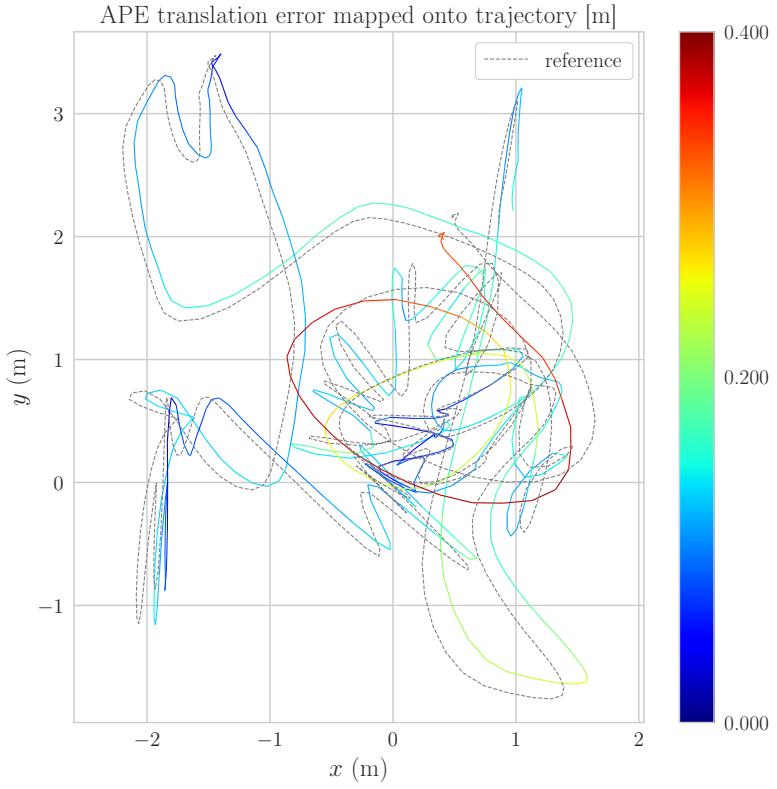


(a) S+P

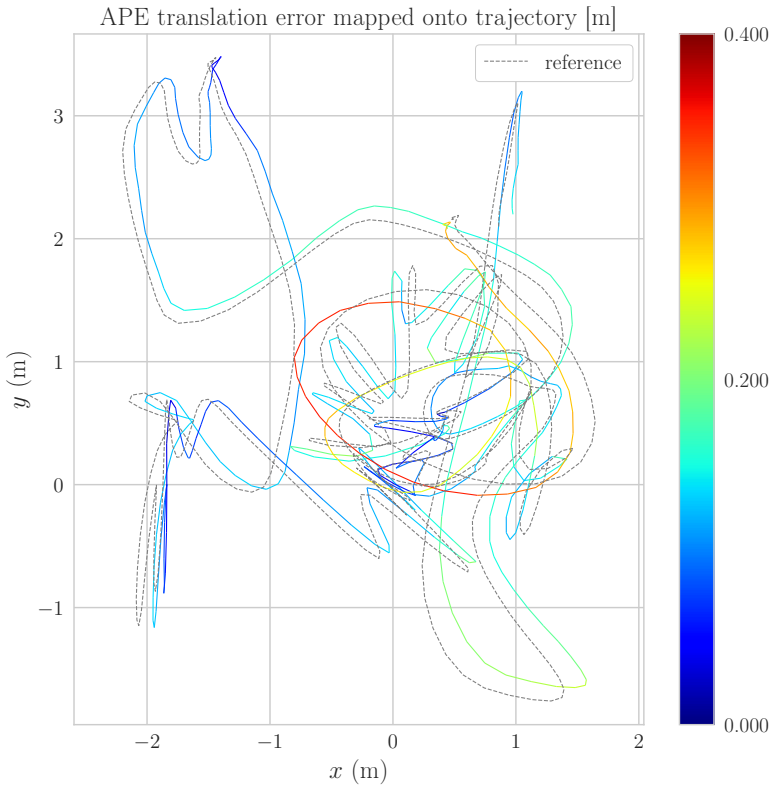


(b) S+P+R

Figure A.15: Dataset `V1_03_difficult`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

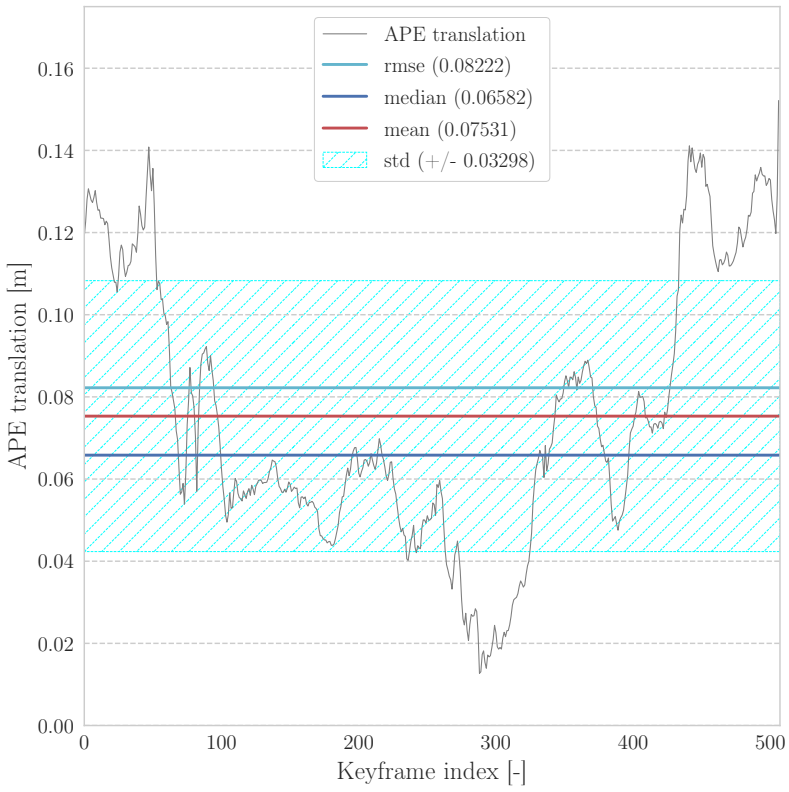


(a) APE translation for S+P

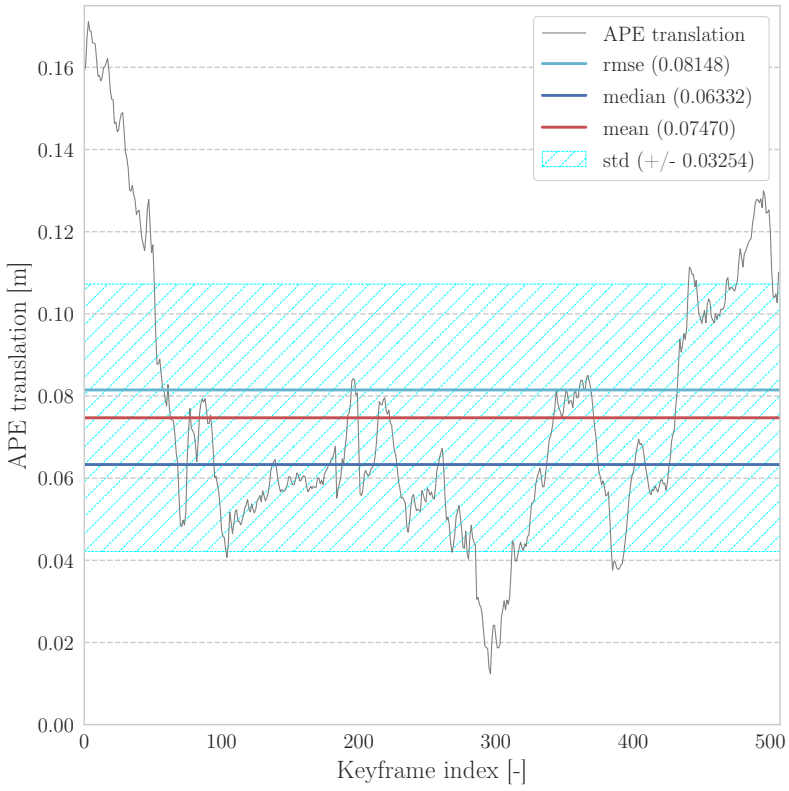


(b) APE translation for S+P+R

Figure A.16: Dataset `V1_03_difficult`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

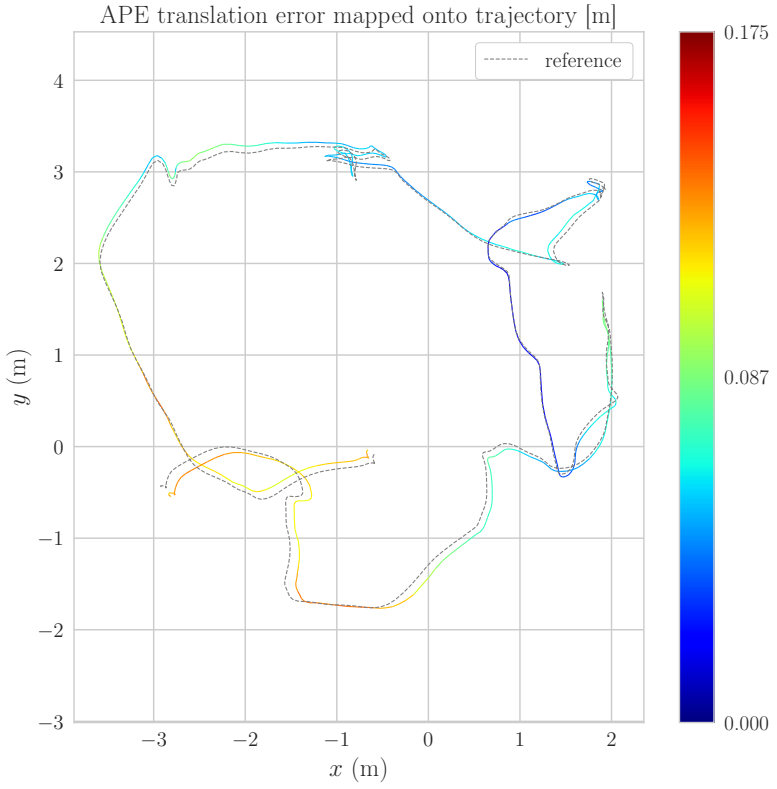


(a) S+P

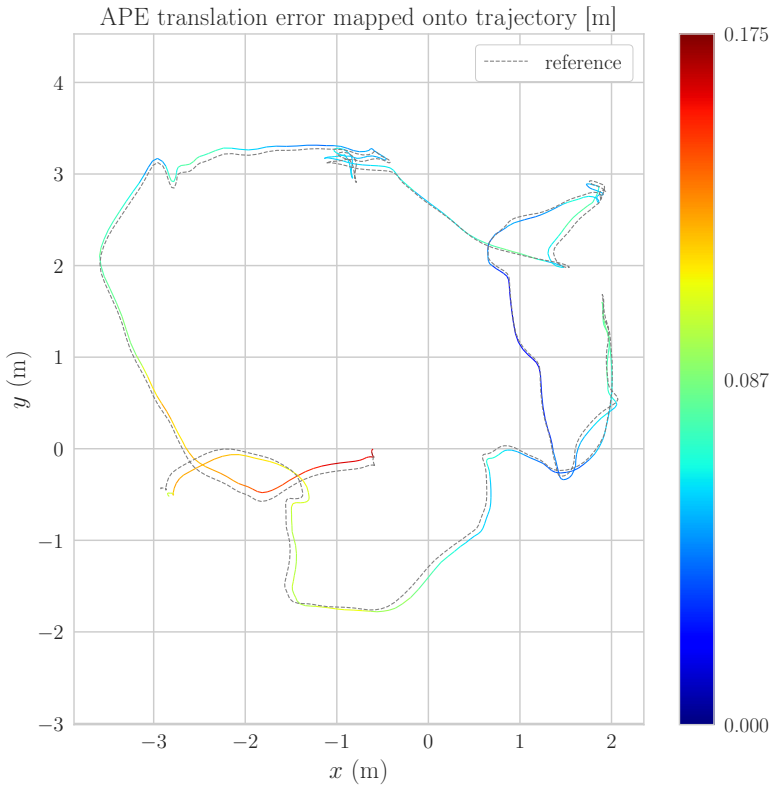


(b) S+P+R

Figure A.17: Dataset `V2_01_easy`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

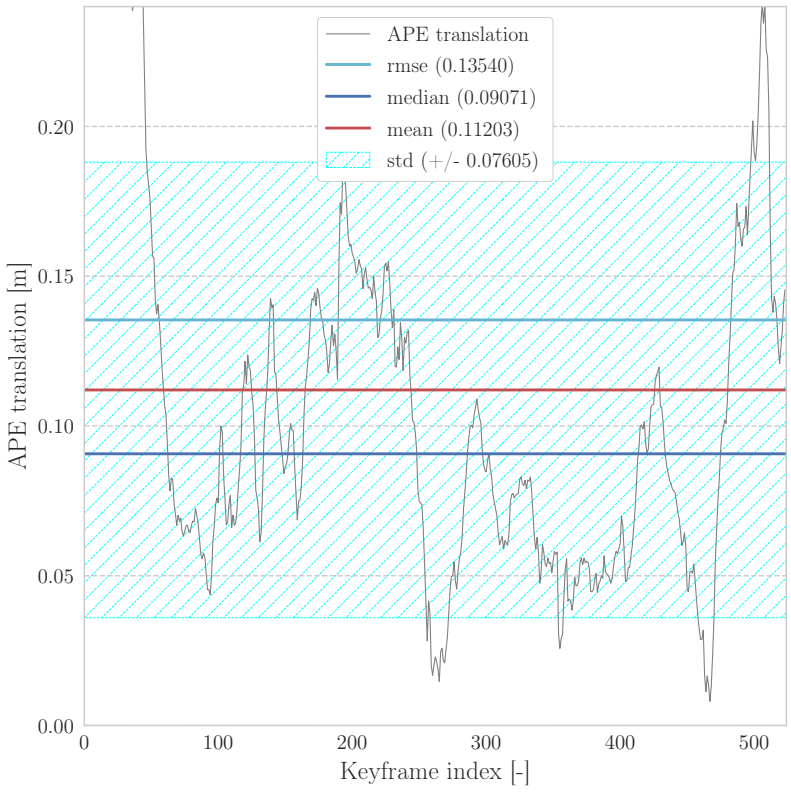


(a) APE translation for S+P

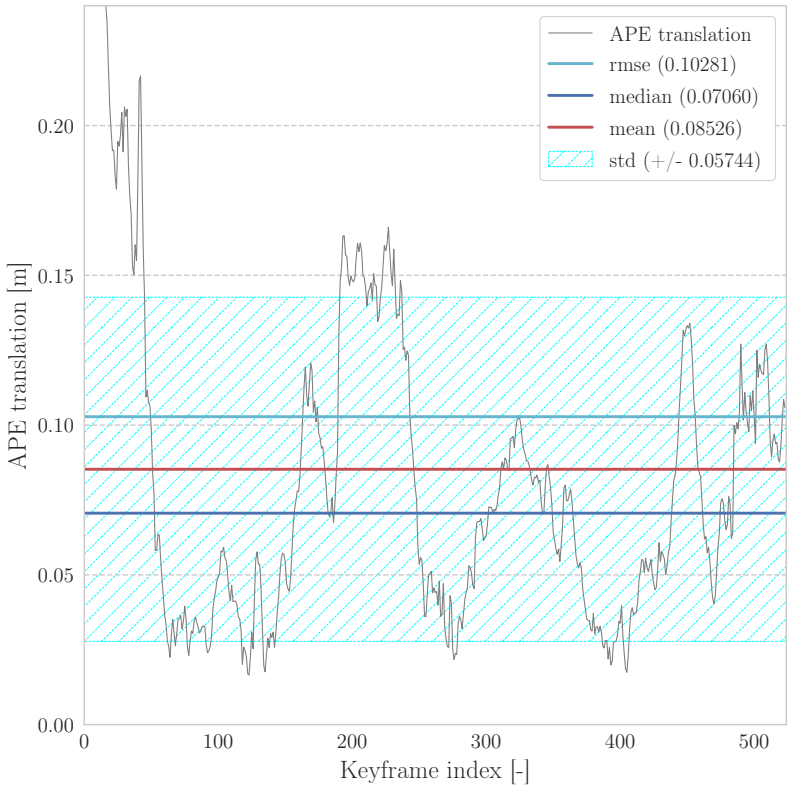


(b) APE translation for S+P+R

Figure A.18: Dataset `V2_01_easy`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

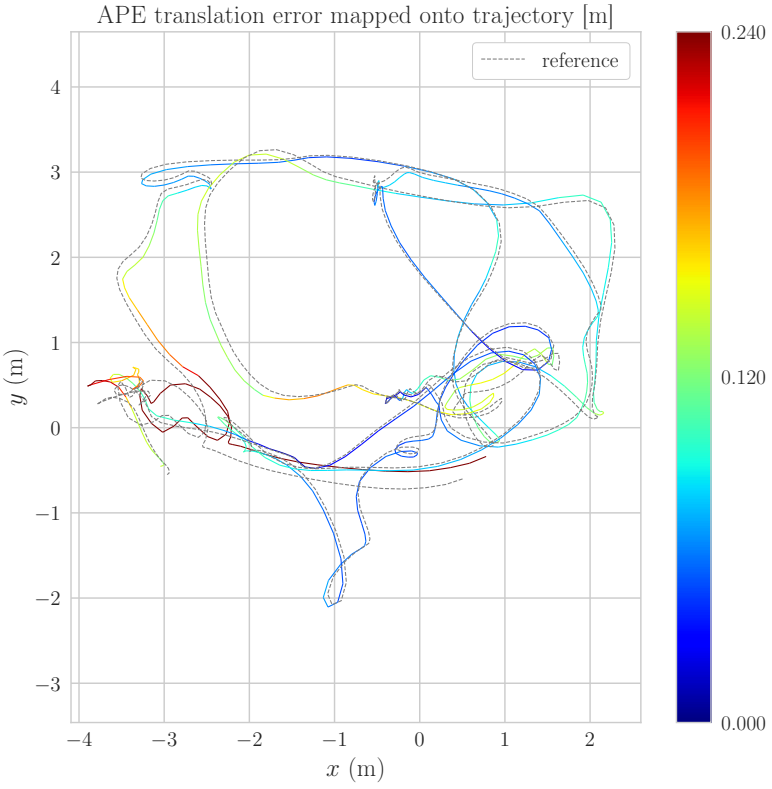


(a) S+P

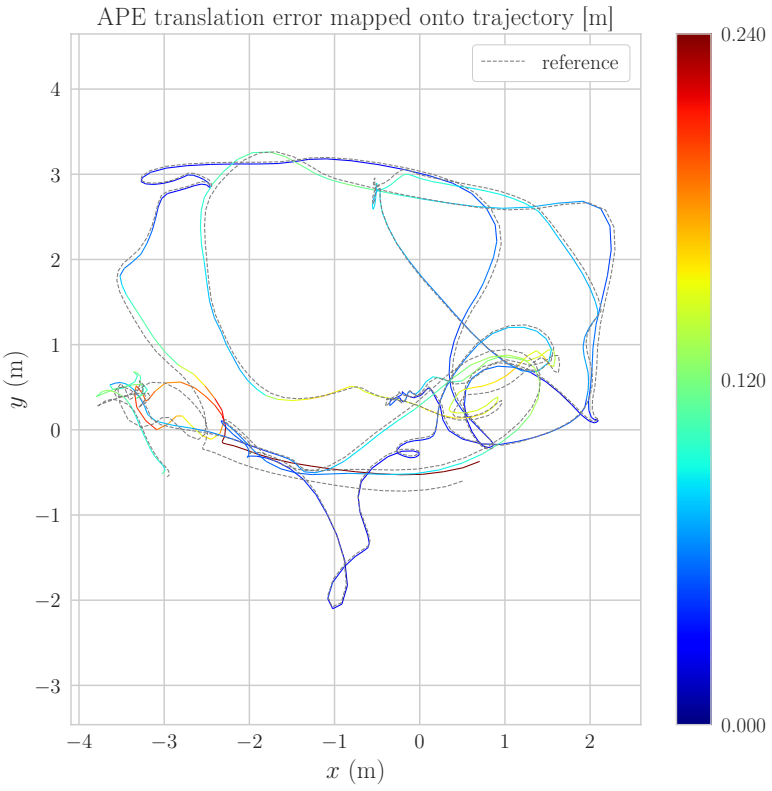


(b) S+P+R

Figure A.19: Dataset `V2_02_medium`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

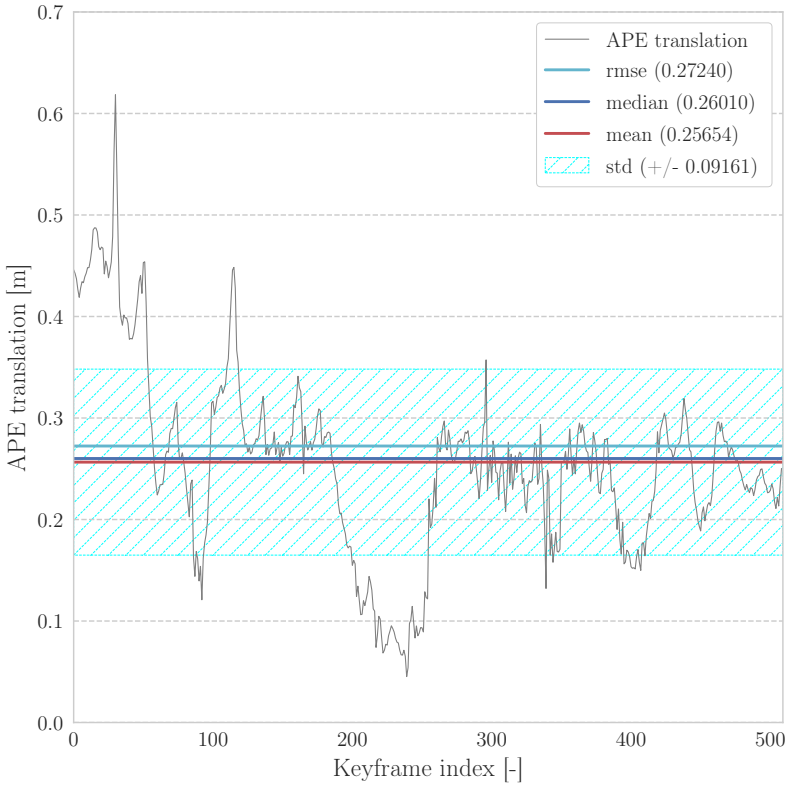


(a) APE translation for S+P

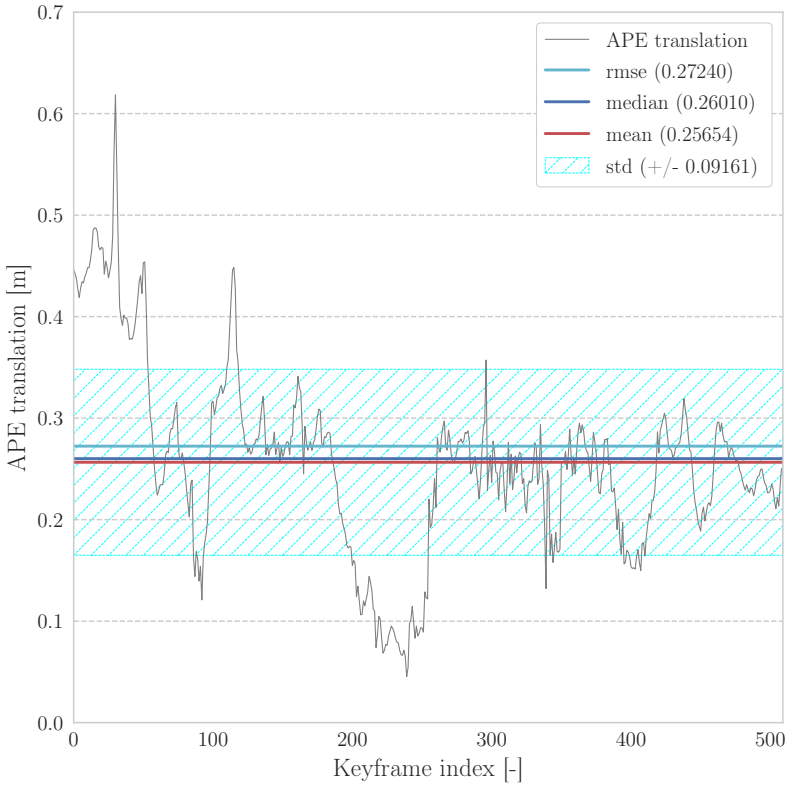


(b) APE translation for S+P+R

Figure A.20: Dataset `V2_02_medium`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

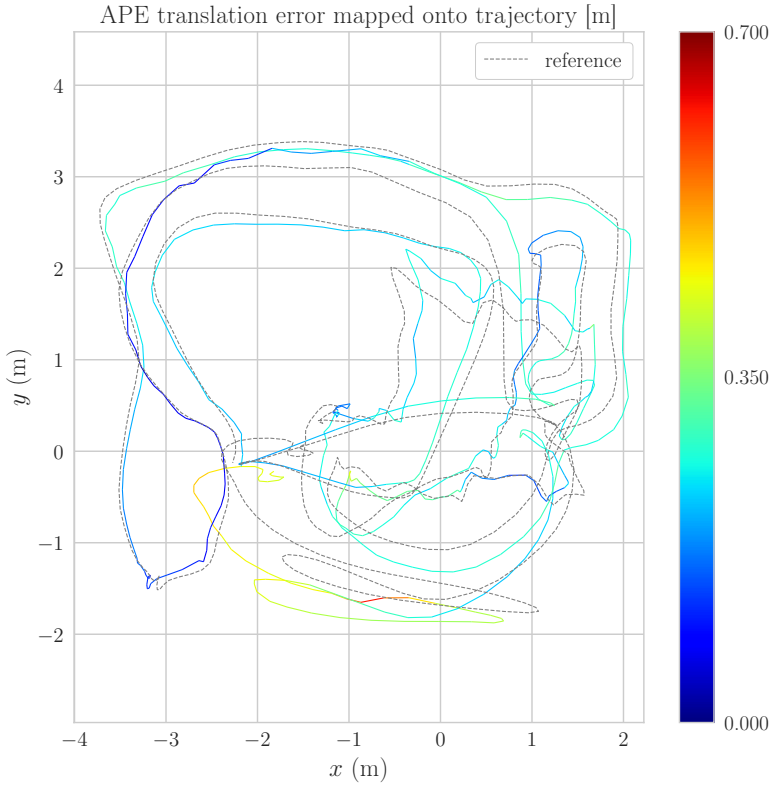


(a) S+P

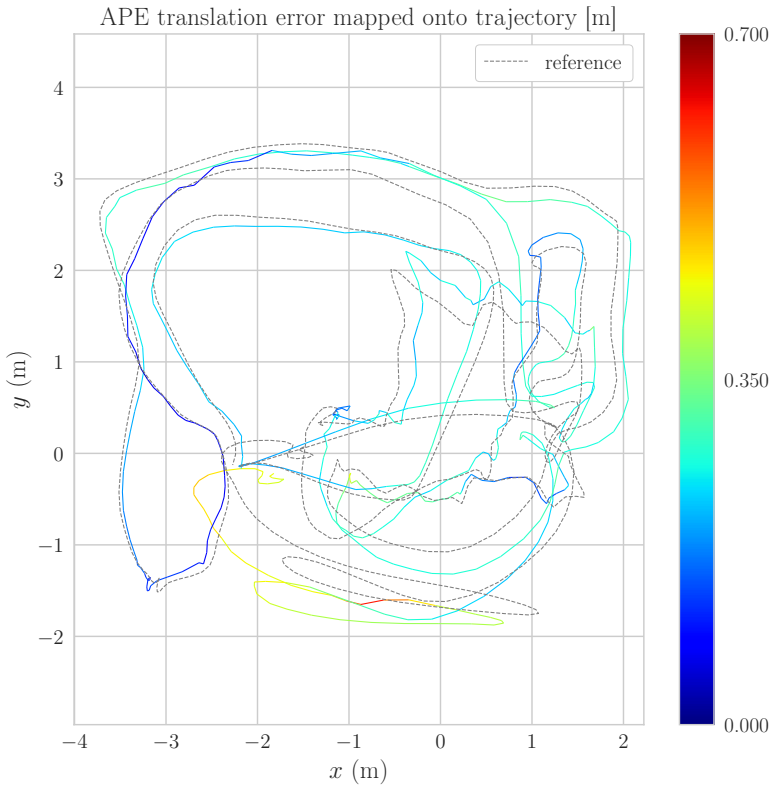


(b) S+P+R

Figure A.21: Dataset `V2_03_difficult`: APE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.



(a) APE translation for S+P

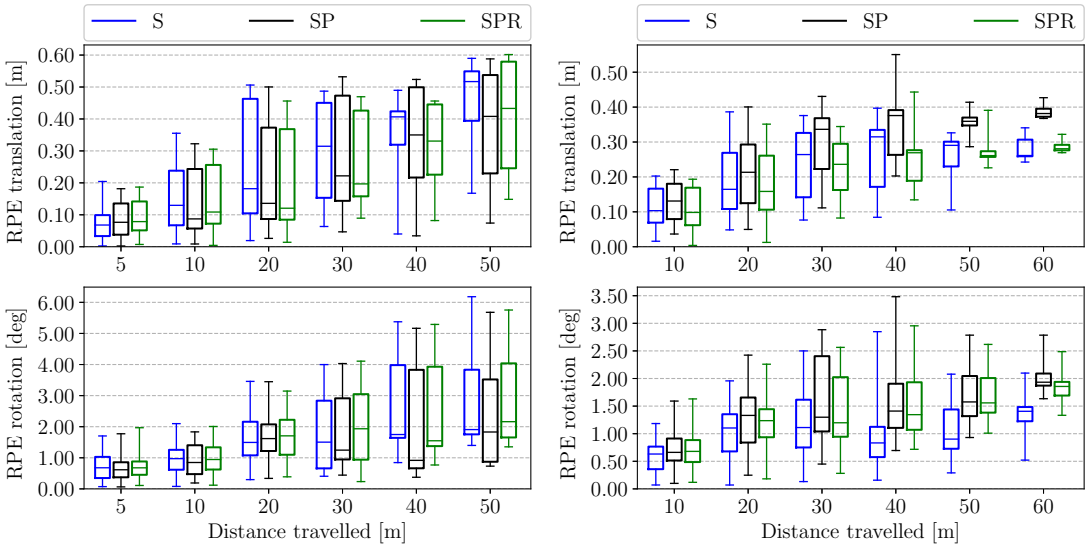


(b) APE translation for S+P+R

Figure A.22: Dataset `V2_03_difficult`: APE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

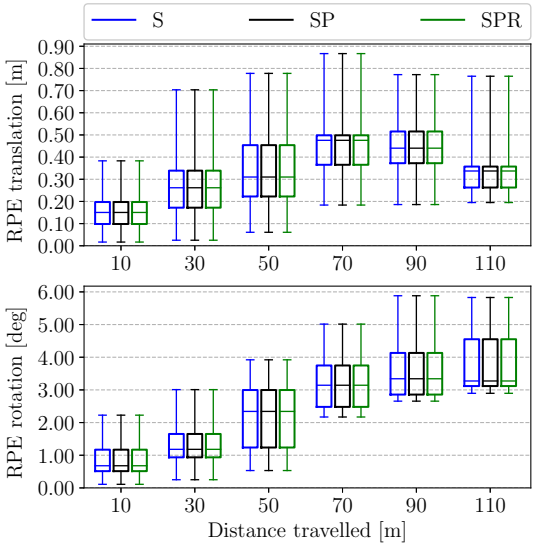
Appendix B

RPE boxplots

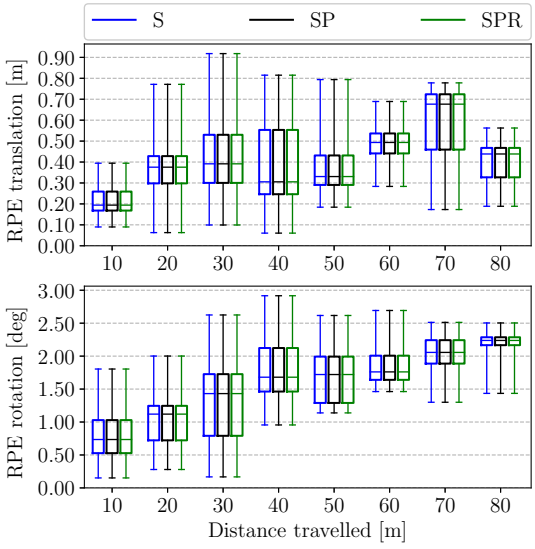


(a) `MH_01_easy`

(b) `MH_02_easy`

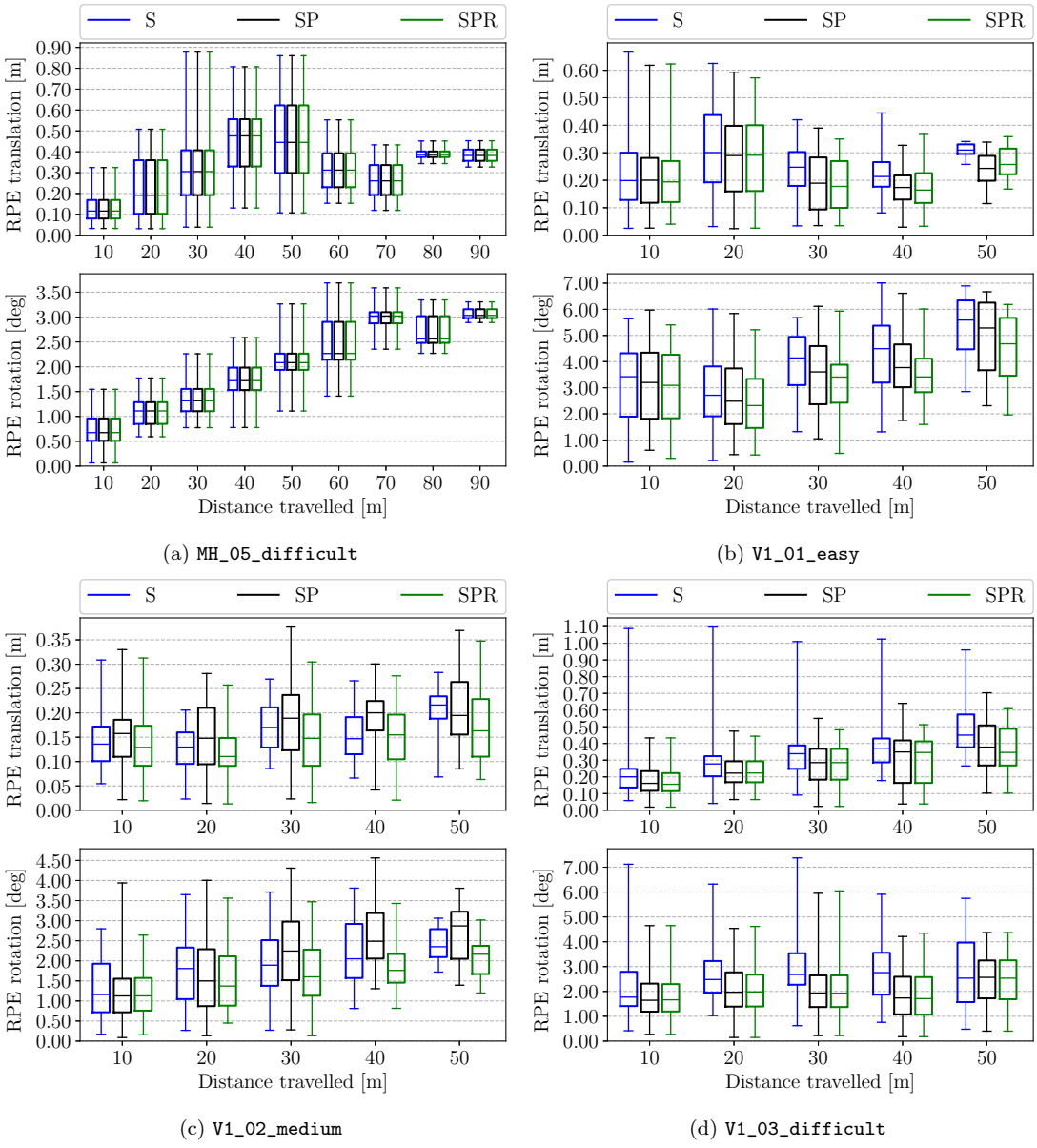


(c) `MH_03_medium`



(d) `MH_04_difficult`

Figure B.1: Detailed comparison of the state estimation accuracy on the EuRoC dataset while using Structureless factors (S), Structureless and Projection factors (SP), and our proposed approach using Structureless, Projection and Regularity factors (SPR).



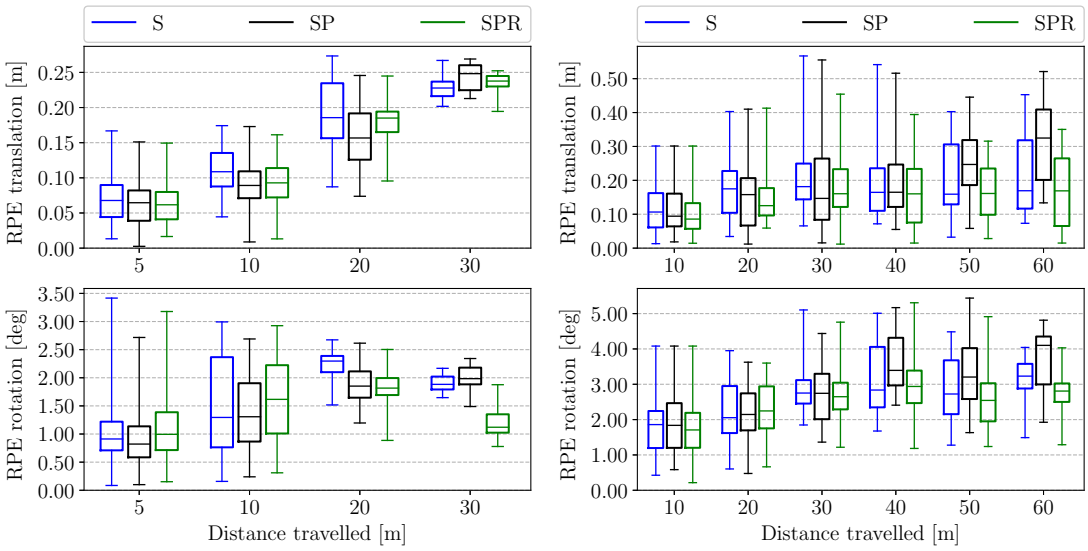
(a) `MH_05_difficult`

(b) `V1_01_easy`

(c) `V1_02_medium`

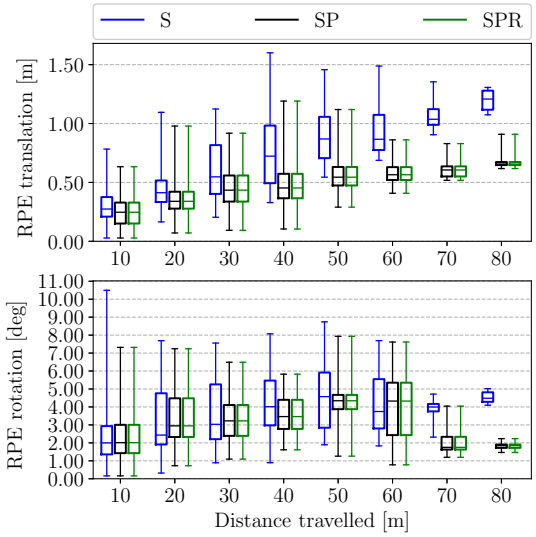
(d) `V1_03_difficult`

Figure B.2: (Continuation) Detailed comparison of the state estimation accuracy on the EuRoC dataset while using Structureless factors (S), Structureless and Projection factors (SP), and our proposed approach using Structureless, Projection and Regularity factors (SPR).



(a) `V2_01_easy`

(b) `V2_02_medium`

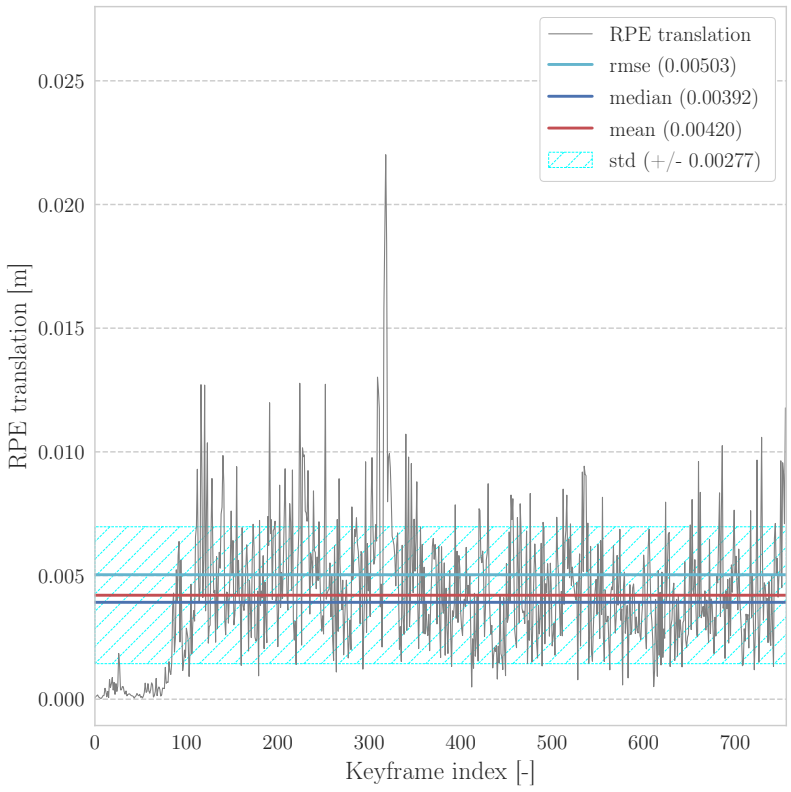


(c) `V2_03_difficult`

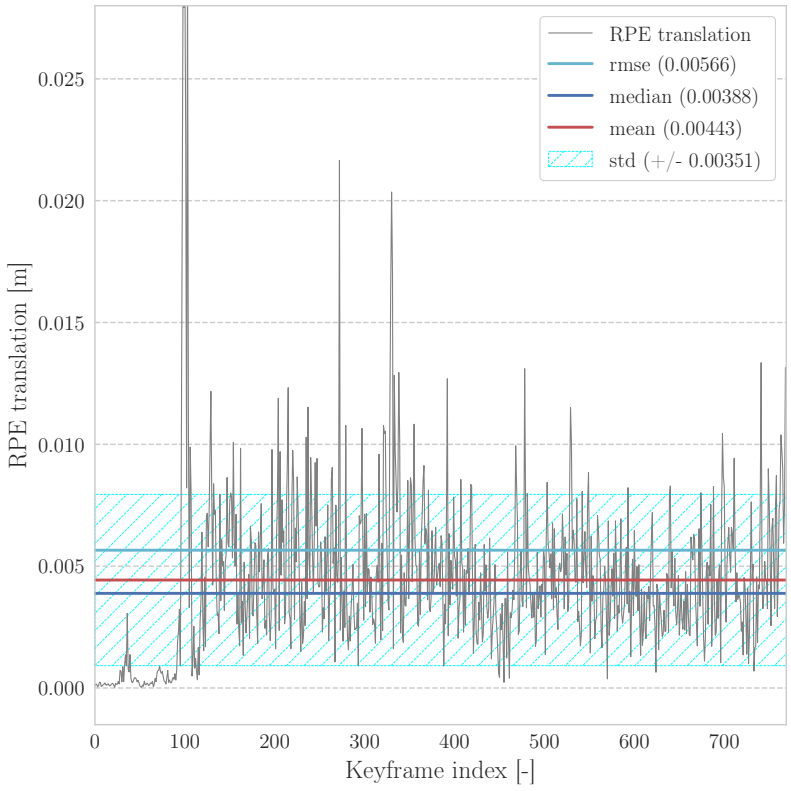
Figure B.3: (Continuation) Detailed comparison of the state estimation accuracy on the EuRoC dataset while using structureless factors (S), structureless and projection factors (P), and our proposed approach using structureless, projection and regularity factors (R).

Appendix C

RPE translation errors

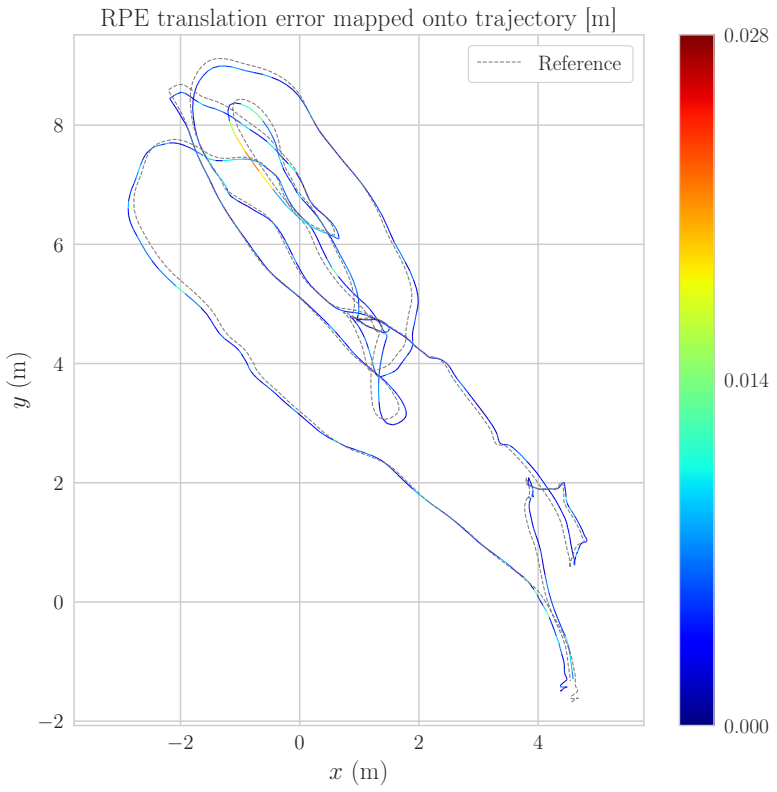


(a) S+P

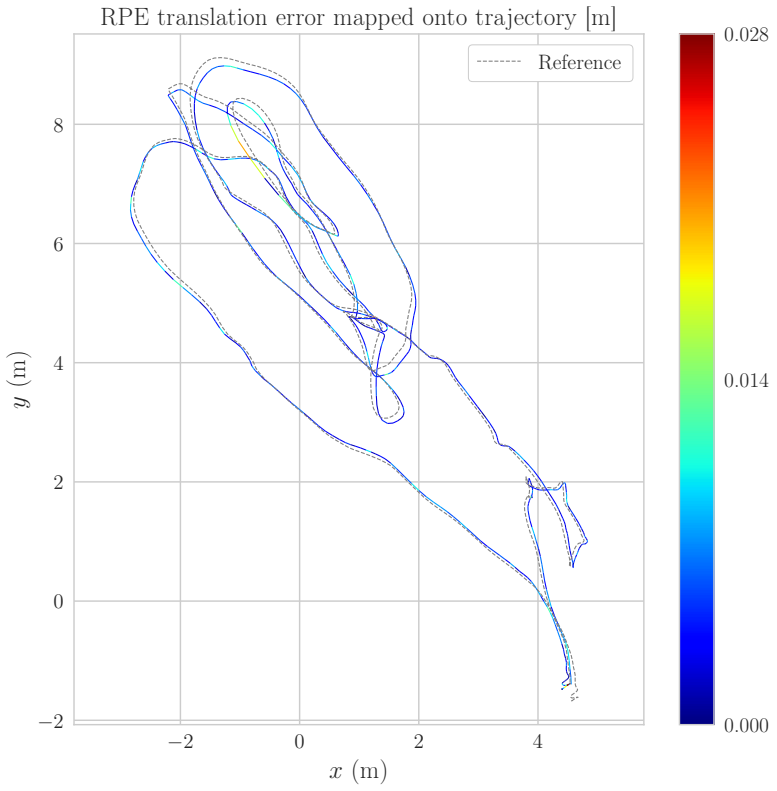


(b) S+P+R

Figure C.1: Dataset `MH_01_easy`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

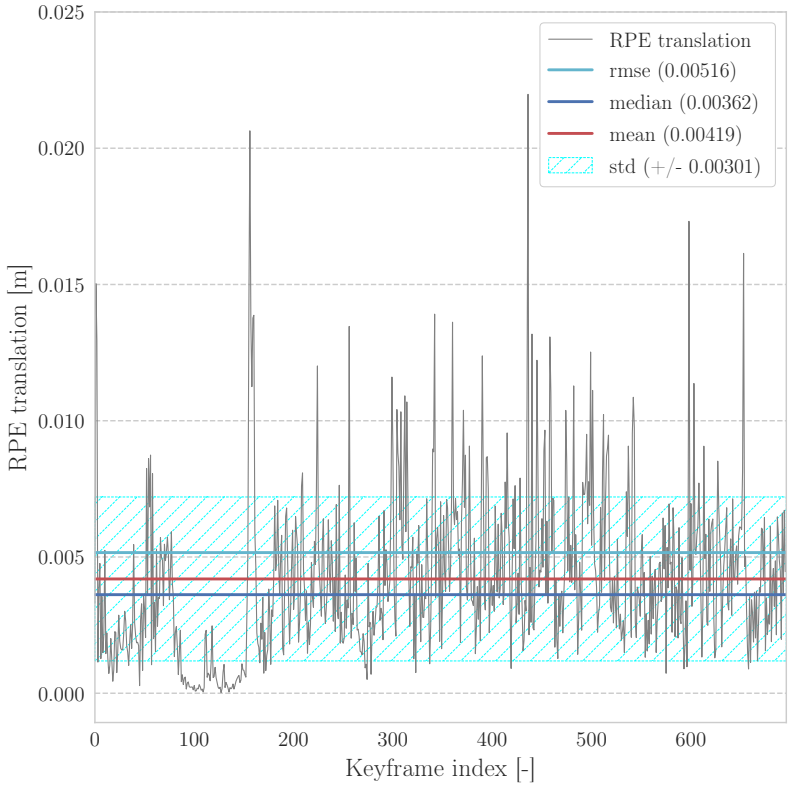


(a) RPE translation for S+P

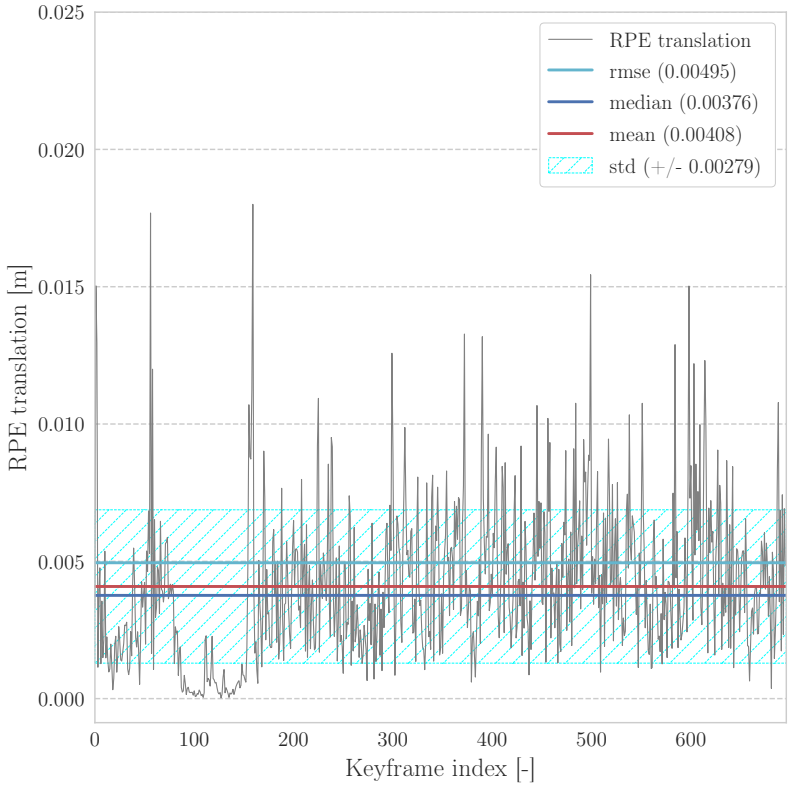


(b) RPE translation for S+P+R

Figure C.2: Dataset `MH_01_easy`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

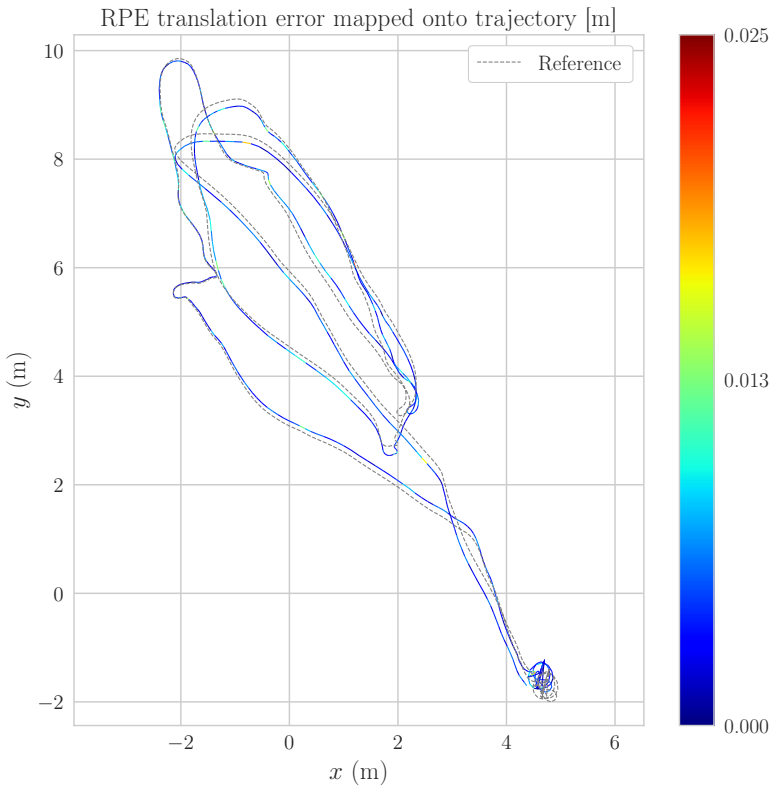


(a) S+P

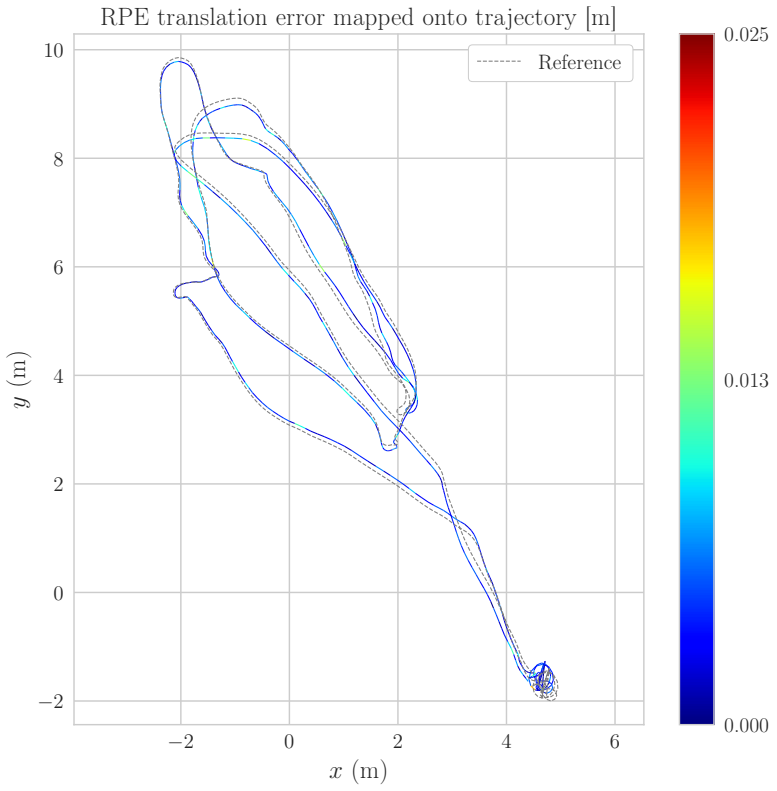


(b) S+P+R

Figure C.3: Dataset `MH_02_easy`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

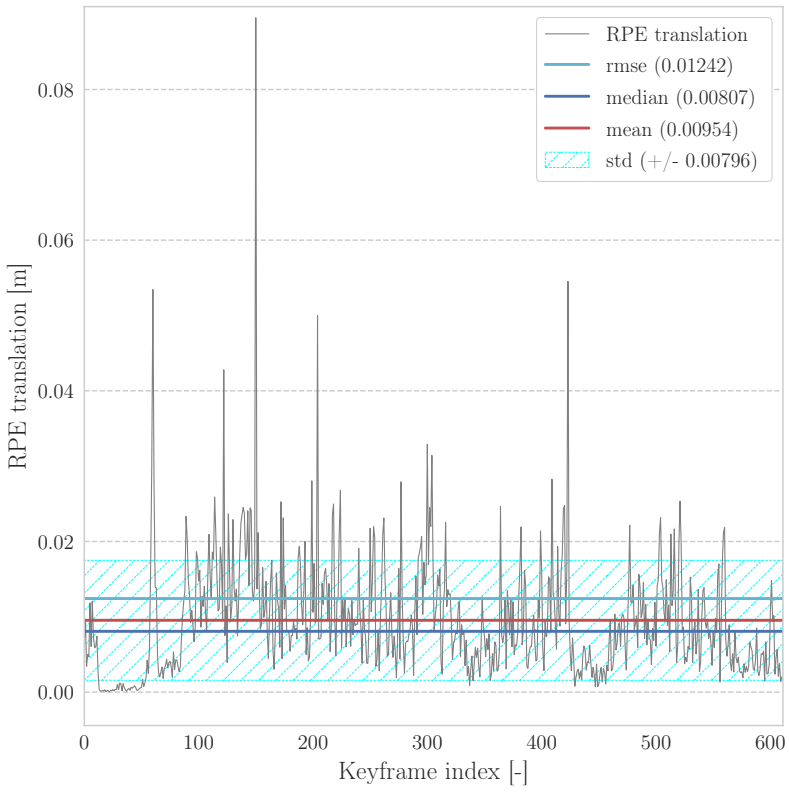


(a) RPE translation for S+P

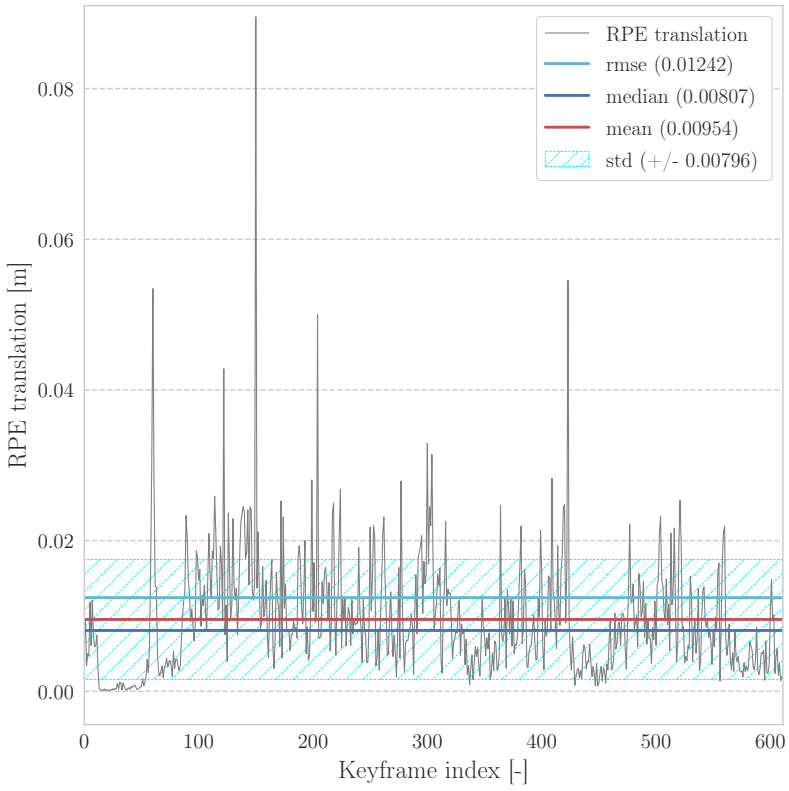


(b) RPE translation for S+P+R

Figure C.4: Dataset `MH_02_easy`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

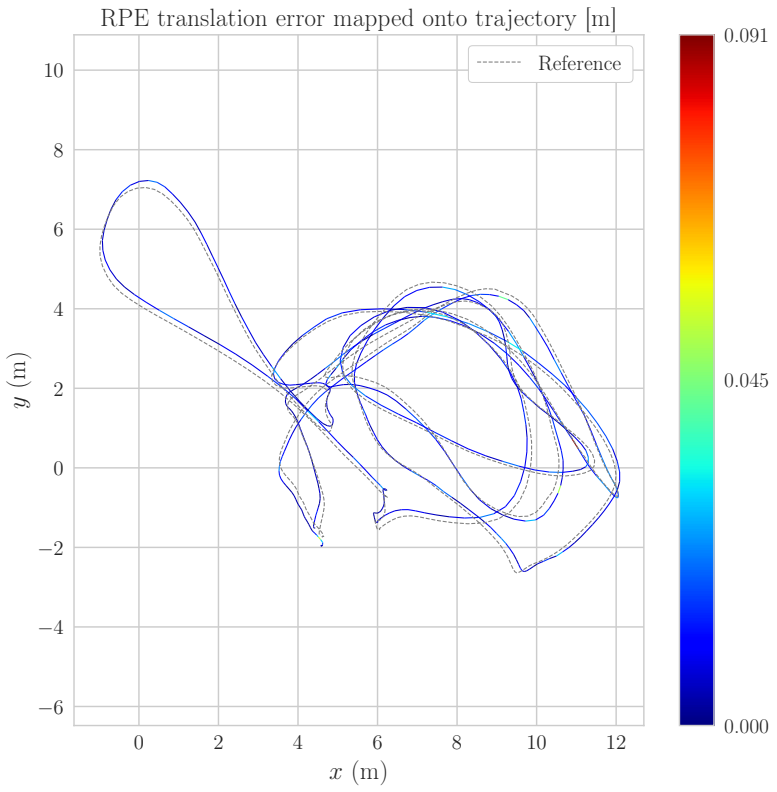


(a) S+P

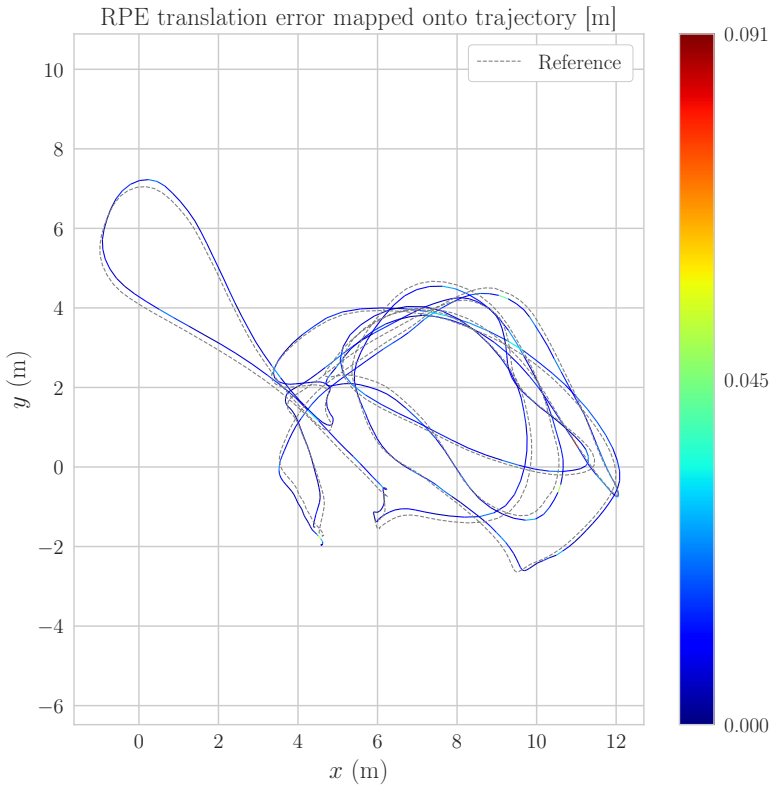


(b) S+P+R

Figure C.5: Dataset `MH_03_medium`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

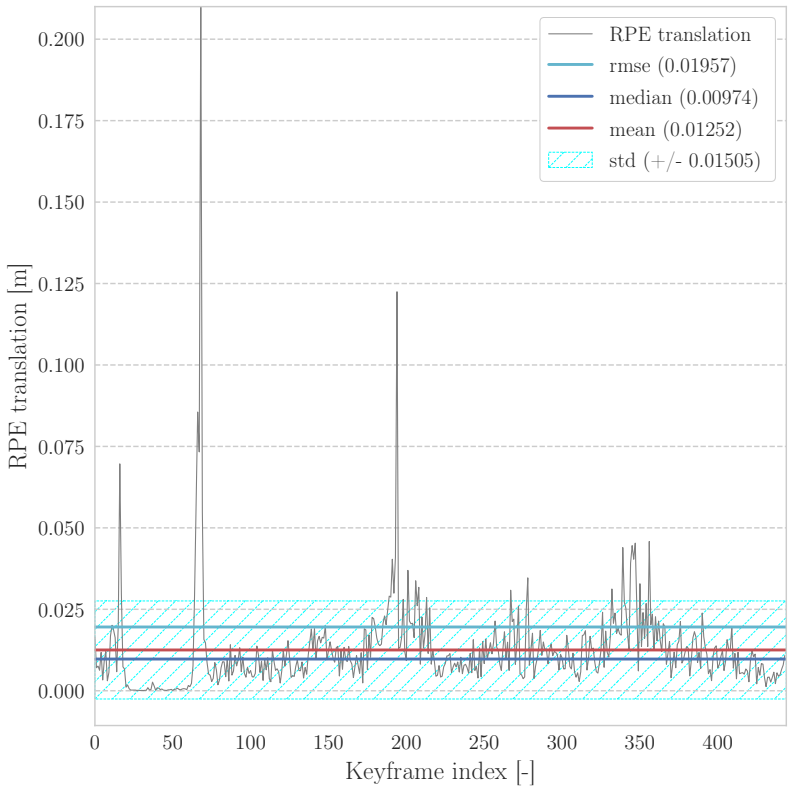


(a) RPE translation for S+P

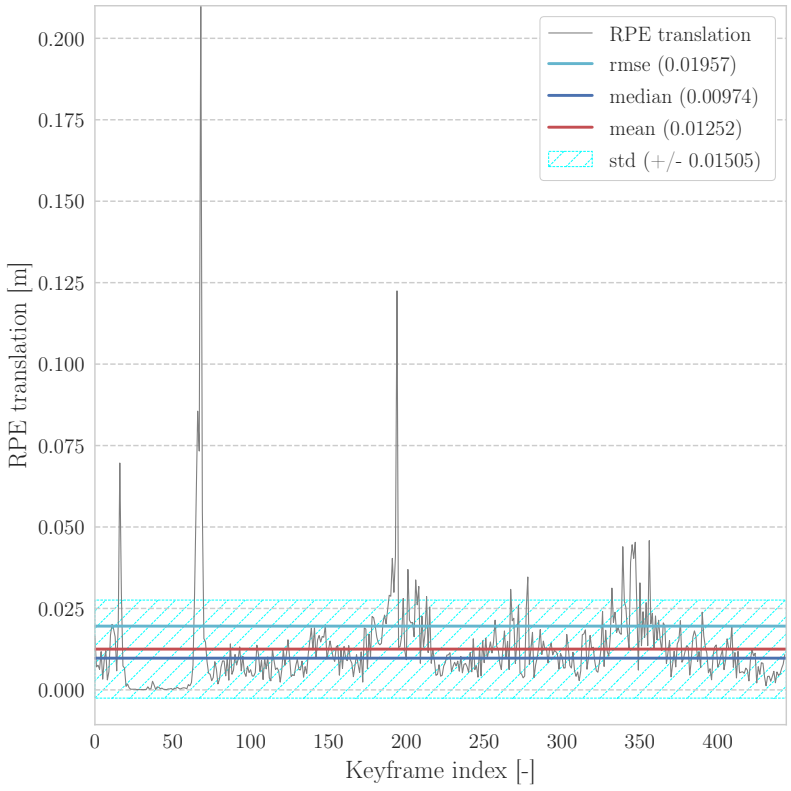


(b) RPE translation for S+P+R

Figure C.6: Dataset `MH_03_medium`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

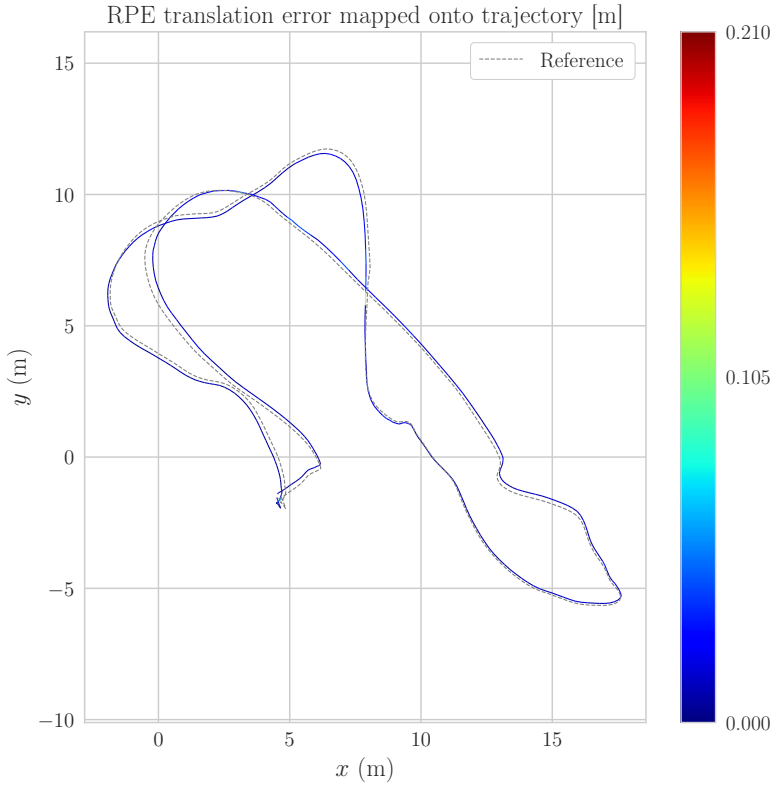


(a) S+P

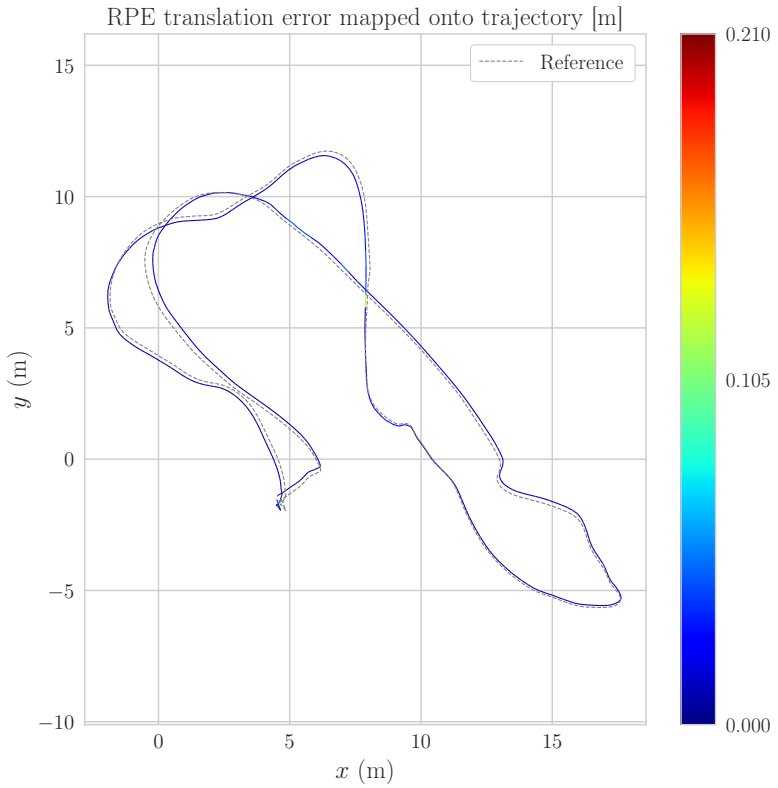


(b) S+P+R

Figure C.7: Dataset `MH_04_difficult`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

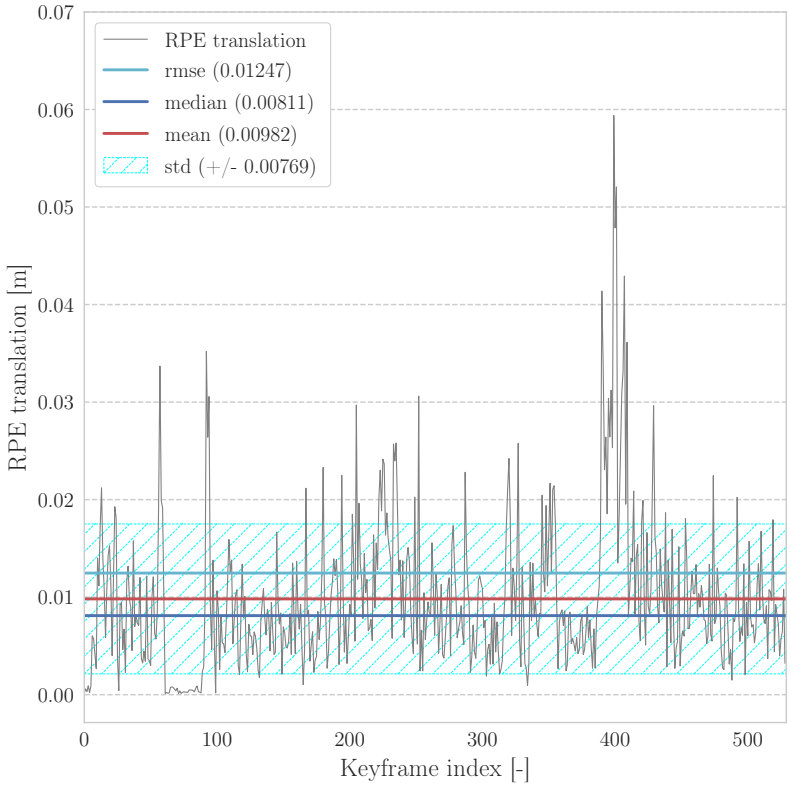


(a) RPE translation for S+P

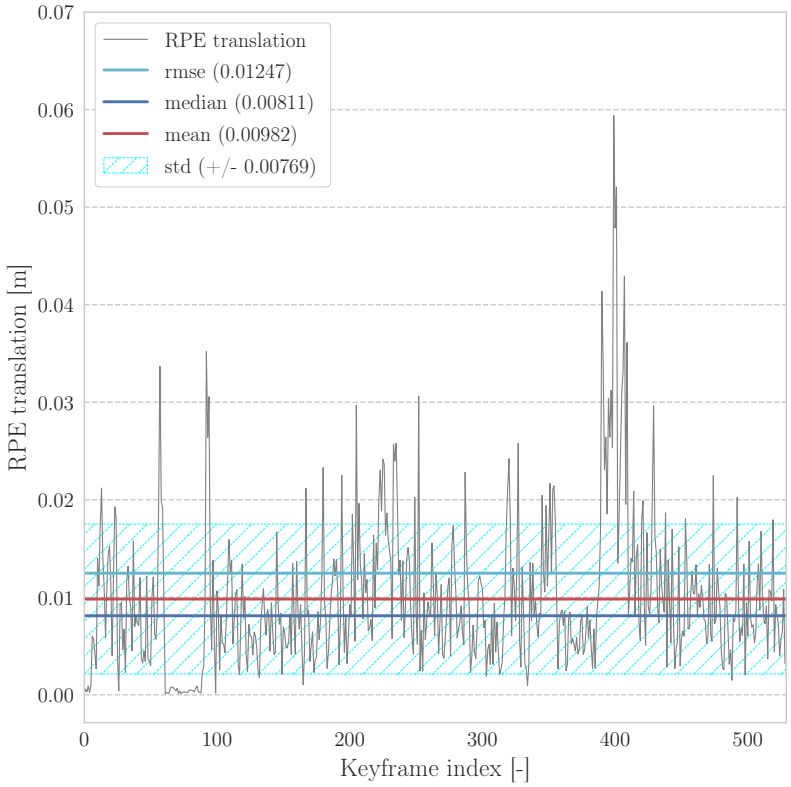


(b) RPE translation for S+P+R

Figure C.8: Dataset `MH_04_difficult`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

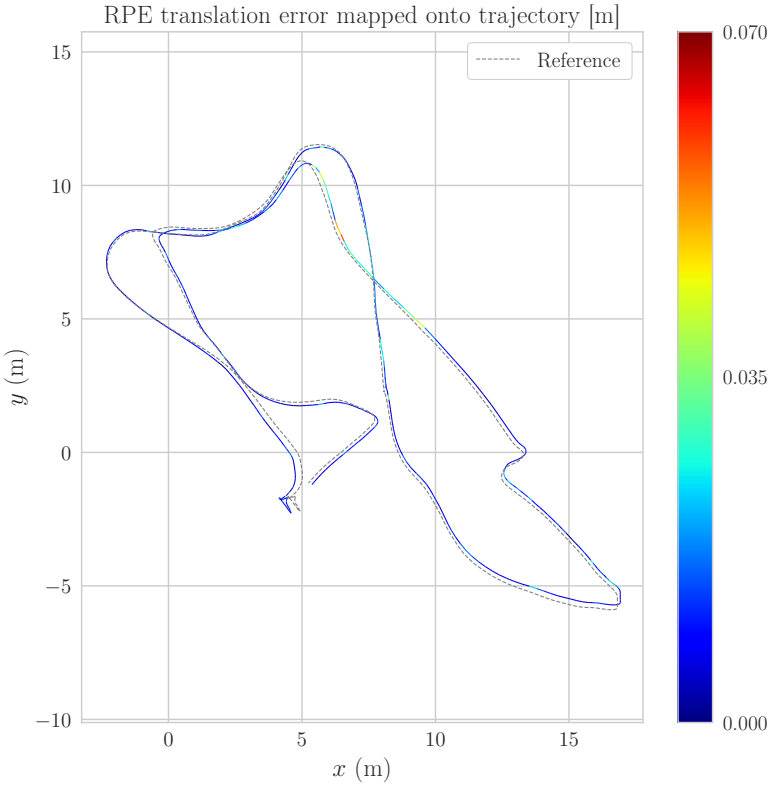


(a) S+P

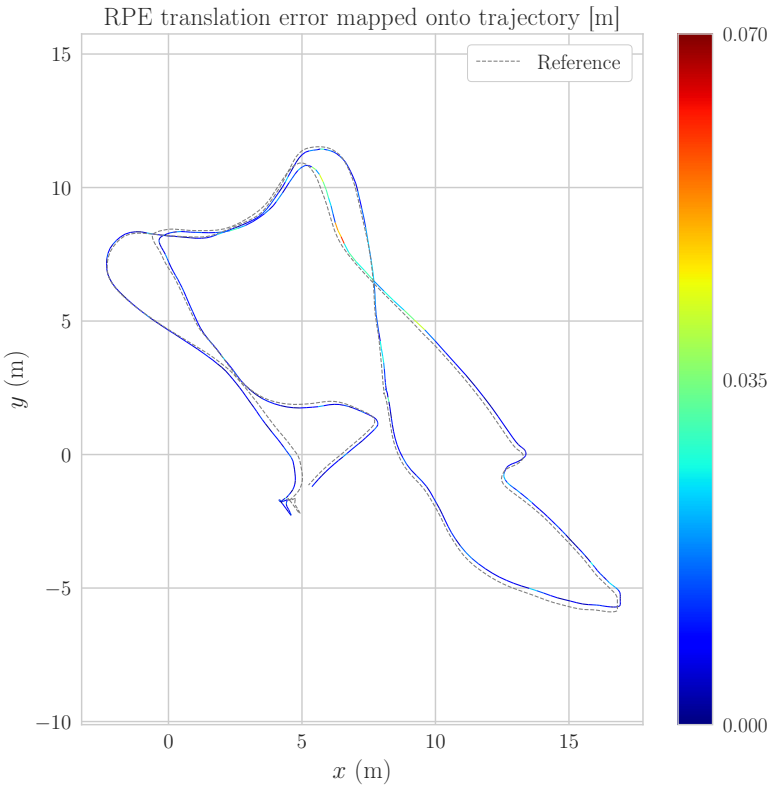


(b) S+P+R

Figure C.9: Dataset `MH_05_difficult`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

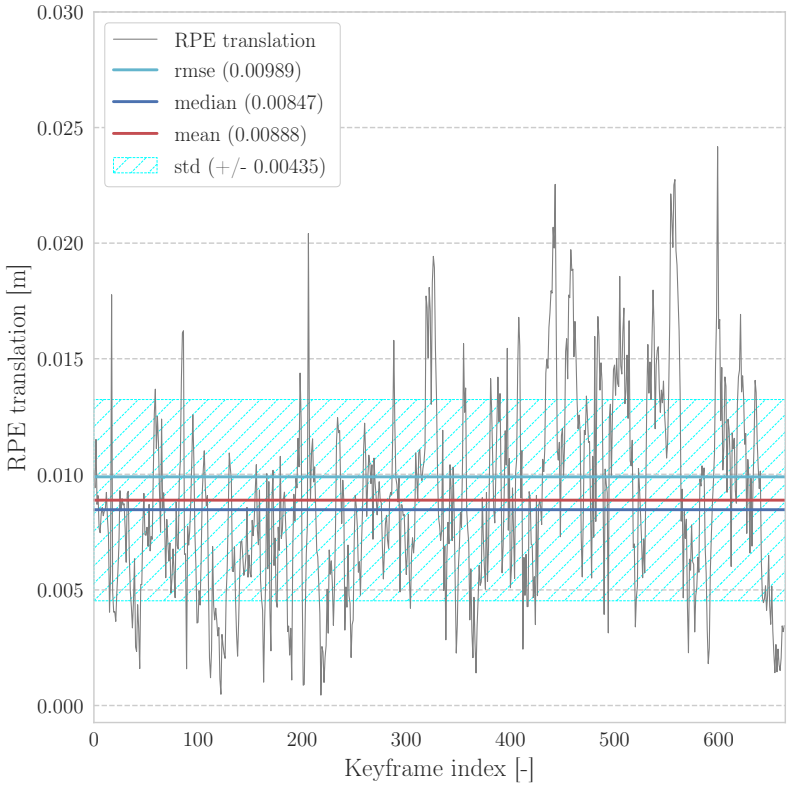


(a) RPE translation for S+P

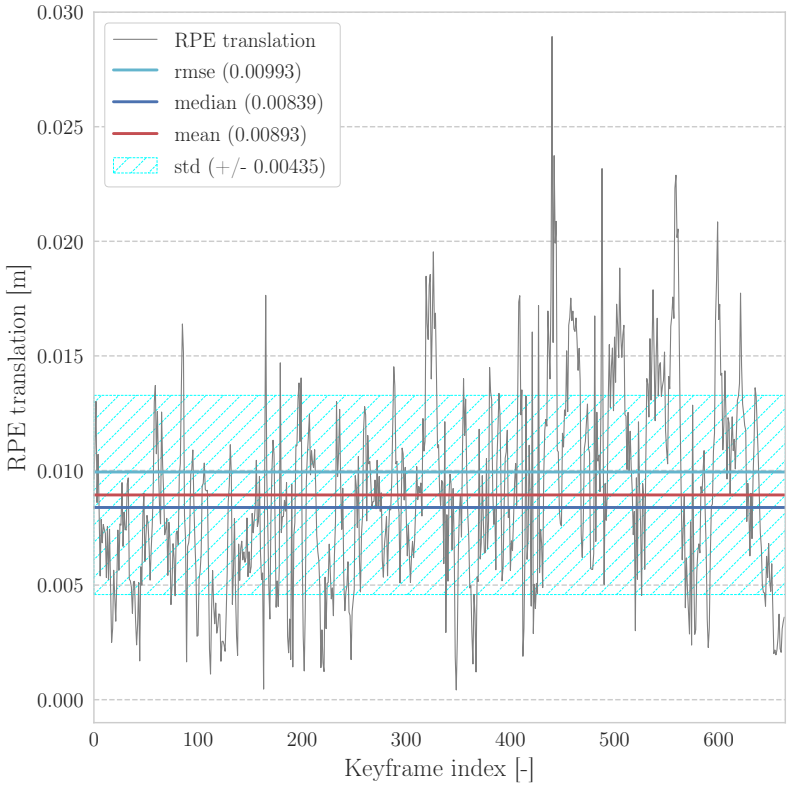


(b) RPE translation for S+P+R

Figure C.10: Dataset `MH_05_difficult`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

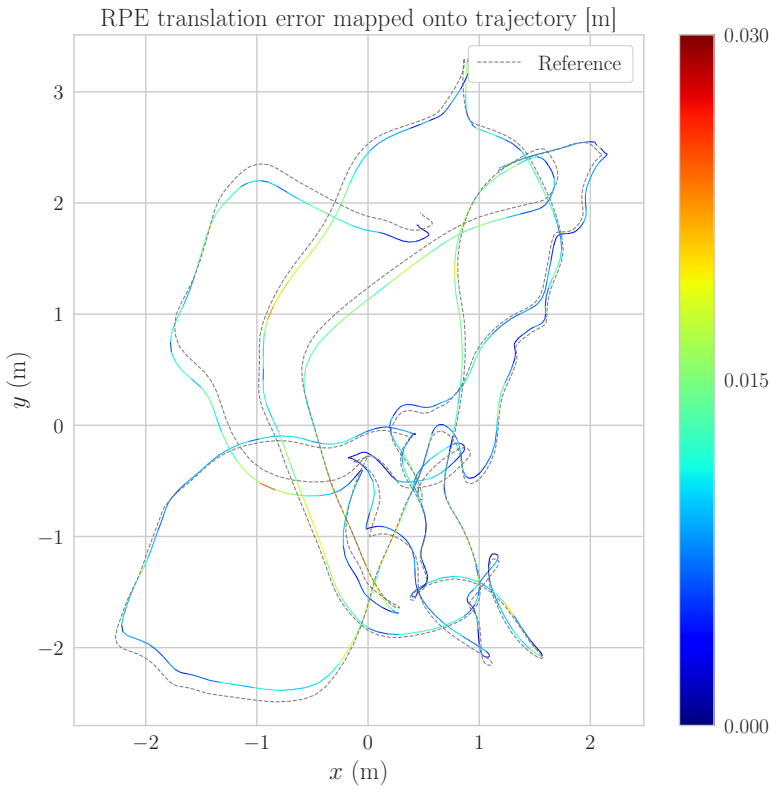


(a) S+P

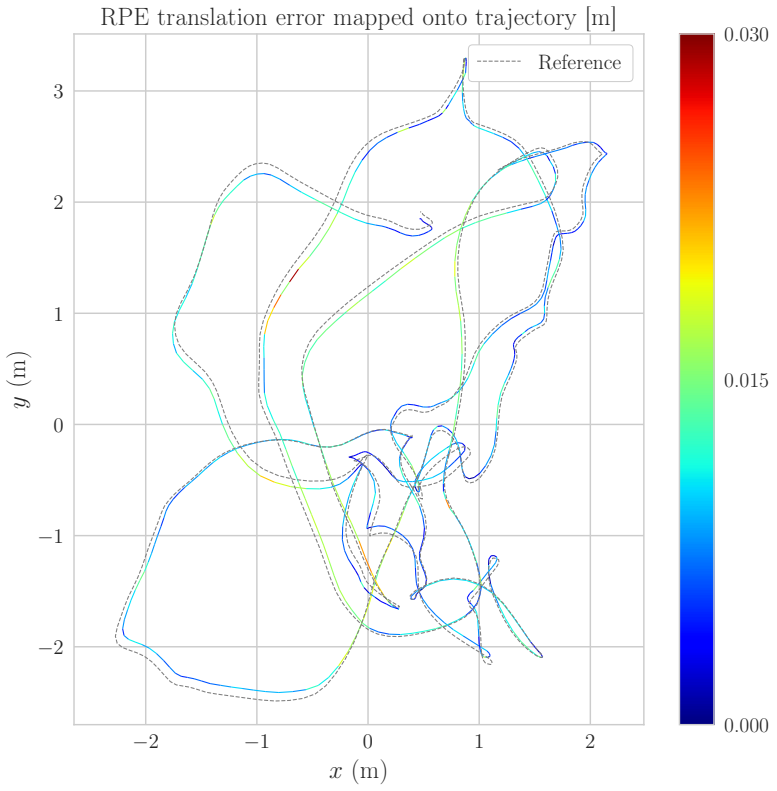


(b) S+P+R

Figure C.11: Dataset `V1_01_easy`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

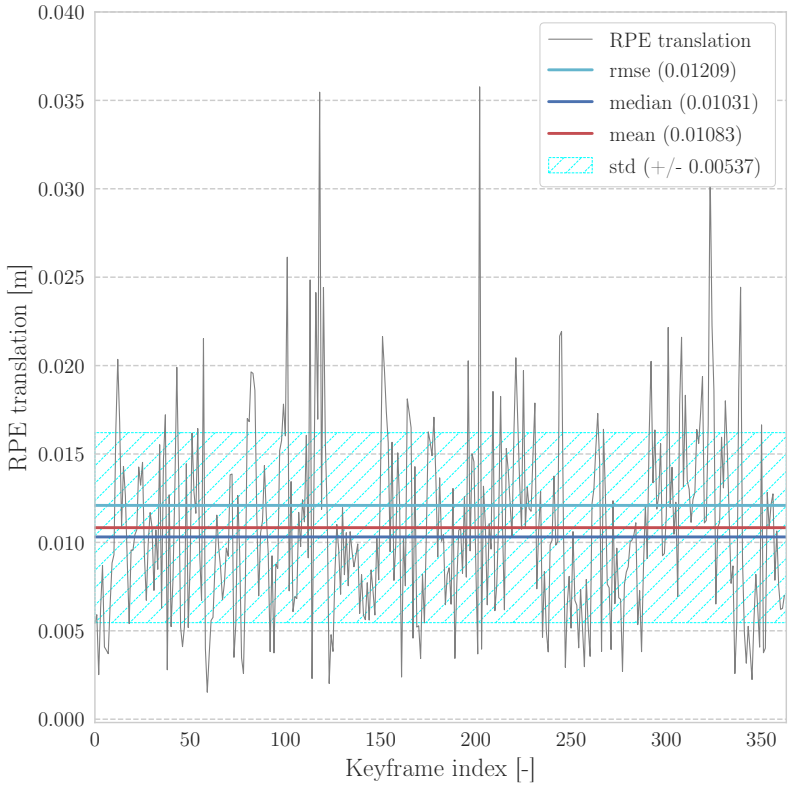


(a) RPE translation for S+P

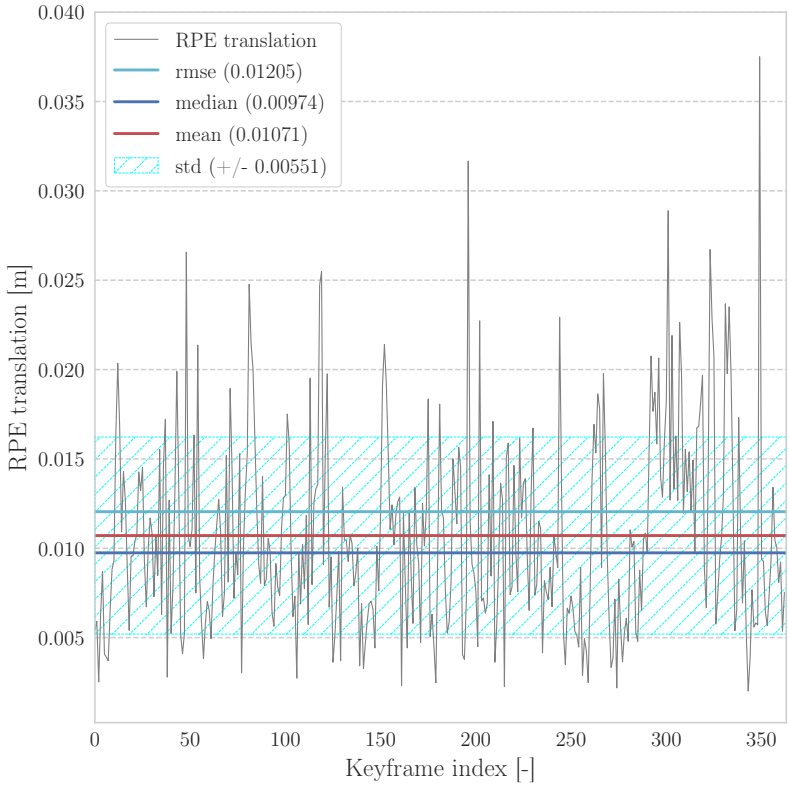


(b) RPE translation for S+P+R

Figure C.12: Dataset `V1_01_easy`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

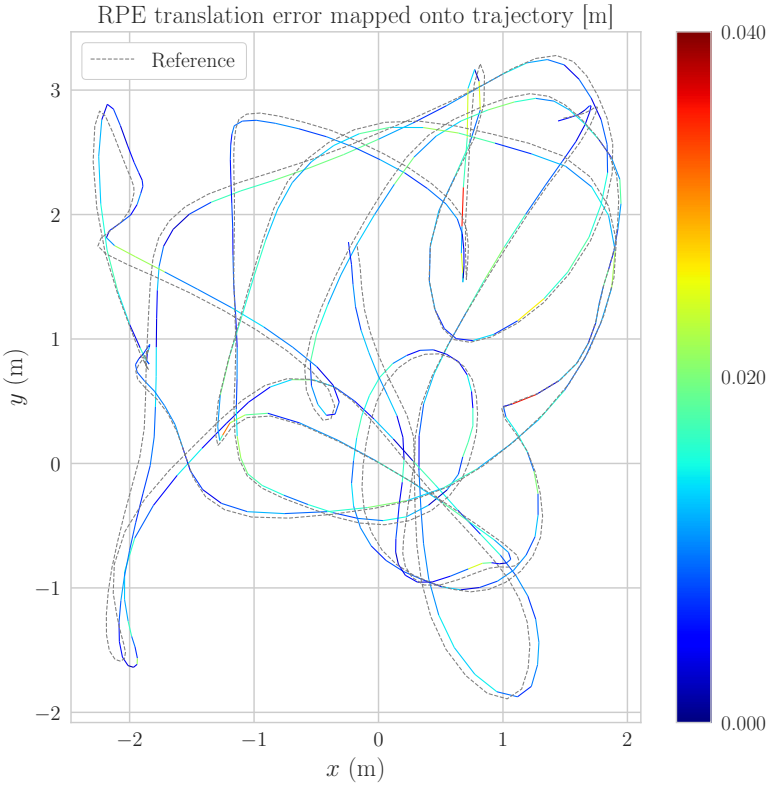


(a) S+P

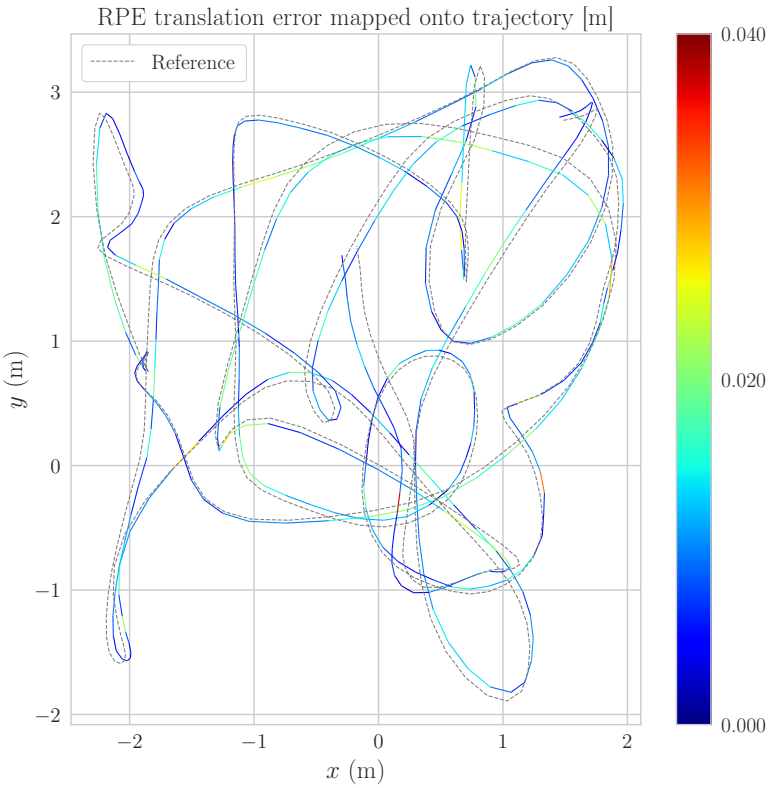


(b) S+P+R

Figure C.13: Dataset `V1_02_medium`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

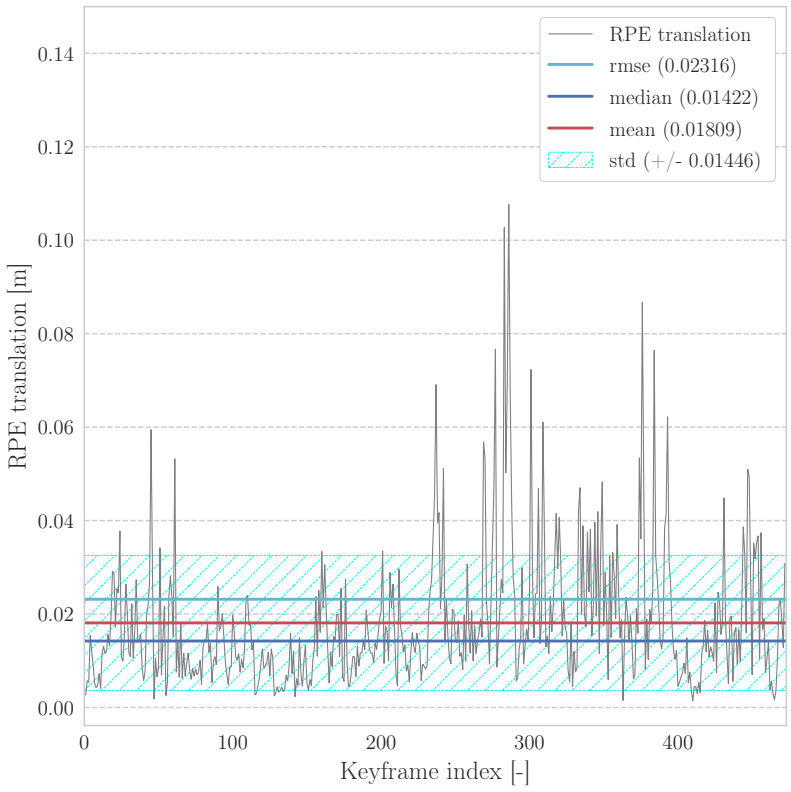


(a) RPE translation for S+P

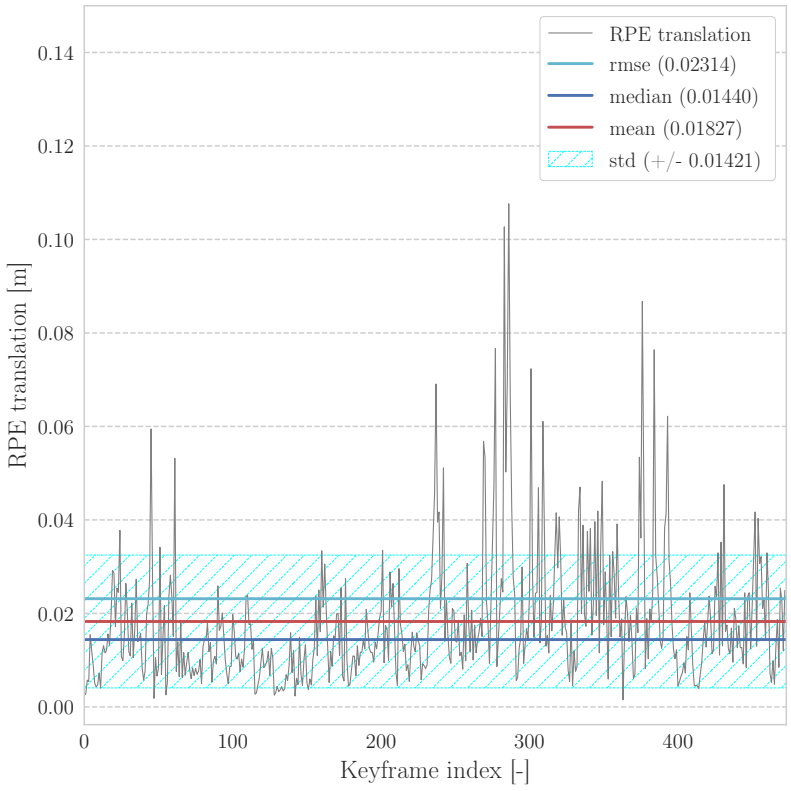


(b) RPE translation for S+P+R

Figure C.14: Dataset `V1_02_medium`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

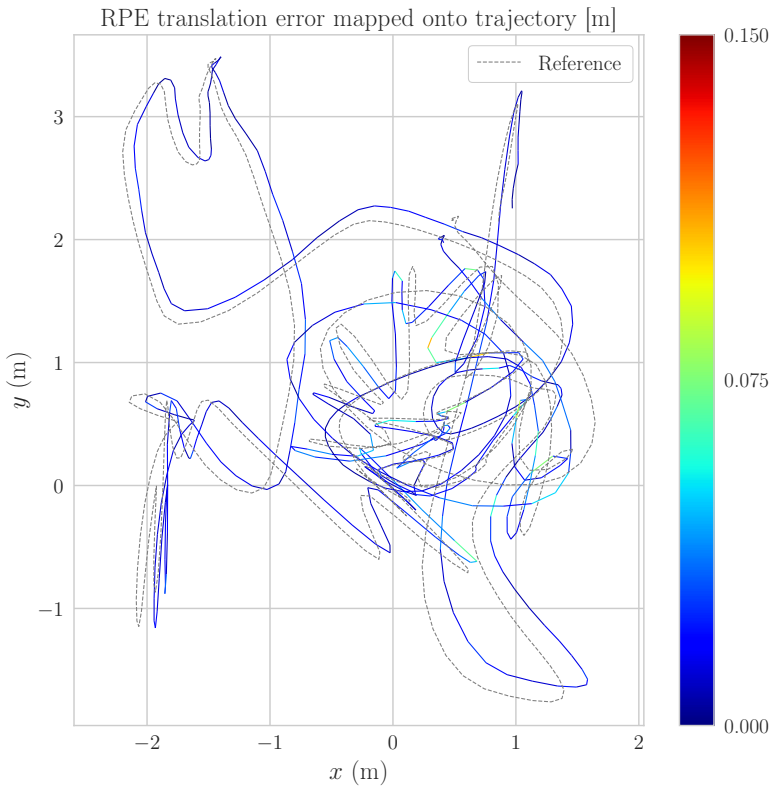


(a) S+P

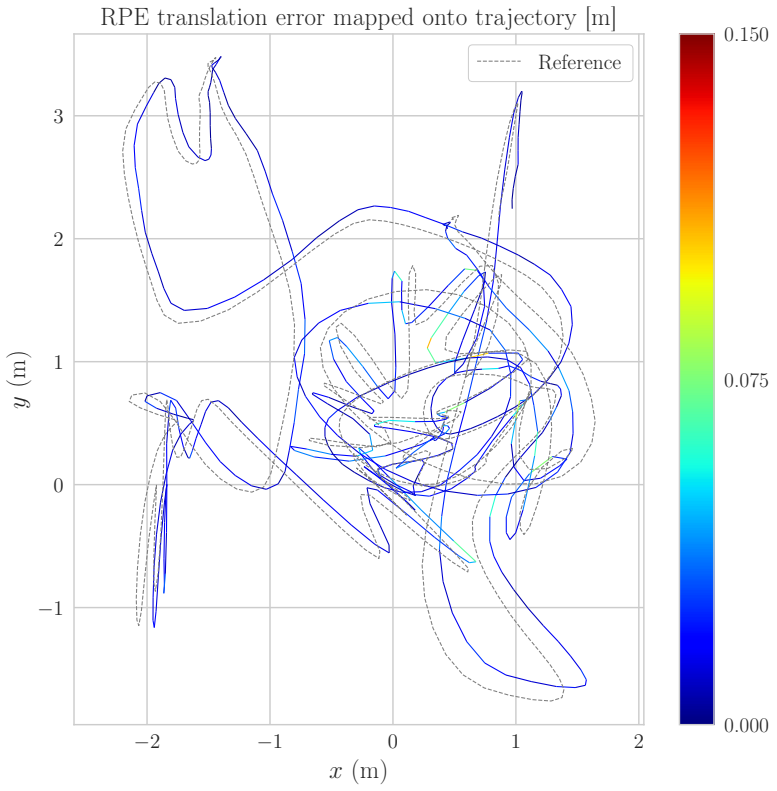


(b) S+P+R

Figure C.15: Dataset `V1_03_difficult`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

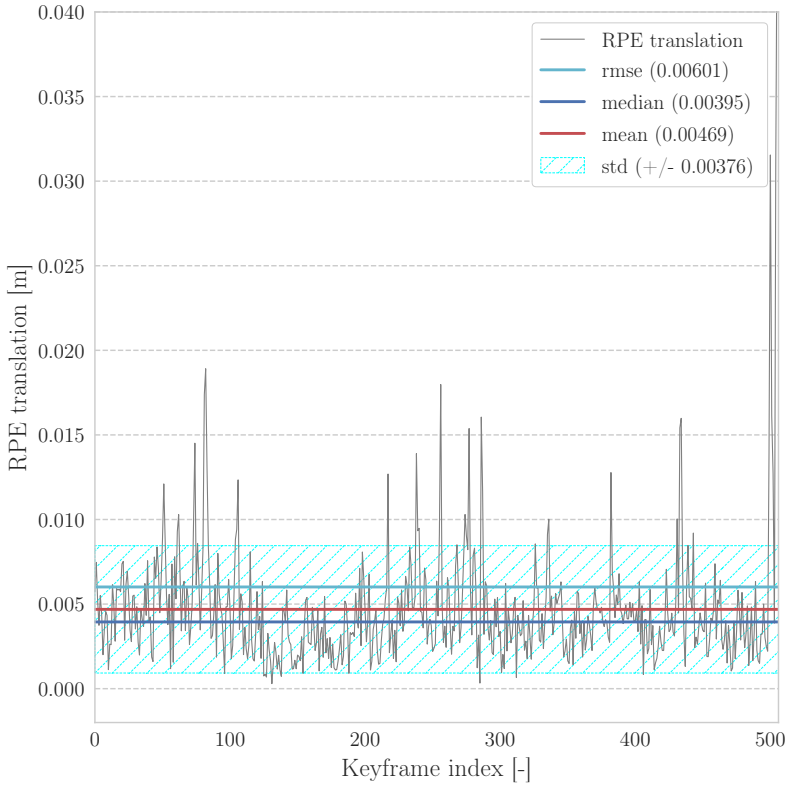


(a) RPE translation for S+P

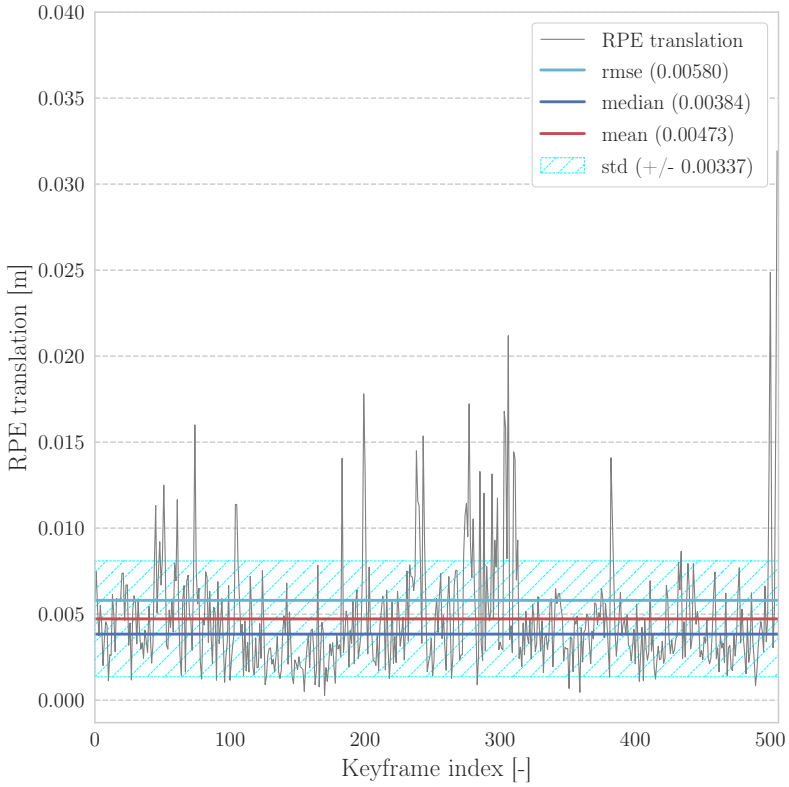


(b) RPE translation for S+P+R

Figure C.16: Dataset `V1_03_difficult`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

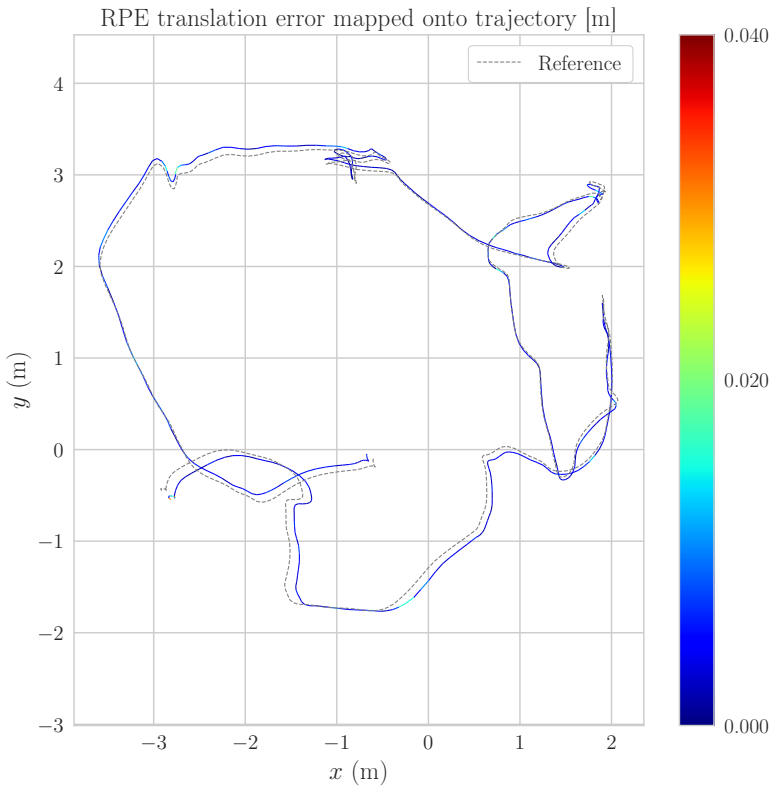


(a) S+P

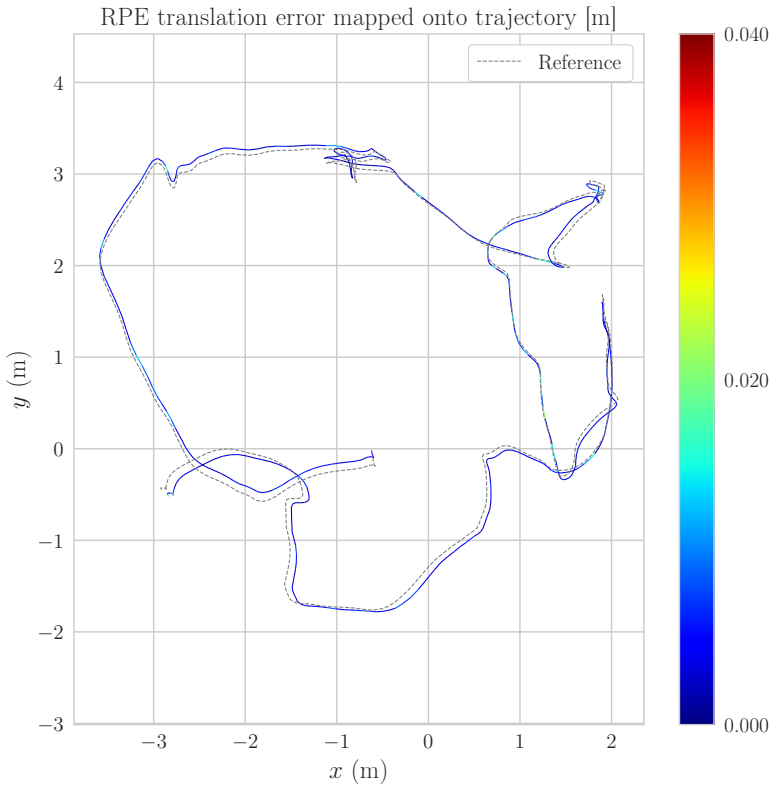


(b) S+P+R

Figure C.17: Dataset `V2_01_easy`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

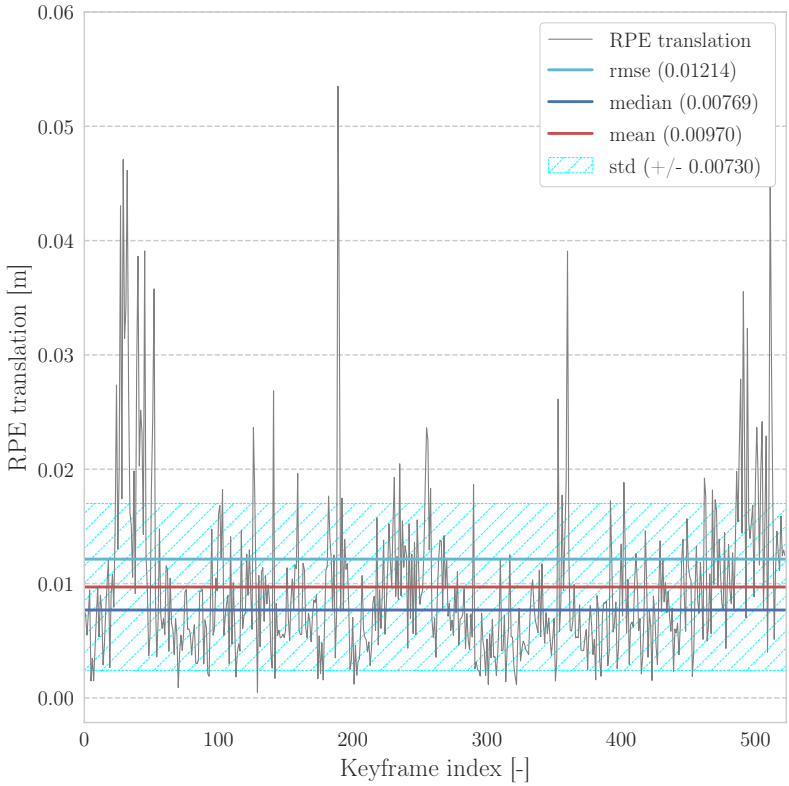


(a) RPE translation for S+P

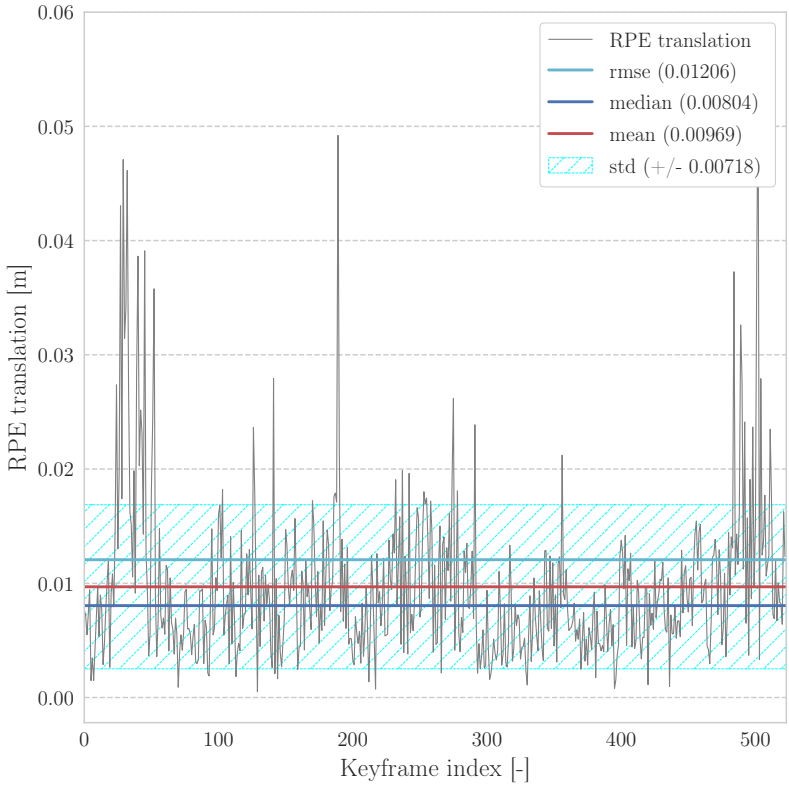


(b) RPE translation for S+P+R

Figure C.18: Dataset `V2_01_easy`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

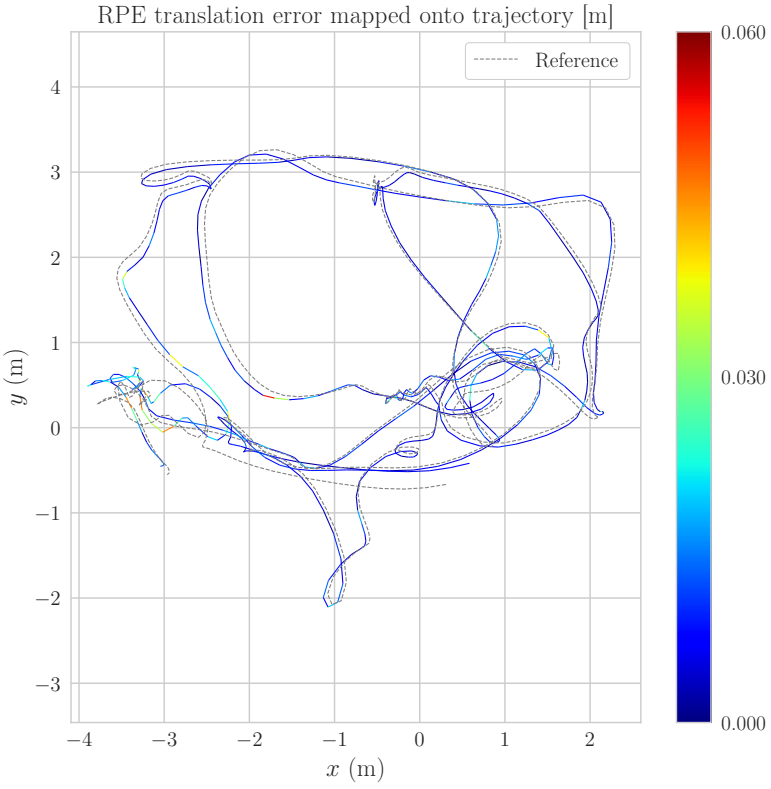


(a) S+P

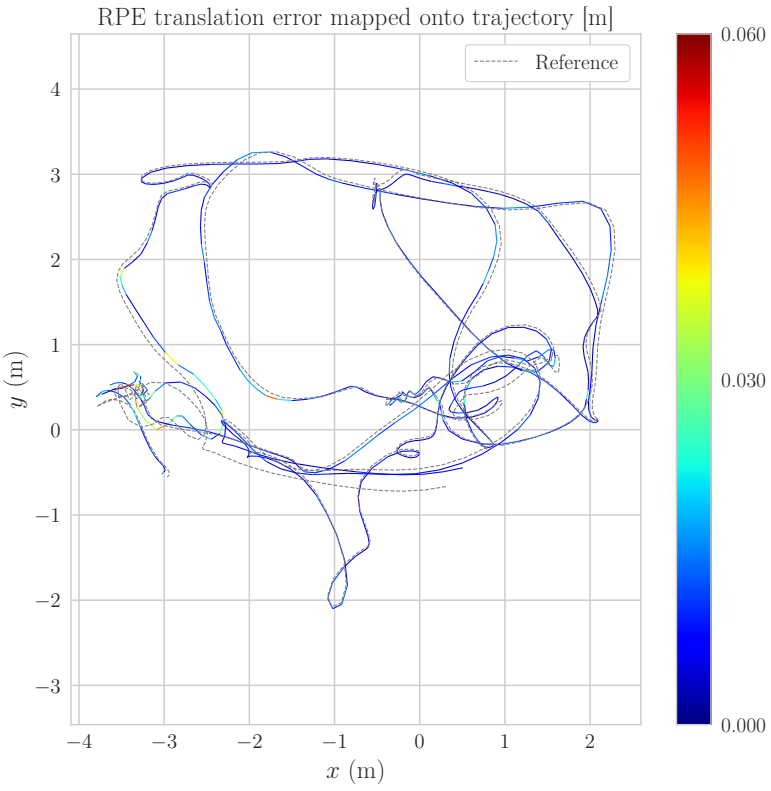


(b) S+P+R

Figure C.19: Dataset `V2_02_medium`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

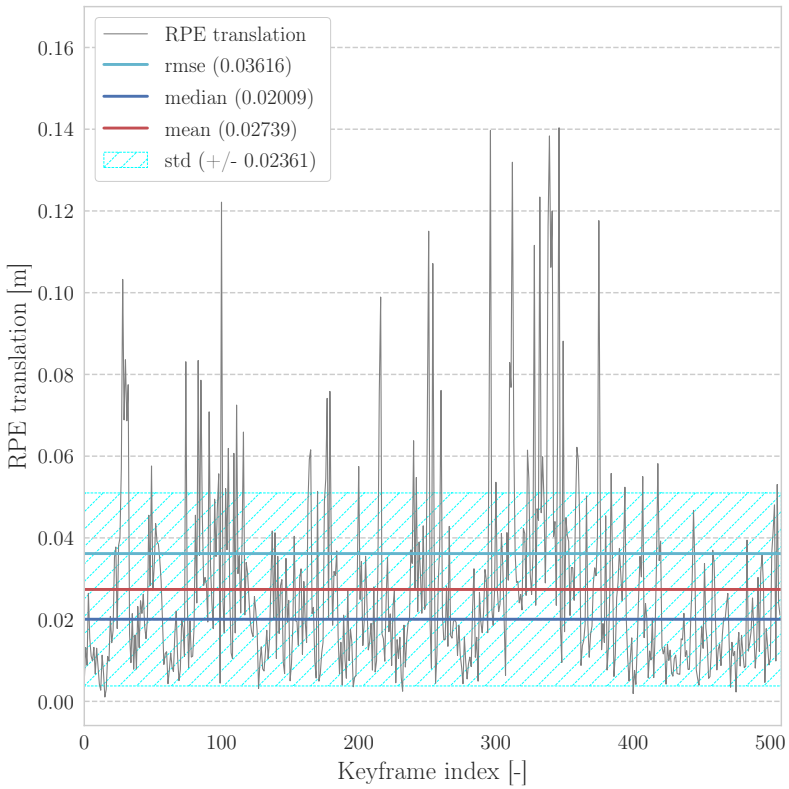


(a) RPE translation for S+P

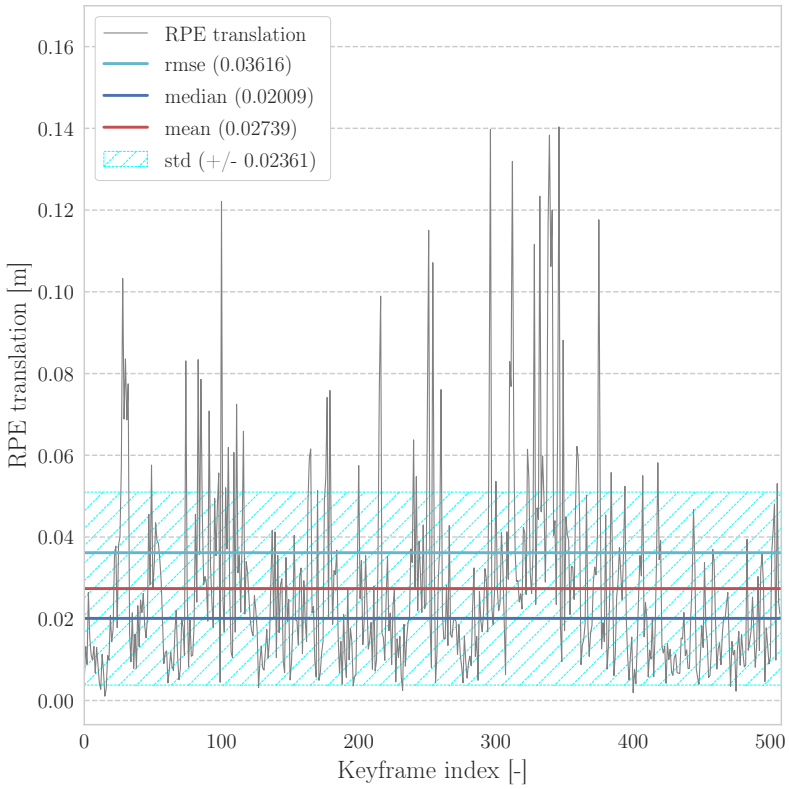


(b) RPE translation for S+P+R

Figure C.20: Dataset `V2_02_medium`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

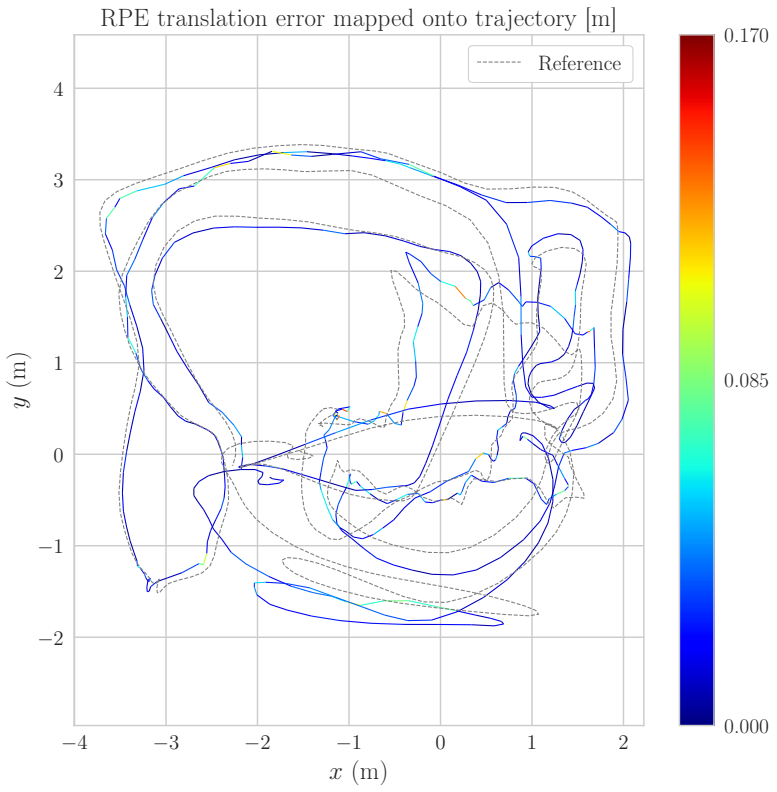


(a) S+P

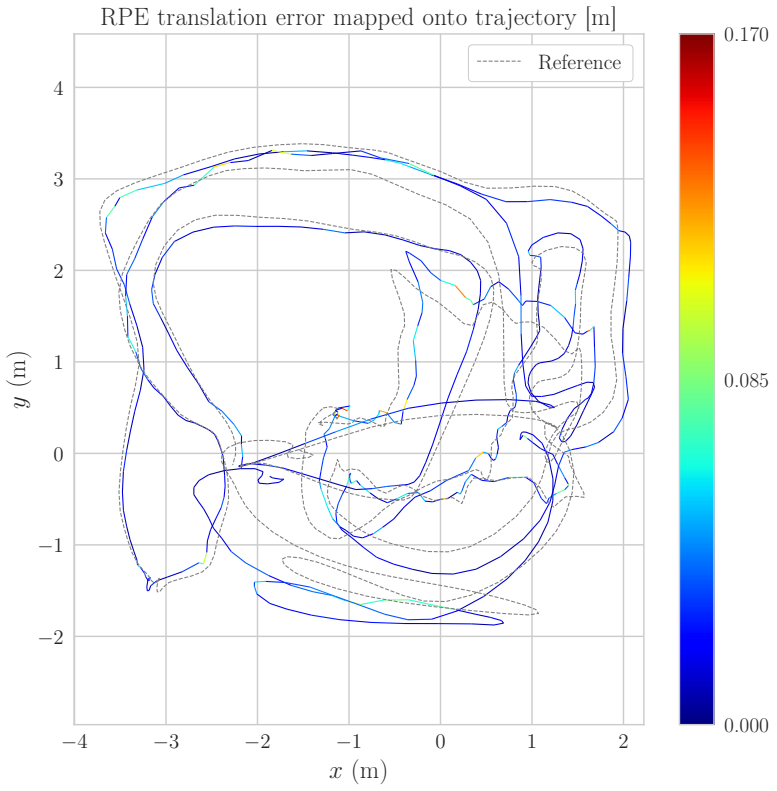


(b) S+P+R

Figure C.21: Dataset `V2_03_difficult`: RPE translation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.



(a) RPE translation for S+P

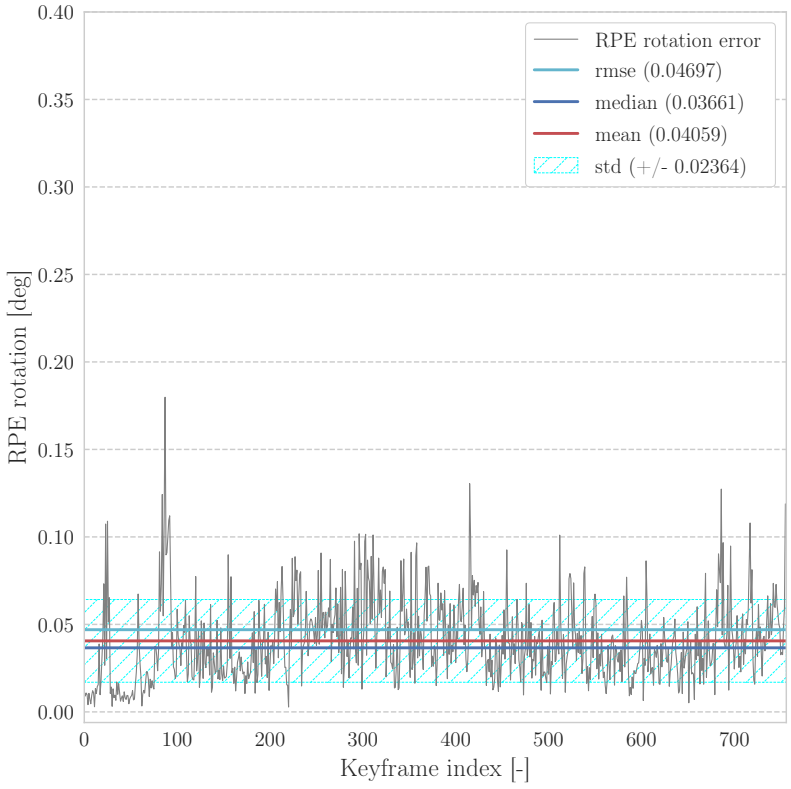


(b) RPE translation for S+P+R

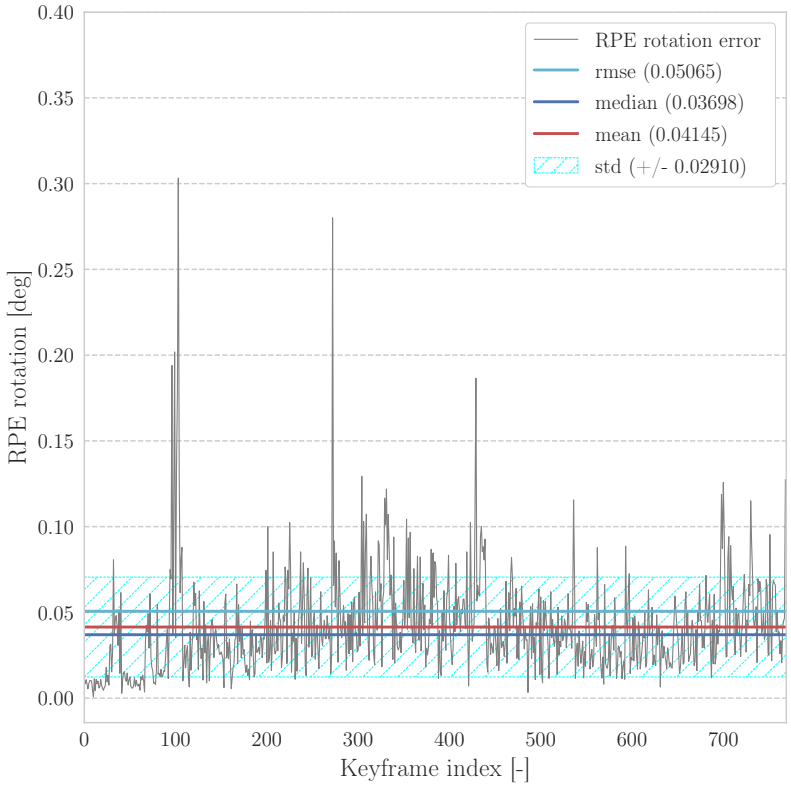
Figure C.22: Dataset `V2_03_difficult`: RPE translation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

Appendix D

RPE rotation errors

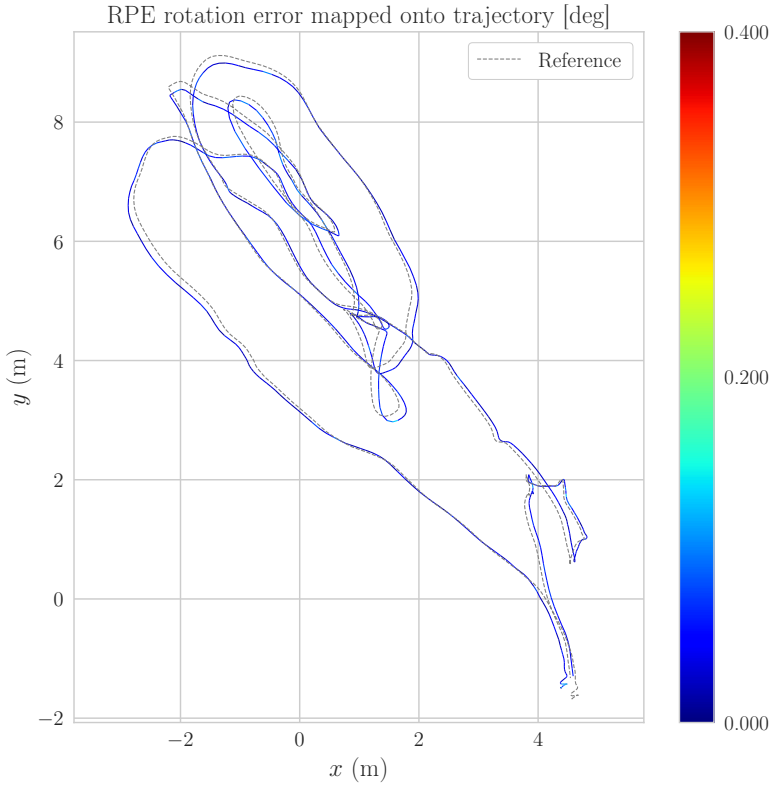


(a) S+P

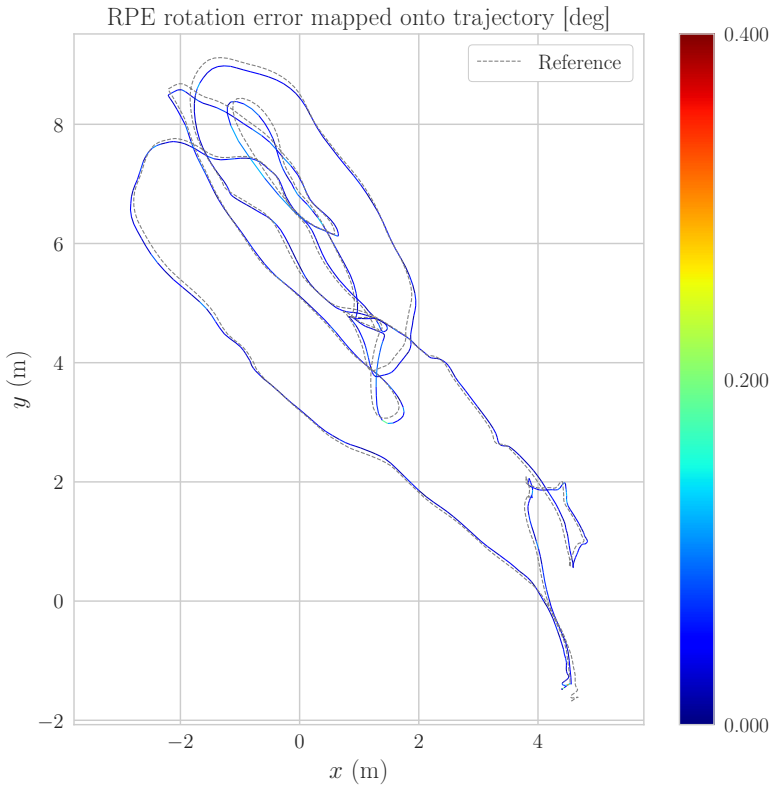


(b) S+P+R

Figure D.1: Dataset `MH_01_easy`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

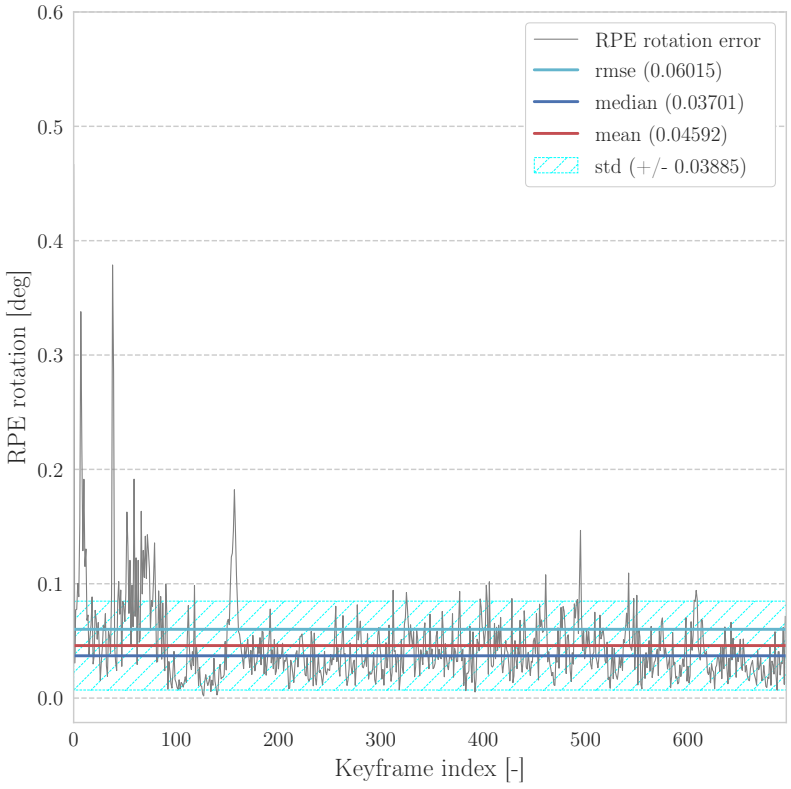


(a) RPE rotation for S+P

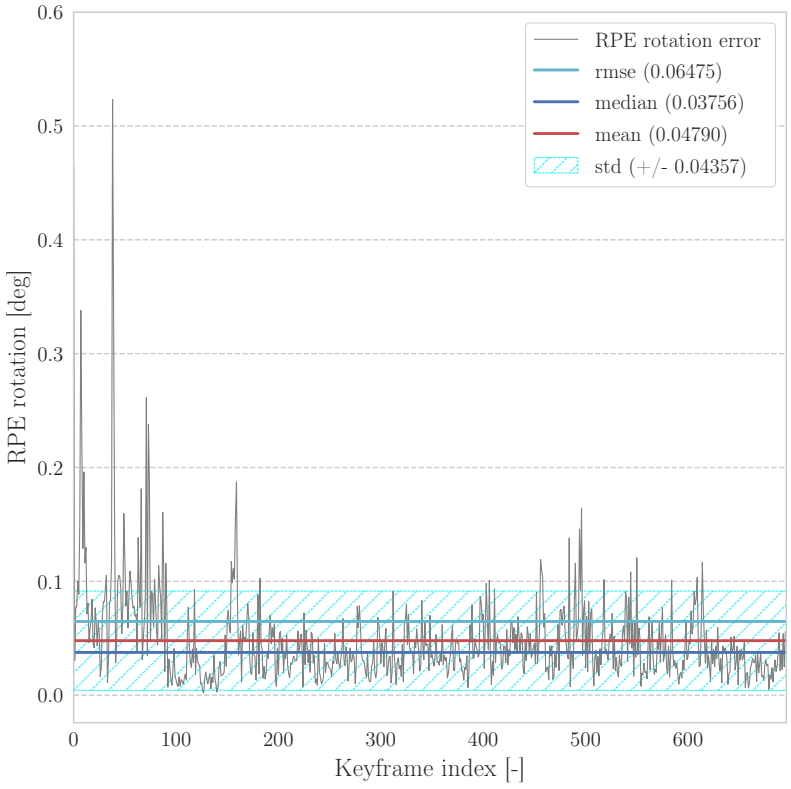


(b) RPE rotation for S+P+R

Figure D.2: Dataset `MH_01_easy`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

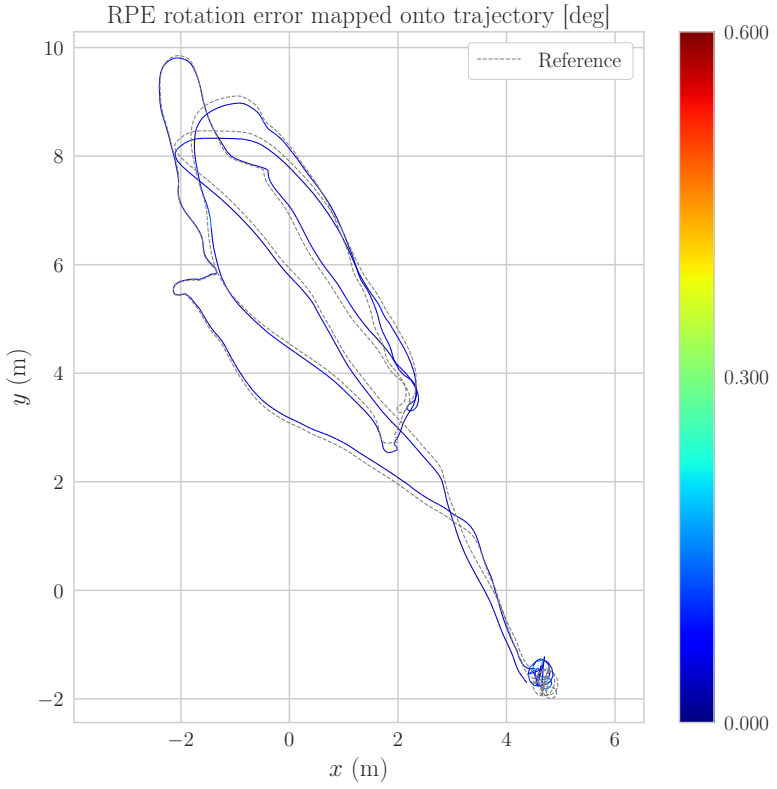


(a) S+P

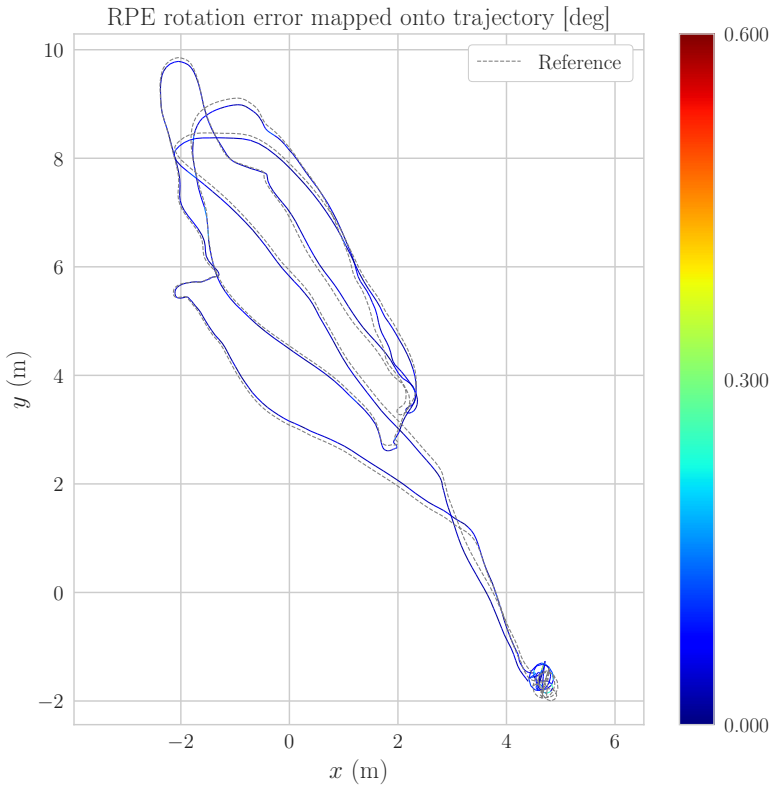


(b) S+P+R

Figure D.3: Dataset `MH_02_easy`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

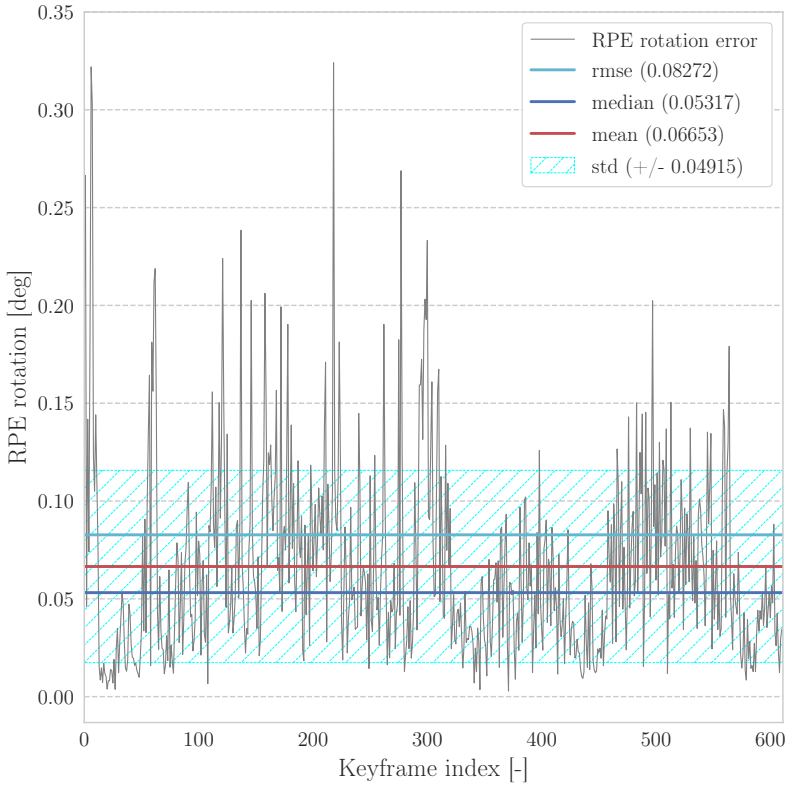


(a) RPE rotation for S+P

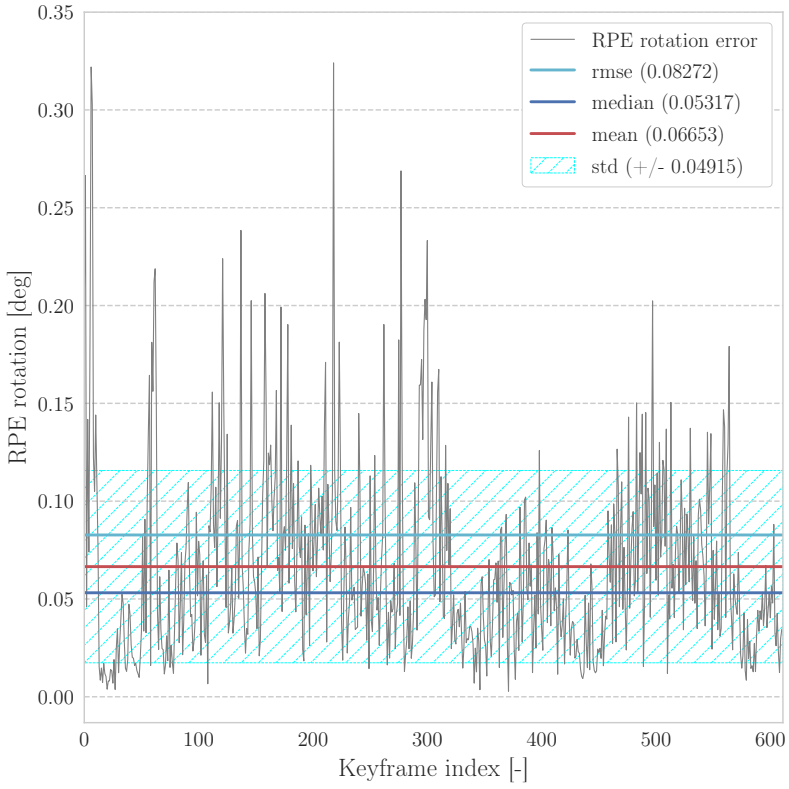


(b) RPE rotation for S+P+R

Figure D.4: Dataset `MH_02_easy`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

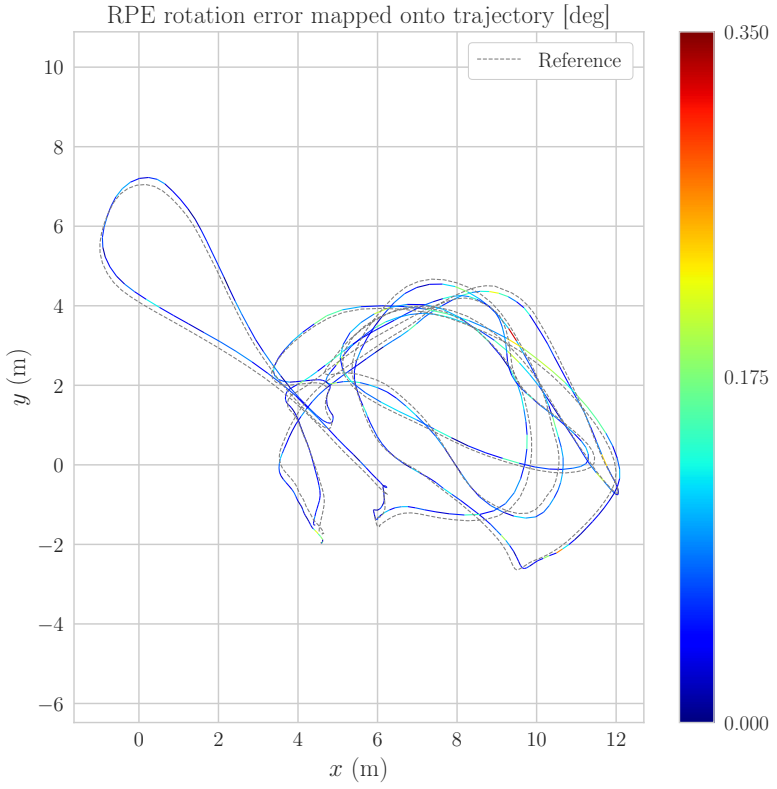


(a) S+P

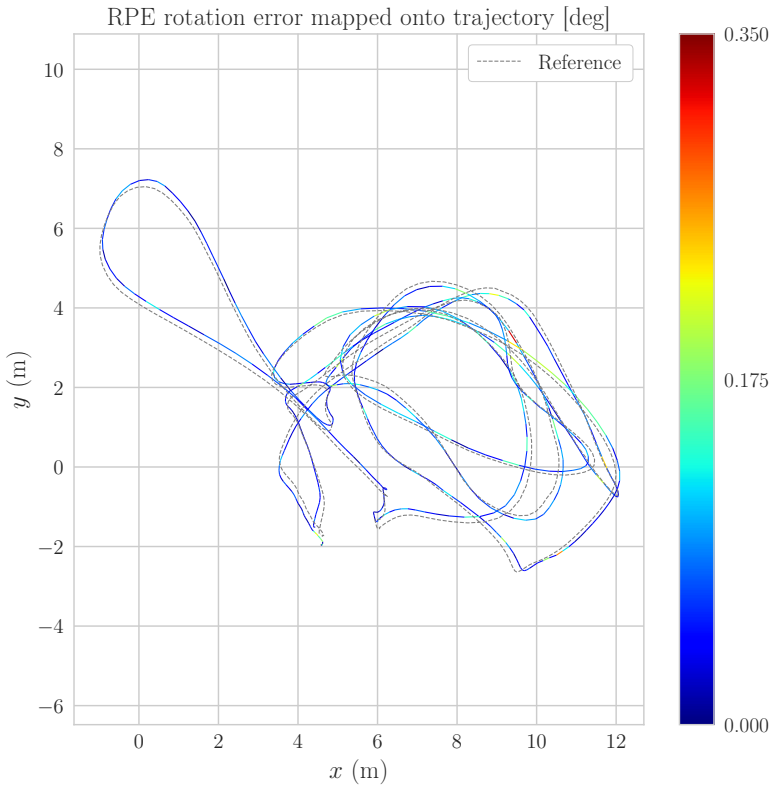


(b) S+P+R

Figure D.5: Dataset `MH_03_medium`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

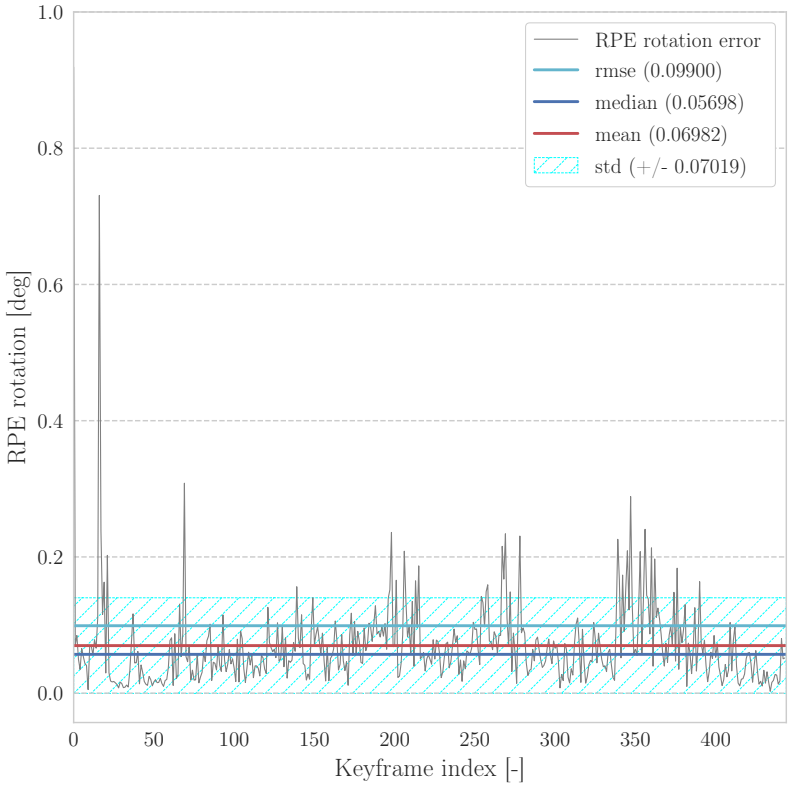


(a) RPE rotation for S+P

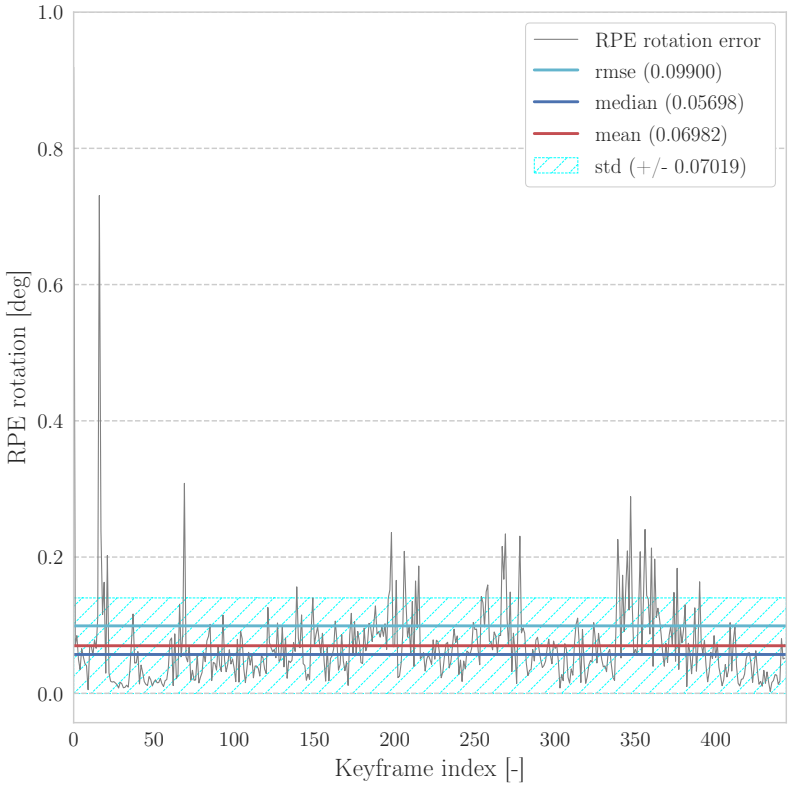


(b) RPE rotation for S+P+R

Figure D.6: Dataset `MH_03_medium`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

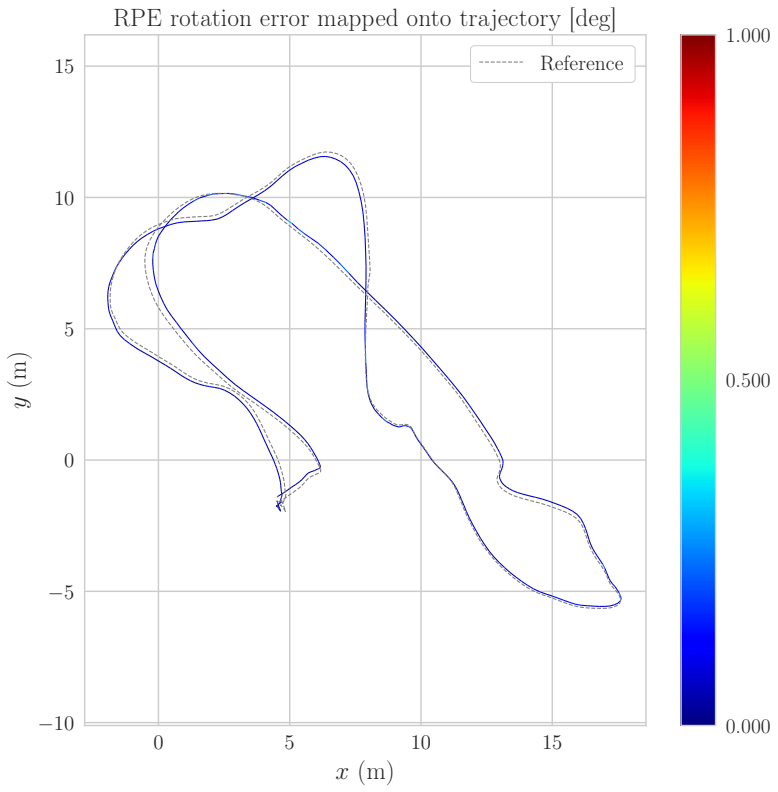


(a) S+P

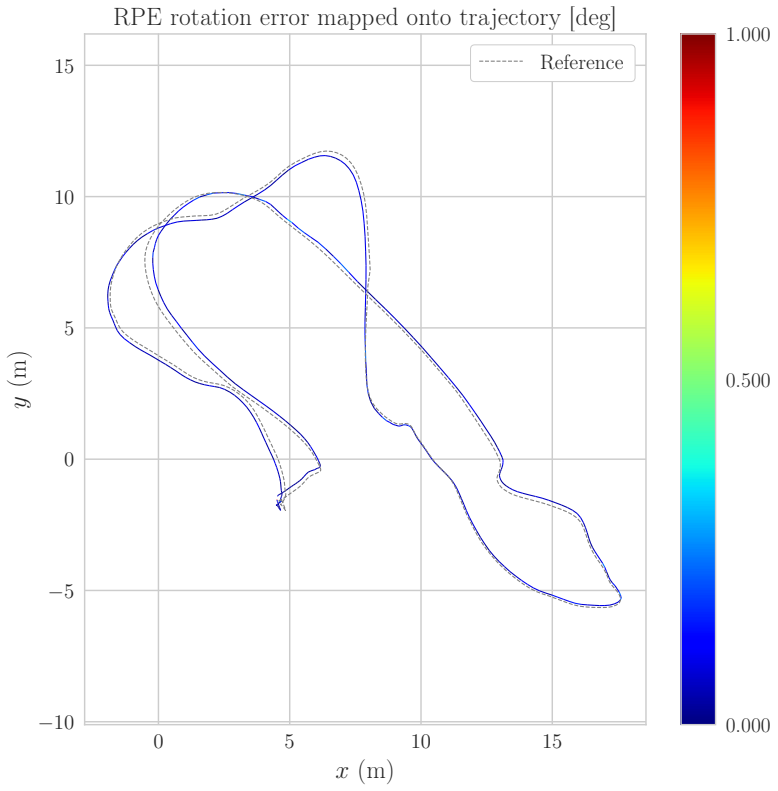


(b) S+P+R

Figure D.7: Dataset `MH_04_difficult`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

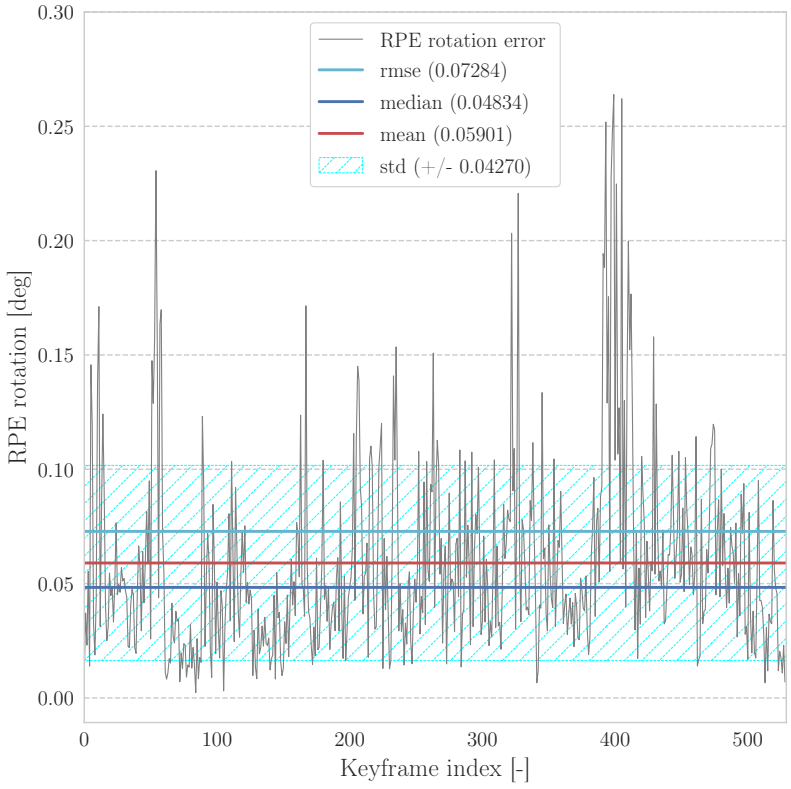


(a) RPE rotation for S+P

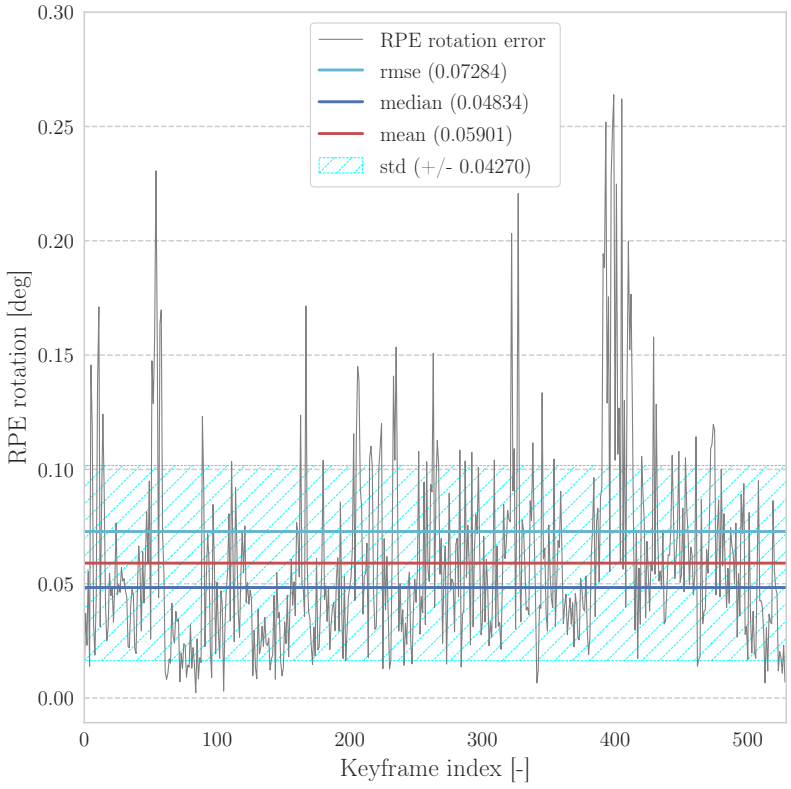


(b) RPE rotation for S+P+R

Figure D.8: Dataset `MH_04_difficult`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

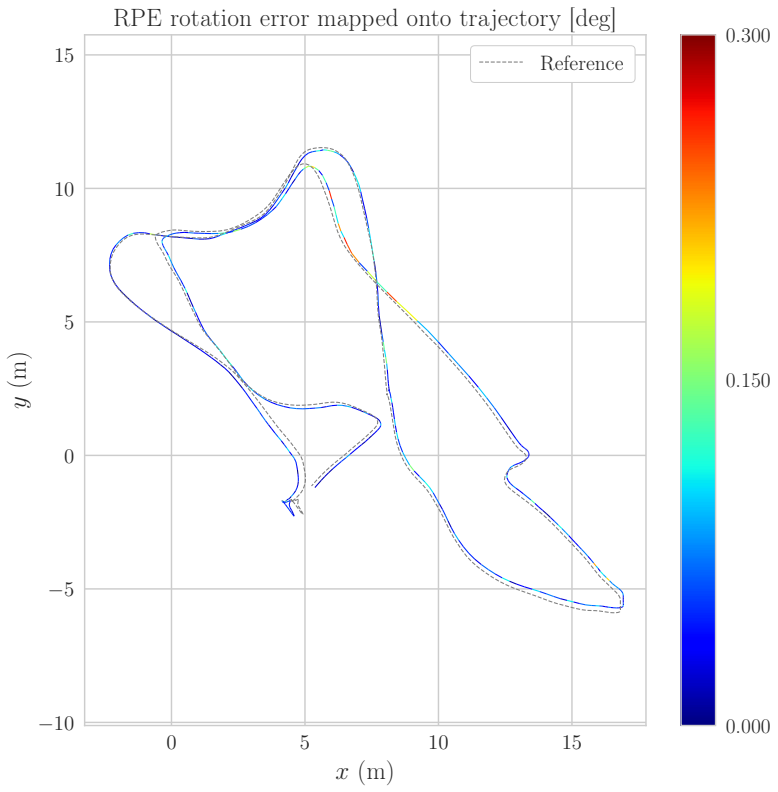


(a) S+P

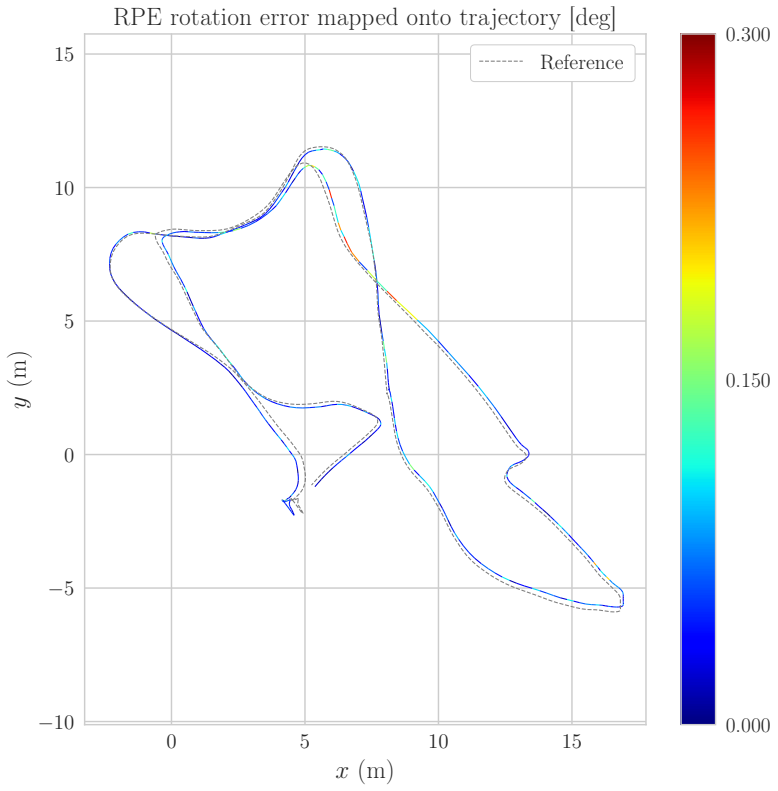


(b) S+P+R

Figure D.9: Dataset `MH_05_difficult`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

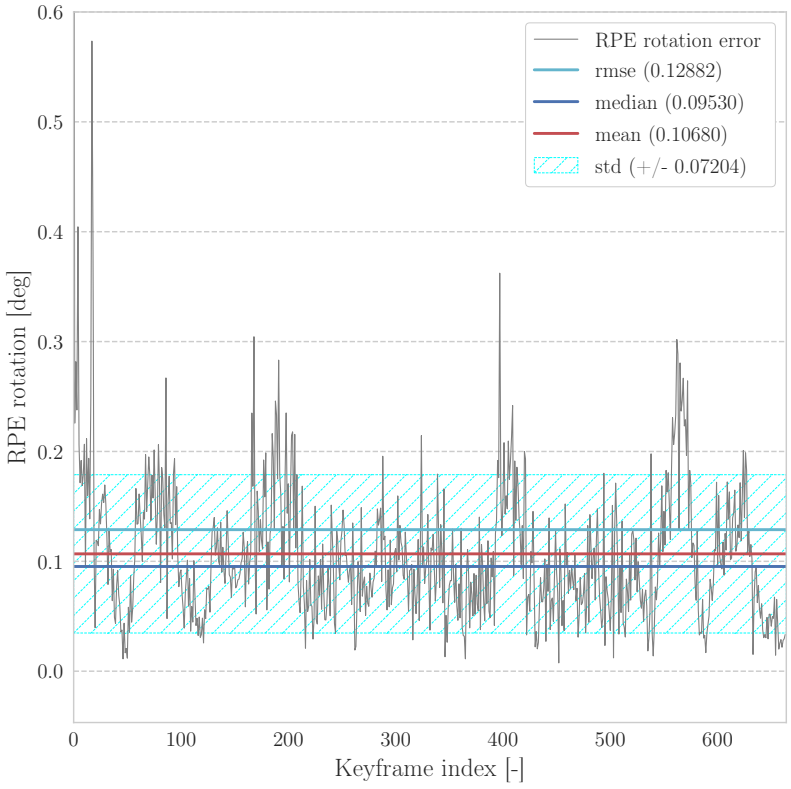


(a) RPE rotation for S+P

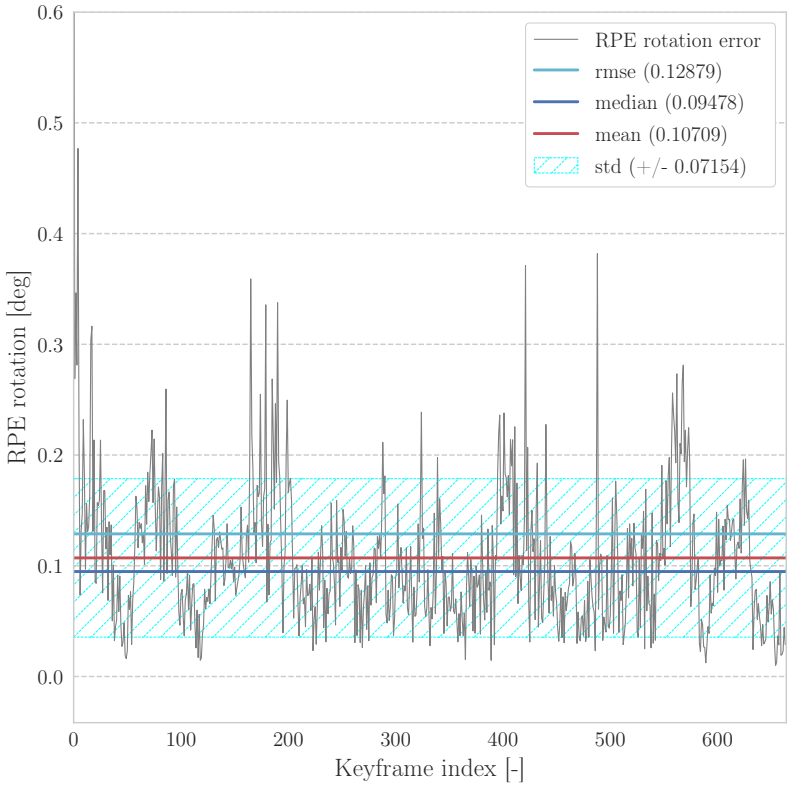


(b) RPE rotation for S+P+R

Figure D.10: Dataset `MH_05_difficult`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

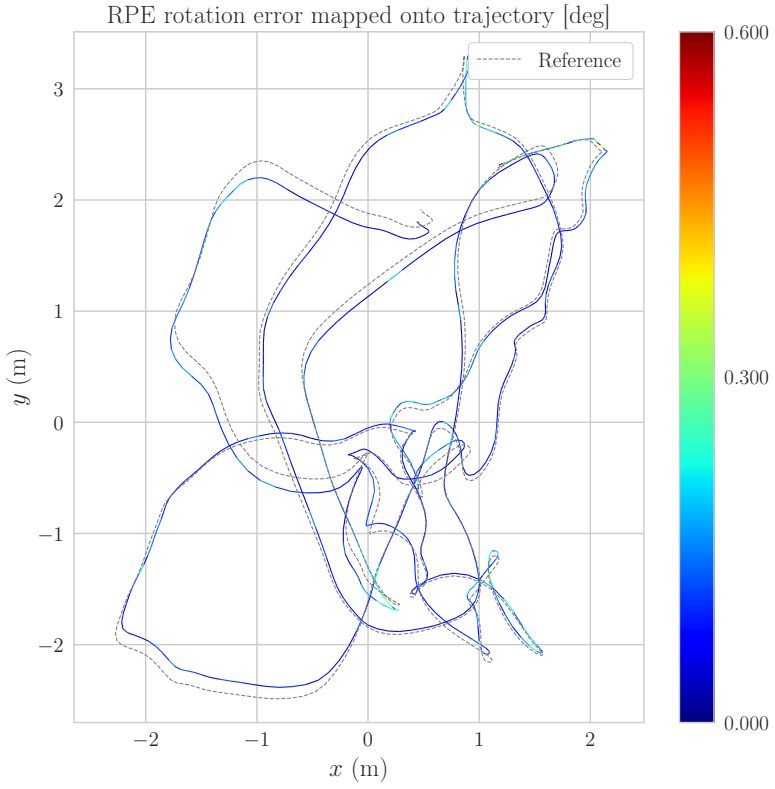


(a) S+P

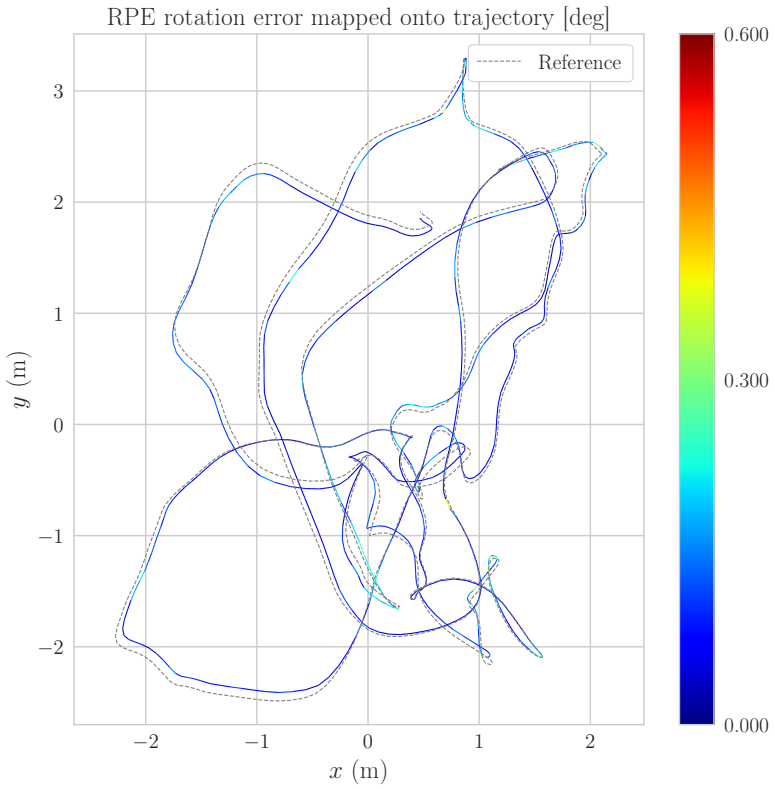


(b) S+P+R

Figure D.11: Dataset `V1_01_easy`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

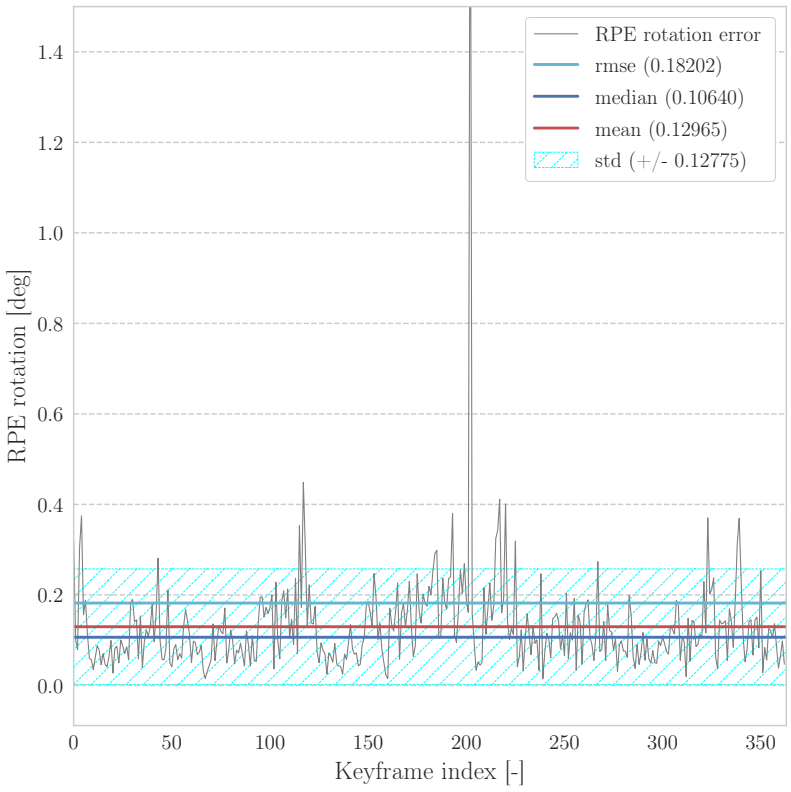


(a) RPE rotation for S+P

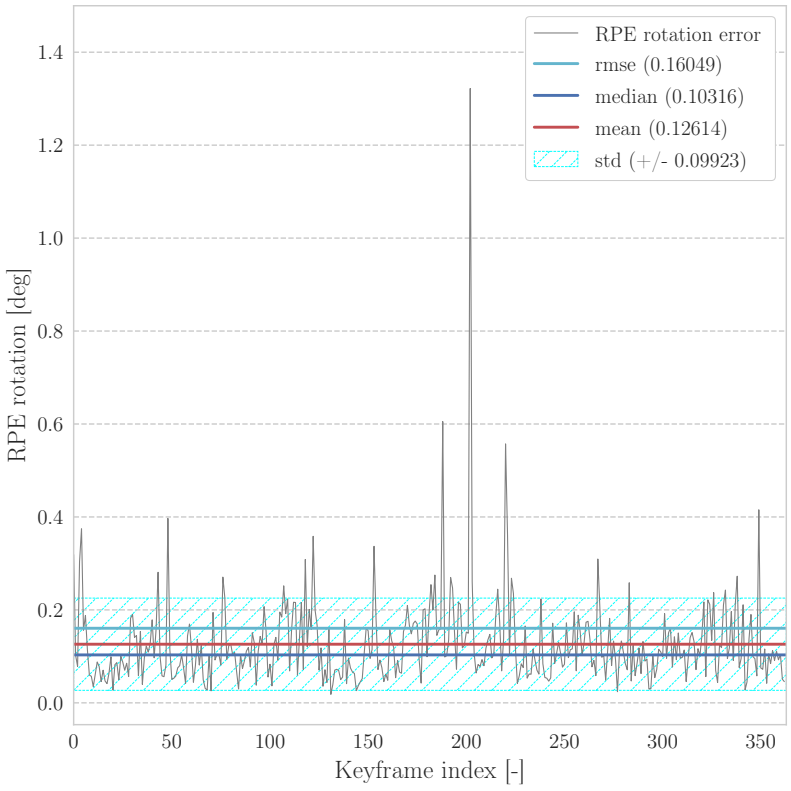


(b) RPE rotation for S+P+R

Figure D.12: Dataset `V1_01_easy`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

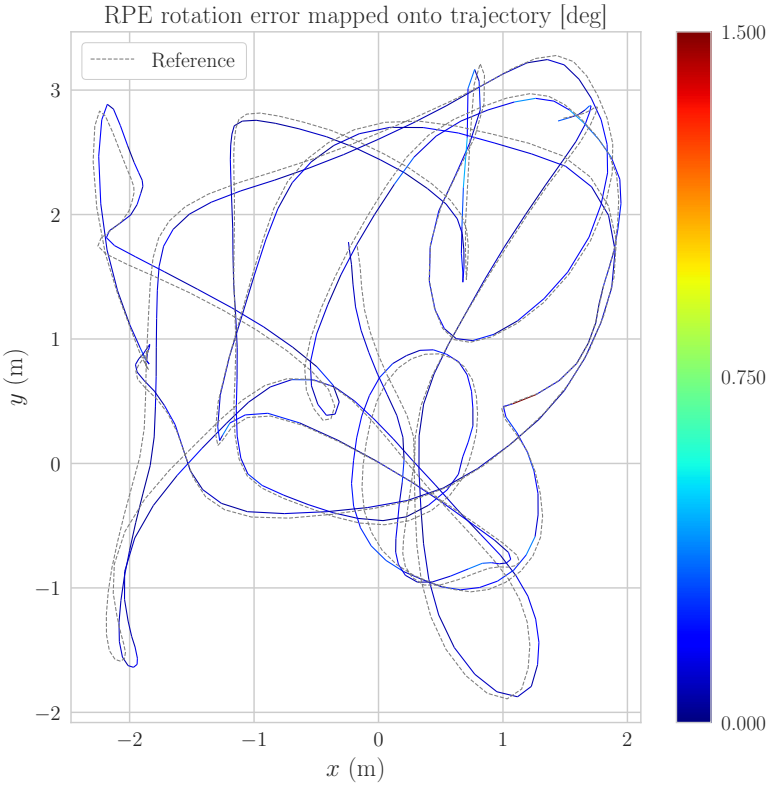


(a) S+P

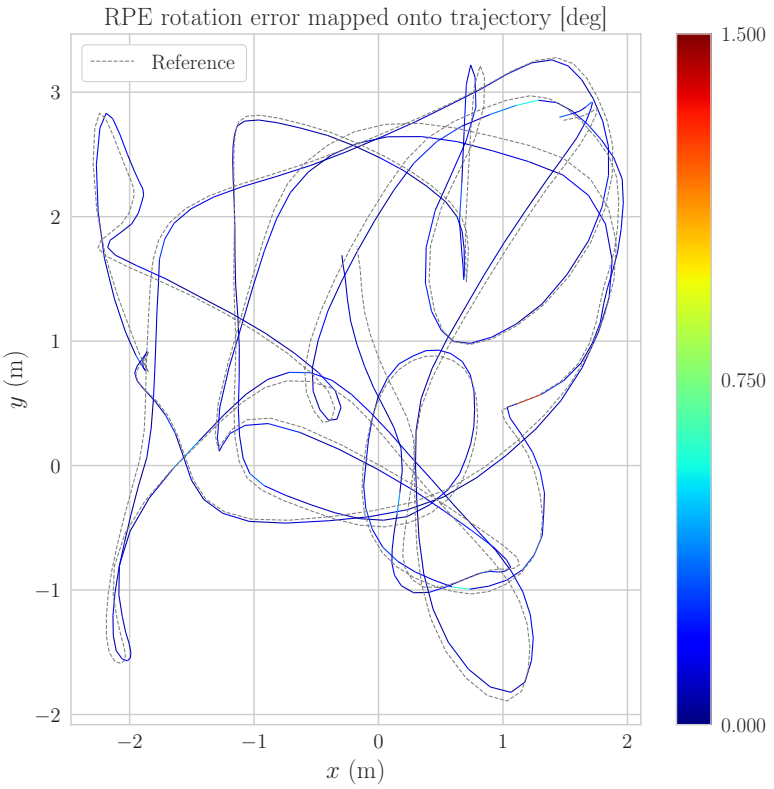


(b) S+P+R

Figure D.13: Dataset `V1_02_medium`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

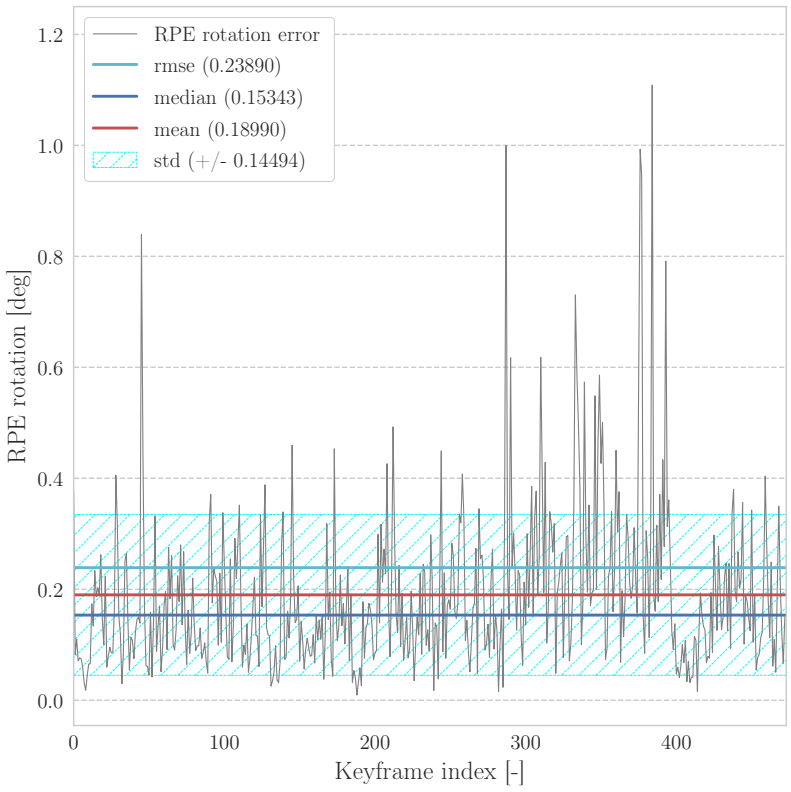


(a) RPE rotation for S+P

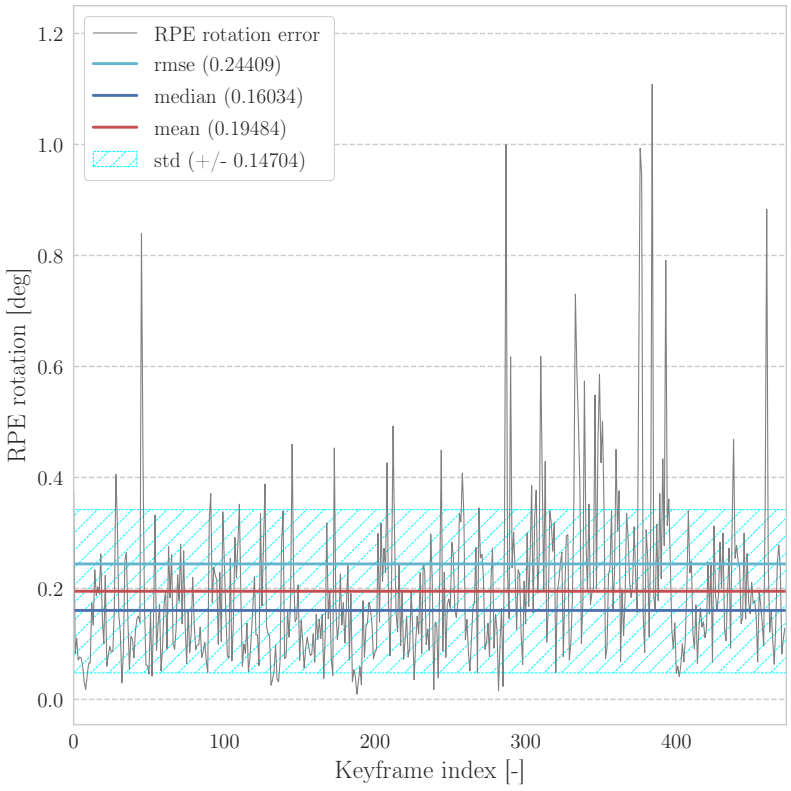


(b) RPE rotation for S+P+R

Figure D.14: Dataset `V1_02_medium`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

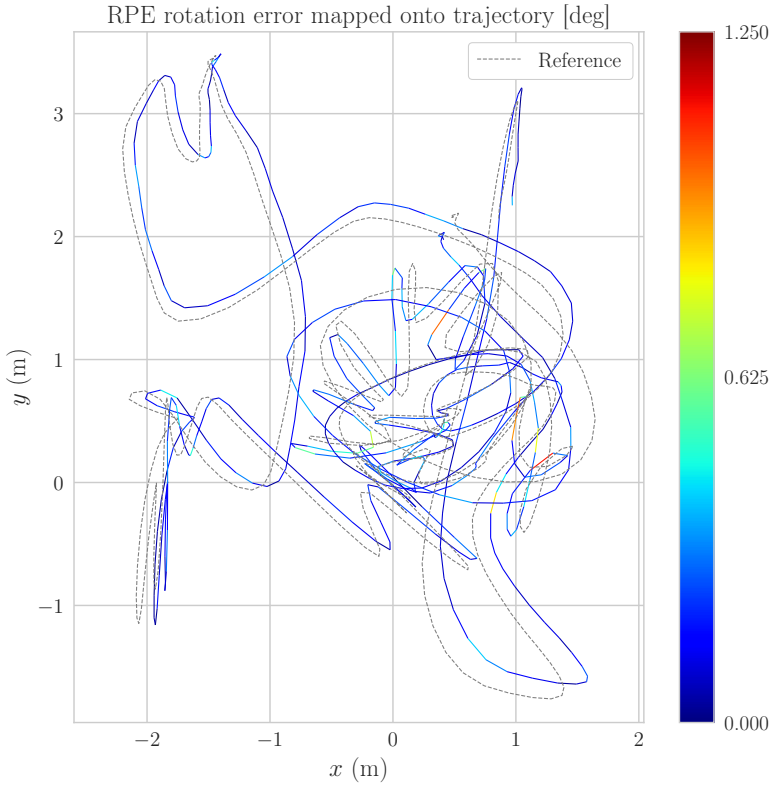


(a) S+P

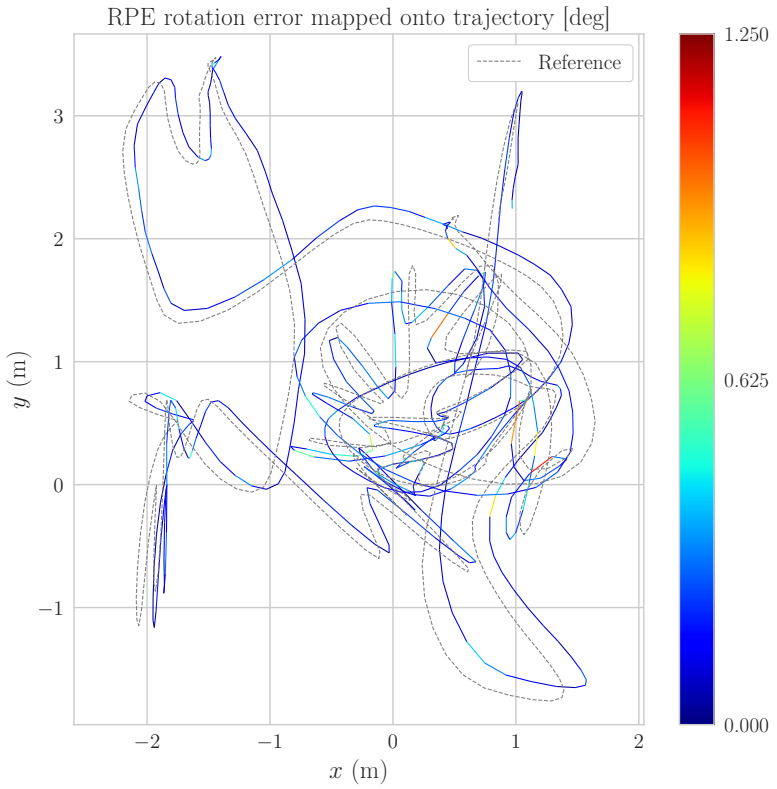


(b) S+P+R

Figure D.15: Dataset `V1_03_difficult`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

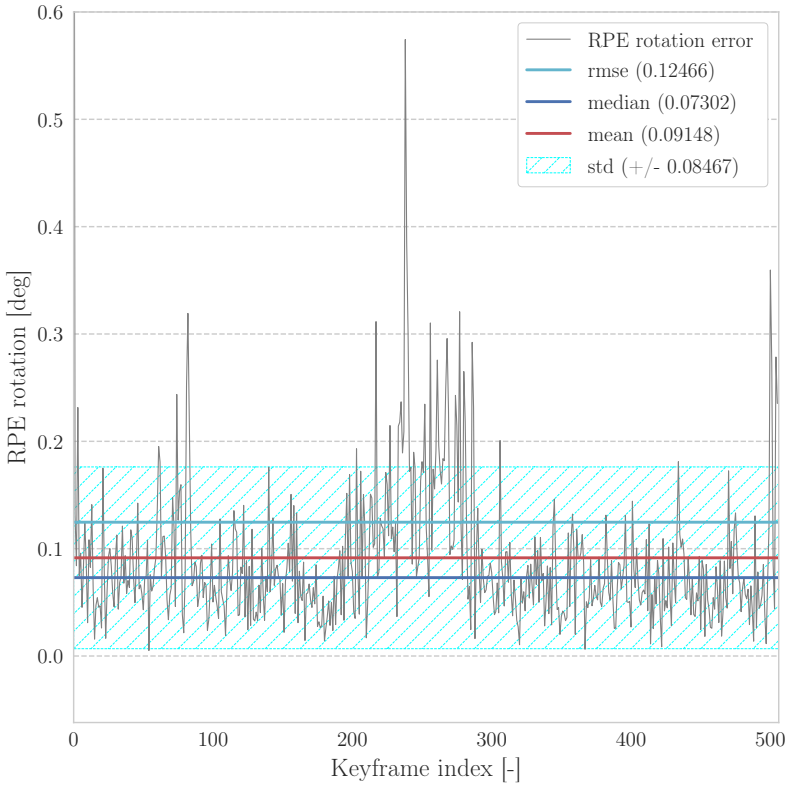


(a) RPE rotation for S+P

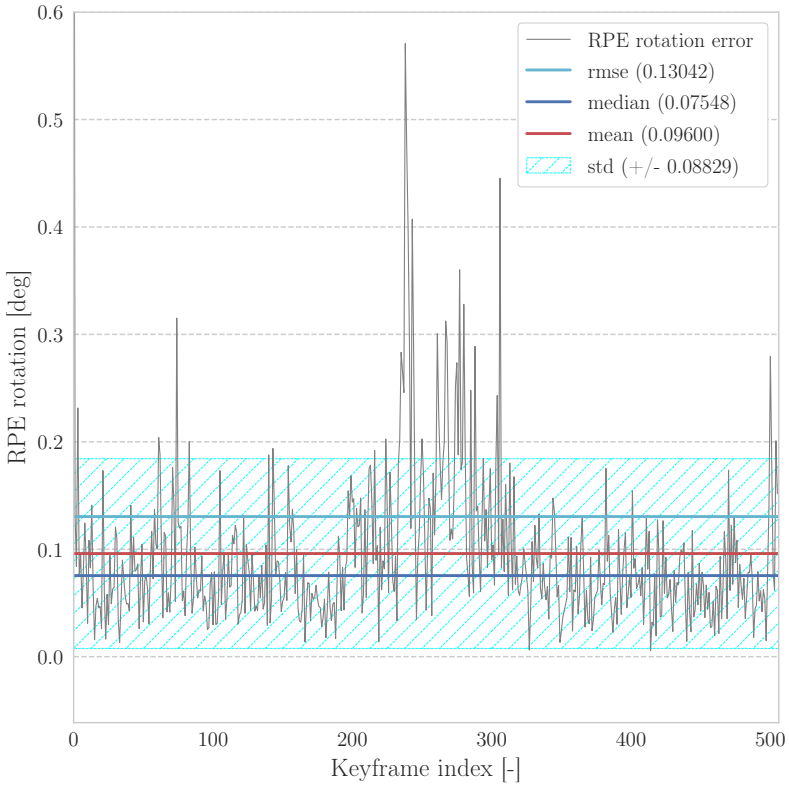


(b) RPE rotation for S+P+R

Figure D.16: Dataset `V1_03_difficult`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

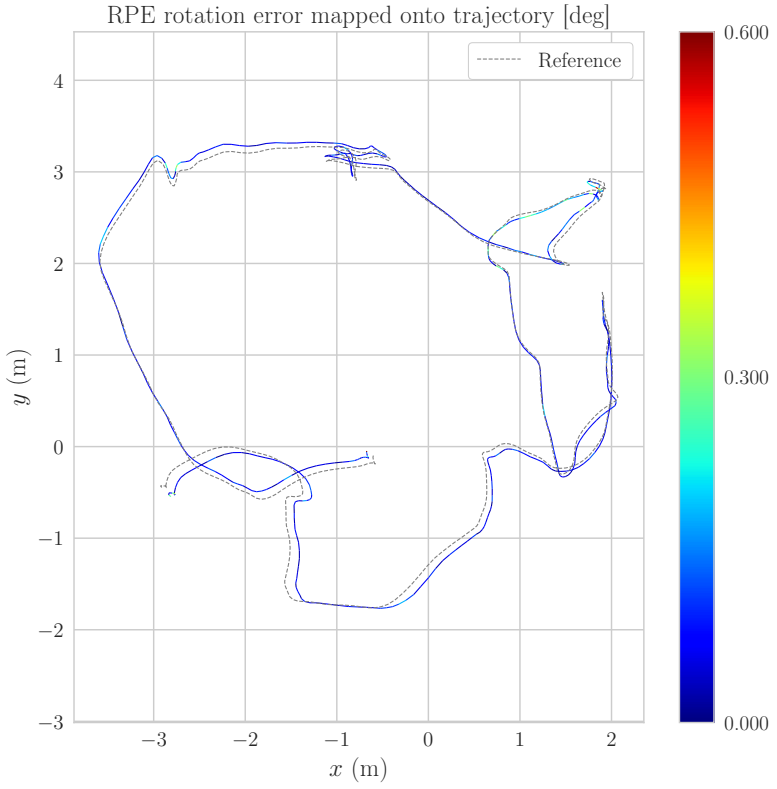


(a) S+P

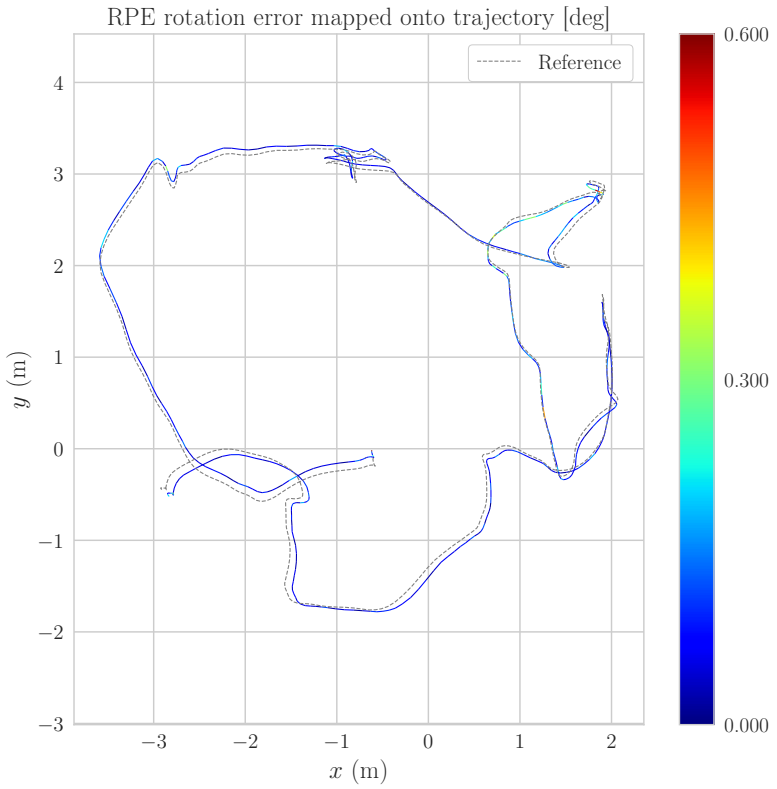


(b) S+P+R

Figure D.17: Dataset `V2_01_easy`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

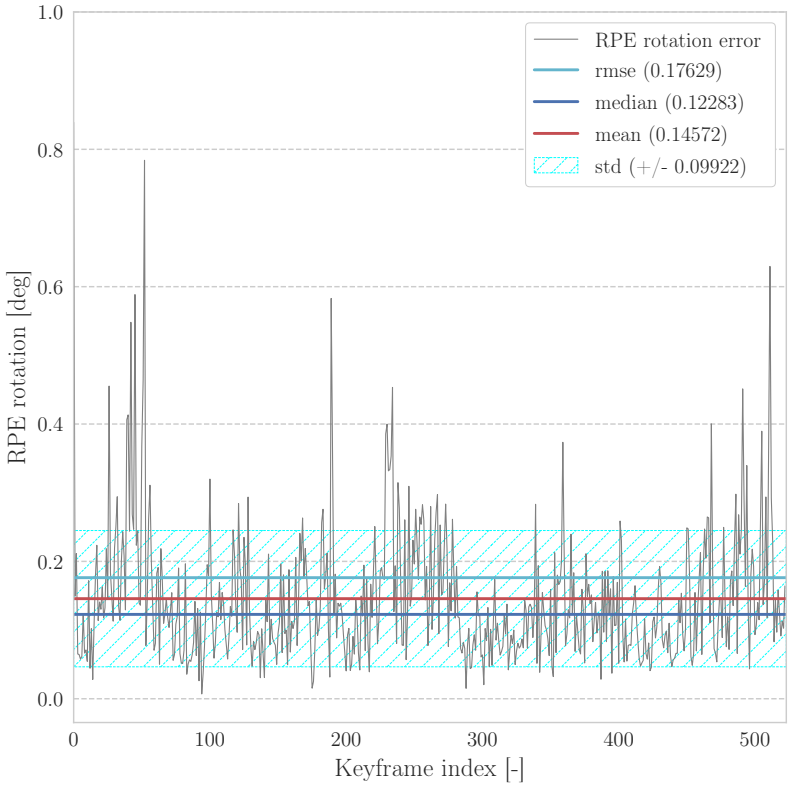


(a) RPE rotation for S+P

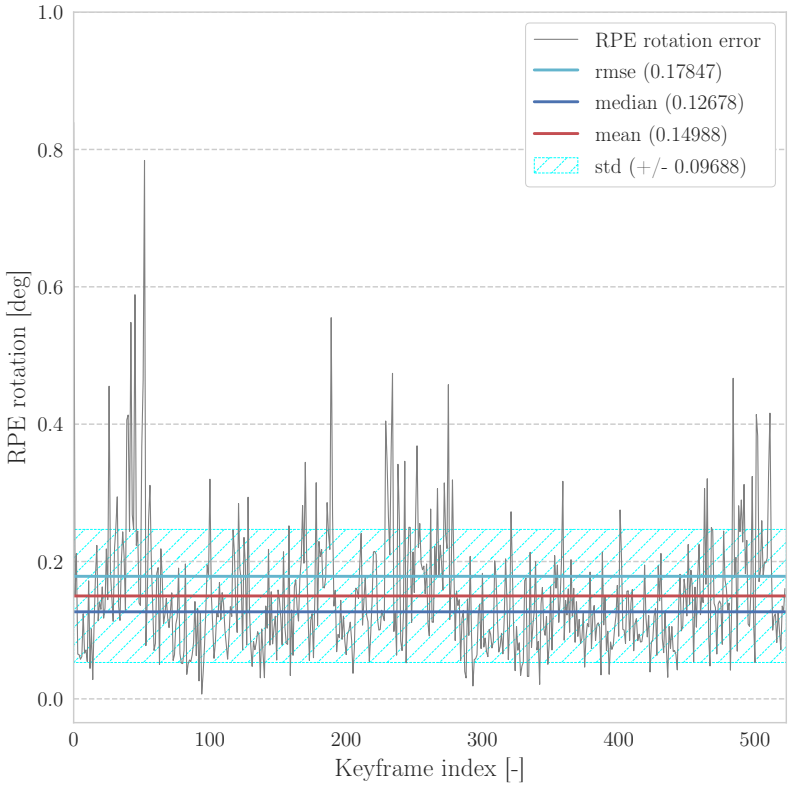


(b) RPE rotation for S+P+R

Figure D.18: Dataset `V2_01_easy`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

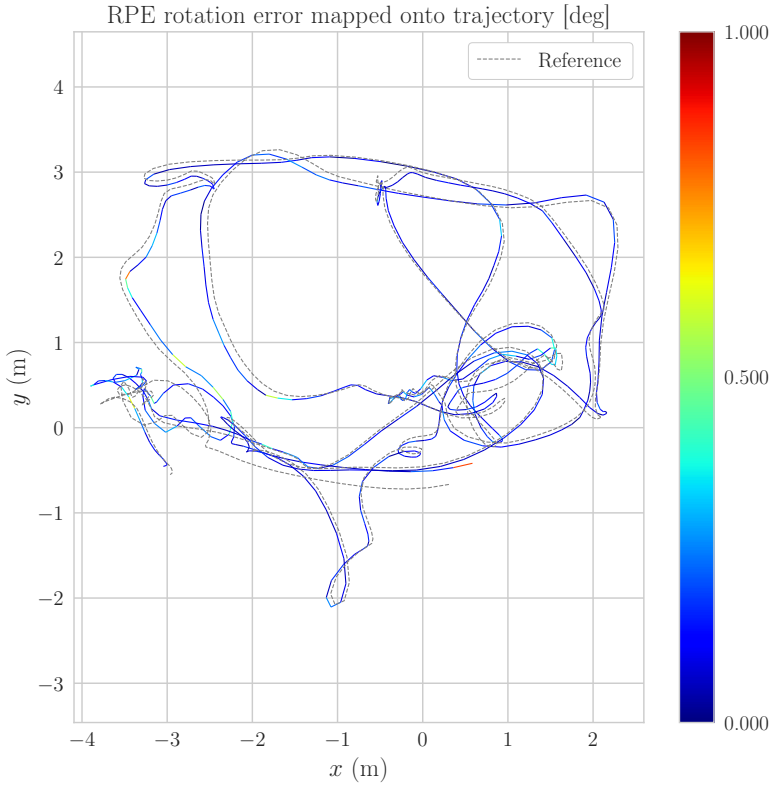


(a) S+P

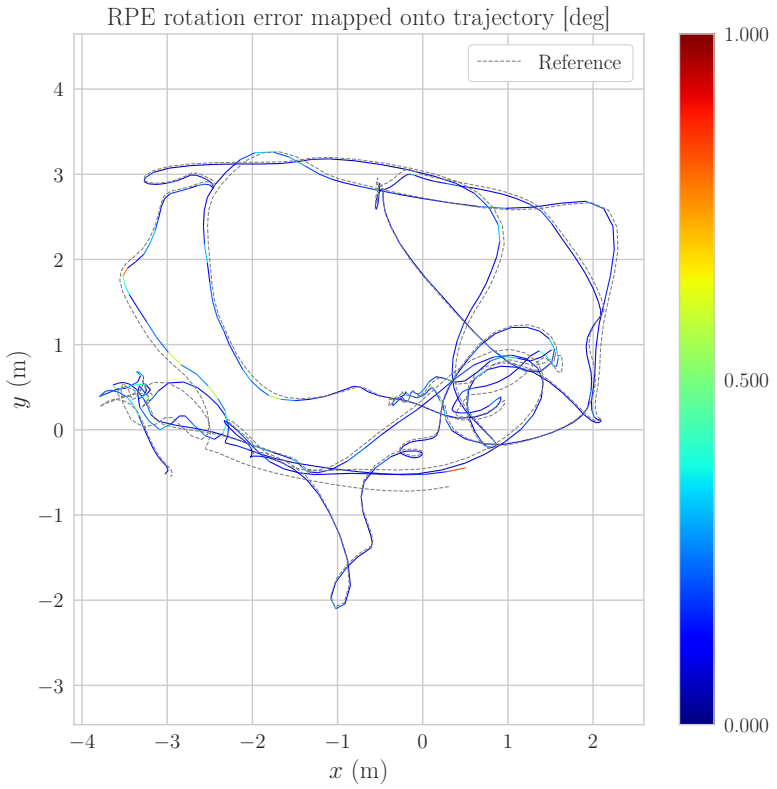


(b) S+P+R

Figure D.19: Dataset `V2_02_medium`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

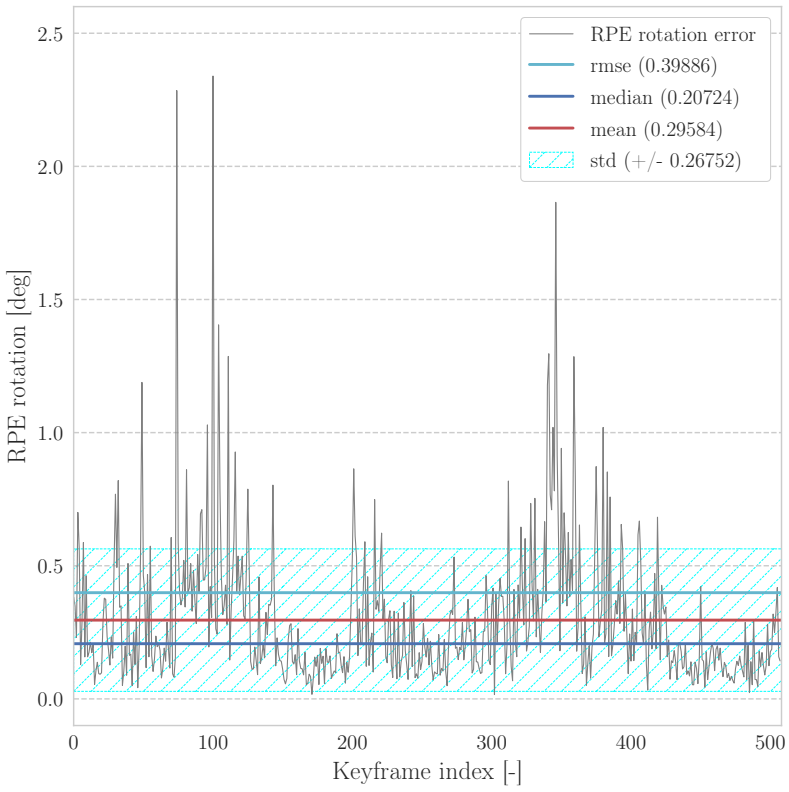


(a) RPE rotation for S+P

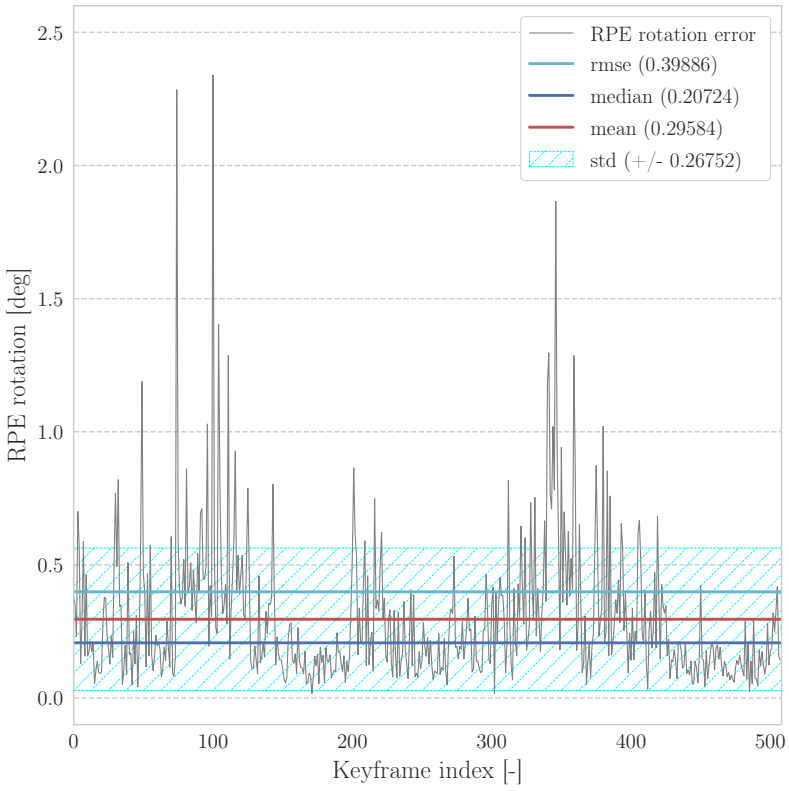


(b) RPE rotation for S+P+R

Figure D.20: Dataset `V2_02_medium`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

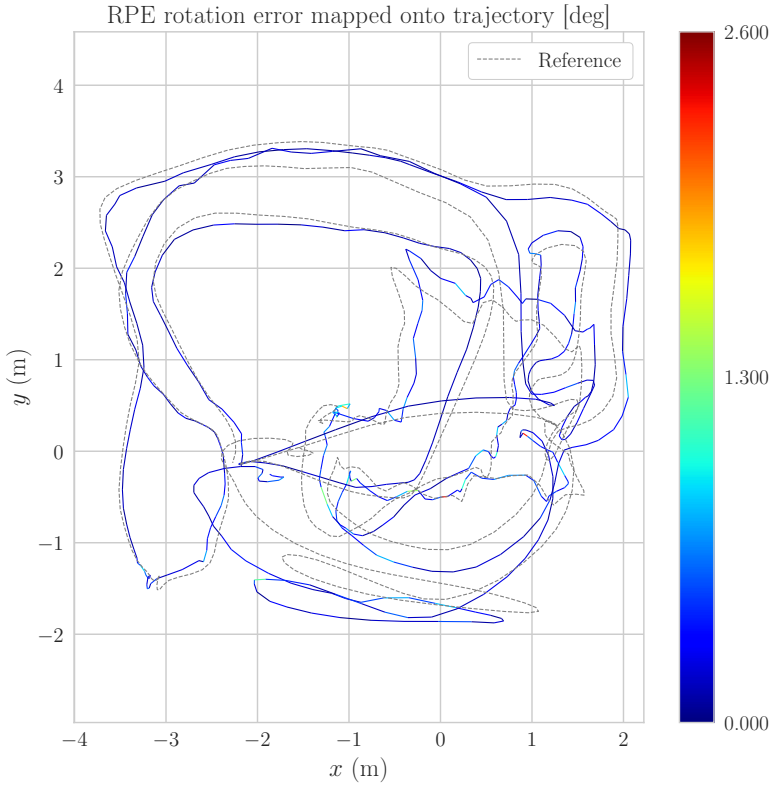


(a) S+P

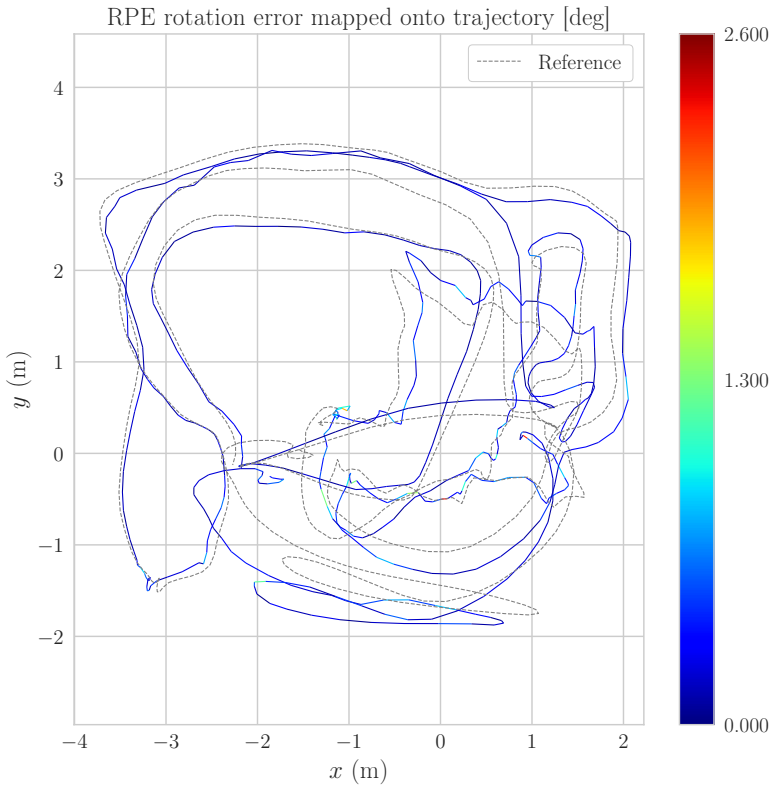


(b) S+P+R

Figure D.21: Dataset `V2_03_difficult`: RPE rotation error of VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.



(a) RPE rotation for S+P



(b) RPE rotation for S+P+R

Figure D.22: Dataset `V2_03_difficult`: RPE rotation error plotted on trajectory estimates for VIO using Strutureless (S) and Projection (P) factors, against S + P + Regularity (R) factors.

Bibliography

[1] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," *Memory*, vol. 10, p. 20, 2018.

[2] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, 2017.

[3] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial SLAM using nonlinear optimization," *Int. J. Robot. Research*, 2015.

[4] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial SLAM using nonlinear optimization," in *Robotics: Science and Systems (RSS)*, 2013.

[5] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[6] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 1498–1505.

[7] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 2320–2327.

[8] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. Davison, and A. Fitzgibbon, "KinectFusion: Real-time dynamic 3D surface reconstruction and interaction," in *SIGGRAPH*, Aug. 2011, p. 23.

[9] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *arXiv preprint arXiv:1708.03852*, 2017.

[10] W. N. Greene and N. Roy, "Flame: Fast lightweight mesh estimation using variational smoothing on delaunay graphs," in *Int. Conf. Comput. Vis. (ICCV)*, 2017.

[11] L. Teixeira and M. Chli, "Real-Time Mesh-based Scene Estimation for Aerial Inspection," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems(IROS)*, 2016.

[12] R. Smith, M. Self, and P. Cheeseman, "Autonomous robot vehicles," I. J. Cox and G. T. Wilfong, Eds. Berlin, Heidelberg: Springer-Verlag, 1990, ch. Estimating Uncertain Spatial Relationships in Robotics, pp. 167–193. [Online]. Available: http://dl.acm.org/citation.cfm?id=93002.93291

[13] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[14] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017. [Online]. Available: http://dx.doi.org/10.1561/2300000043

[15] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, Apr. 2007, pp. 3565–3572.

[16] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 4238–4243.

[17] K. Tsotsos, A. Chiuso, and S. Soatto, "Robust inference for visual-inertial sensor fusion," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5203–5210.

[18] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. IEEE, 2015.

[19] J. Aulinas, Y. R. Petillot, J. Salvi, and X. Lladó, "The slam problem: a survey." *CCIA*, vol. 184, no. 1, pp. 363–371, 2008.

[20] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[21] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2013, vol. 3.

[22] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[23] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003, second Edition.

[24] H. Strasdat, J. Montiel, and A. Davison, "Visual SLAM: Why filter?" *Image Vis. Comput.*, 2012.

[25] F. Dellaert, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," Georgia Institute of Technology, Tech. Rep. GIT-GVU-05-11, 2005.

[26] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Research*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006.

[27] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[28] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The Bayes tree: Enabling incremental reordering and fluid relinearization for online mapping," Computer Science and Artificial Intelligence Laboratory, MIT, Tech. Rep. MIT-CSAIL-TR-2010-021, Jan. 2010.

[29] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Research*, vol. 31, pp. 217–236, Feb. 2012.

[30] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 15–22.

[31] S.-H. Jung and C. Taylor, "Camera trajectory estimation using inertial sensor measurements and structure fom motion results," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2001.

[32] D. Sterlow and S. Singh, "Motion estimation from image and inertial measurements," *Int. J. Robot. Research*, 2004.

[33] M. Bryson, M. Johnson-Roberson, and S. Sukkarieh, "Airborne smoothing and mapping using vision and inertial sensors," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2009, pp. 3143–3148.

[34] V. Indelman, S. Wiliams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *J. Robot. and Auton. Syst.*, vol. 61, no. 8, pp. 721–738, Aug. 2013.

[35] S. Weiss, M. Achtelik, L. Kneip, D. Scaramuzza, and R. Siegwart, "Intuitive 3D maps for MAV terrain exploration and obstacle avoidance," *J. Intell. Robot. Syst.*, vol. 61, pp. 473–493, 2011.

[36] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE, 2016, pp. 1462–1468.

[37] L. Ma, M. Ghafarianzadeh, D. Coleman, N. Correll, and G. Sibley, "Simultaneous localization, mapping, and manipulation for unsupervised object discovery," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on.* IEEE, 2015, pp. 1344–1351.

[38] X. Huang, I. Walker, and S. Birchfield, "Occlusion-aware reconstruction and manipulation of 3D articulated objects," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012.

[39] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[40] Q. Pan, G. Reitmayr, and T. Drummond, "Proforma: Probabilistic feature-based on-line rapid model acquisition." in *BMVC*, vol. 2. Citeseer, 2009, p. 6.

[41] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell *et al.*, "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

[42] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nistér, and M. Pollefeys, "Real-time visibility-based fusion of depth maps," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on.* IEEE, 2007, pp. 1–8.

[43] R. Pajarola, "Overview of quadtree-based terrain triangulation and visualization," 2002.

[44] G. Klein and T. Drummond, "A single-frame visual gyroscope," in *British Machine Vis. Conf. (BMVC)*, 2005.

[45] J. Engel, J. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Eur. Conf. Comput. Vis. (ECCV)*, 2014.

[46] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," Feb. 2015, `arXiv:1502.00956`.

[47] R. Mur-Artal and J. D. Tardos, "Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM," in *Robotics: Science and Systems (RSS)*, 2015.

[48] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1449–1456.

[49] N. Krombach, D. Droeschel, and S. Behnke, "Combining feature-based and direct methods for semi-dense real-time stereo visual odometry," in *International Conference on Intelligent Autonomous Systems.* Springer, 2016, pp. 855–868.

[50] A. Concha and J. Civera, "Dense piecewise planar tracking and mapping from a monocular sequence," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[51] S. Pillai, S. Ramalingam, and J. Leonard, "High-performance and tunable stereo reconstruction," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016.

[52] P. Pinies, L. M. Paz, and P. Newman, "Dense mono reconstruction: Living with the pain of the plain plane," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5226–5231.

[53] D. I. Lovi *et al.*, "Incremental free-space carving for real-time 3d reconstruction," Ph.D. dissertation, University of Alberta, 2011.

[54] M. Lhuillier, "2-manifold tests for 3d delaunay triangulation-based surface reconstruction," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 98–105, 2015.

[55] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 351–358.

[56] A. Romanoni, A. Delaunoy, M. Pollefeys, and M. Matteucci, "Automatic 3d reconstruction of manifold meshes via delaunay triangulation and mesh sweeping," *arXiv preprint arXiv:1604.06258*, 2016.

[57] V. Litvinov and M. Lhuillier, "Incremental solid modeling from sparse and omnidirectional structure-from-motion data," in *British Machine Vision Conference*, 2013.

[58] A. Romanoni and M. Matteucci, "Incremental reconstruction of urban environments by edge-points delaunay triangulation," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 4473–4479.

[59] E. Piazza, A. Romanoni, and M. Matteucci, "Real-time cpu-based large-scale 3d mesh reconstruction," *arXiv preprint arXiv:1801.05230*, 2018.

[60] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 67–76.

[61] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 303–312.

[62] C. Zach, T. Pock, and H. Bischof, "A globally optimal algorithm for robust tv-l 1 range image integration," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.

[63] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 303–312.

[64] T. Whelan, J. B. McDonald, M. Kaess, M. F. Fallon, H. Johannsson, and J. J. Leonard, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul. 2012.

[65] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3d mapping in real-time on a cpu." in *ICRA*, 2014, pp. 2021–2028.

[66] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[67] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 335–342.

[68] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, "Real-time camera tracking and 3d reconstruction using signed distance functions." in *Robotics: Science and Systems*, vol. 2, 2013.

[69] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2013.

[70] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017.

[71] G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Mav visual slam with plane constraint," in *2011 IEEE International Conference on Robotics and Automation*, May 2011.

[72] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "Structslam: Visual slam with building structure lines," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1364–1375, 2015.

[73] M. Henein, M. Abello, V. Ila, and R. Mahony, "Exploring the effect of meta-structural information on the global consistency of slam," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 1616–1623.

[74] M. Hsiao, E. Westmat, and M. Kaess, "Dense planar-inertial slam with structural constraints," 2018.

[75] S. Yang and S. Scherer, "Direct monocular odometry using points and lines," *arXiv preprint arXiv:1703.06380*, 2017.

[76] R. Gomez-Ojeda, J. Briales, and J. Gonzalez-Jimenez, "Pl-svo: Semi-direct monocular visual odometry by combining points and line segments," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4211–4216.

[77] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane slam for hand-held 3d sensors." in *ICRA*, 2013, pp. 5182–5189.

[78] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and T. Garaas, "Tracking an rgb-d camera using points and planes," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 51–58.

[79] G. Zhang, D. H. Kang, and I. H. Suh, "Loop closure through vanishing points in a line-based monocular slam," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4565–4570.

[80] F. Camposeco and M. Pollefeys, "Using vanishing points to improve visual-inertial odometry," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5219–5225.

[81] L. Nicholson, M. Milford, and N. Sünderhauf, "Quadricslam: Constrained dual quadrics from object detections as landmarks in semantic slam," *arXiv preprint arXiv:1804.04011*, 2018.

[82] M. Hosseinzadeh, Y. Latif, T. Pham, N. Suenderhauf, and I. Reid, "Towards semantic slam: Points, planes and objects," *arXiv preprint arXiv:1804.09111*, 2018.

[83] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. Montiel, "Towards semantic slam using a monocular camera," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on.* IEEE, 2011, pp. 1277–1284.

[84] Y. Lu and D. Song, "Visual navigation using heterogeneous landmarks and unsupervised geometric constraints," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 736–749, 2015.

[85] J. Weingarten and R. Siegwart, "3d slam using planar segments," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.* IEEE, 2006, pp. 3062–3067.

[86] A. P. Gee, D. Chekhlov, W. W. Mayol-Cuevas, and A. Calway, "Discovering planes and collapsing the state space in visual slam." in *BMVC*, 2007, pp. 1–10.

[87] F. Servant, E. Marchand, P. Houlier, and I. Marchal, "Visual planes-based simultaneous localization and model refinement for augmented reality," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on.* IEEE, 2008, pp. 1–4.

[88] A. P. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas, "Discovering higher level structure in visual slam," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 980–990, 2008.

[89] J. Martínez-Carranza and A. Calway, "Unifying planar and point mapping in monocular slam." in *BMVC*, 2010, pp. 1–11.

[90] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "Cpa-slam: Consistent plane-model alignment for direct rgb-d slam," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE, 2016, pp. 1285–1291.

[91] M. Hsiao, E. Westman, G. Zhang, and M. Kaess, "Keyframe-based dense planar slam," in *IEEE International Conference on Robotics and Automation, ICRA, Singapore*, 2017.

[92] A. J. B. Trevor, J. G. Rogers, and H. I. Christensen, "Planar surface slam with 3d and 2d sensors," in *2012 IEEE International Conference on Robotics and Automation*, May 2012.

[93] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4605–4611.

[94] J. Neira and J. D. Tardos, "Data association in stochastic mapping using the joint compatibility test," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 890–897, Dec. 2001.

[95] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on.* IEEE, 2014, pp. 6218–6225.

[96] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time plane segmentation using rgb-d cameras," in *Robot Soccer World Cup.* Springer, 2011, pp. 306–317.

[97] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[98] N. Srinivasan, L. Carlone, and F. Dellaert, "Structural symmetries from motion for scene reconstruction and understanding." in *BMVC*, 2015, pp. 136–1.

[99] Y. Lu, J. Lee, S.-H. Yeh, H.-M. Cheng, B. Chen, and D. Song, "Sharing heterogeneous spatial knowledge: Map fusion between asynchronous monocular vision and lidar or other prior inputs," in *The International Symposium on Robotics Research (ISRR), Puerto Varas, Chile*, 2017.

[100] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2013.

[101] P. T. Furgale, "Extensions to the visual odometry pipeline for the exploration of planetary surfaces," Ph.D. dissertation, University of Toronto, 2011.

[102] L. Dietz, "Directed factor graph notation for generative models," *Max Planck Institute for Informatics, Tech. Rep*, 2010.

[103] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep. GT-RIM-CP&R-2012-002, Sep. 2012.

[104] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International journal of computer vision*, vol. 103, no. 3, pp. 267–305, 2013.

[105] P. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[106] Z. Zhang, "Parameter estimation techniques: a tutorial with application to conic fitting," *Image Vision Comput.*, vol. 15, pp. 59–76, 1997.

[107] J. Fox *et al.*, "Robust regression," *An R and S-Plus companion to applied regression*, p. 91, 2002.

[108] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE ACM Int. Sym. Mixed and Augmented Reality (ISMAR)*, Nara, Japan, Nov. 2007, pp. 225–234.

[109] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of The Fourth Alvey Vision Conference*, vol. 15, 1988, pp. 147–151.

[110] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," 2001.

[111] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 6, pp. 756–777, 2004.

[112] C. Zhang, Z. Li, Y. Cheng, R. Cai, H. Chao, and Y. Rui, "Meshstereo: A global stereo model with mesh alignment regularization for view interpolation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 2057–2065.

[113] T. R. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 943–949.

[114] S. Thrun, C. Martin, Y. Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard, "A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 433–443, 2004.

[115] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Research*, vol. 35, pp. 1157–1163, 2015.

[116] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr. 1987. [Online]. Available: http://josaa.osa.org/abstract.cfm?URI=josaa-4-4-629

[117] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 4, 1991.

[118] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, Oct 1997. [Online]. Available: https://doi.org/10.1023/A:1008854305733

[119] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014.

[120] A. Rosiñol Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[121] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, p. 387, Sep 2009. [Online]. Available: https://doi.org/10.1007/s10514-009-9155-6

[122] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2012.

[123] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2012.

[124] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[125] Cloudcompare.org, "Cloudcompare - open source project," https://www.cloudcompare.org.

[126] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[127] T. Whelan, L. Ma, E. Bondarev, J. McDonald *et al.*, "Incremental and batch planar simplification of dense point cloud maps," *Robotics and Autonomous Systems*, vol. 69, pp. 3–14, 2015.

[128] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 801–808.