Fusion of GNSS measurements into VSLAM Bundle Adjustment process for geo-referenced positioning

by

Shubham Singh

19109699

This report is submitted as part requirement for the MRes Degree in ’Virtual Reality’, at University College London. It is substantially the result of my own work except where explicitly indicated in the text.

Acknowledgement

Foremost, I would like to acknowledge the support of project supervisors - Prof. Simon Julier and Prof. Anthony Steed. Without their guidance and teachings, this project wouldn't have been possible. Next, I would also like to thank Prof. Niloy Mitra for initial discussion and guidance related to this project.

The key contribution of this project is the fusion of GNSS measurement in visual simultaneous localization and mapping process, which is largely my own work. However, I would like to acknowledge the following open source projects, which I have used in this project:

- **OpenVSLAM** by xdspacelab (https://github.com/xdspacelab/openvslam), used for visual SLAM process.
- **GNSS compare** (https://github.com/TheGalfins/GNSS_Compare), used to process raw ephemeris data of Satellites.

Abstract

Over the last two decades, we have seen an increasing applications for simultaneous localization and mapping (SLAM) systems, predominantly in the field of autonomous driving, piloting unmanned aerial vehicle, geographical mapping and lately in augmented reality systems. The accurate localization and spatial understanding of the physical environment is a common factor in all of these applications. To support these applications visual SLAM system relies upon the fusion of multiple sensor inputs, like - camera, Light Detection and Ranging (LiDAR), inertial measurement unit (IMU), Global Navigation Satellite System (GNSS), etc.

In this project, we investigate the fusion of Global Navigation Satellite System (GNSS) measurements into the visual SLAM (VSLAM) system, to achieve accurate localization of the camera on a geographical scale. The state-of-the-art monocular VSLAM system can be used to very accurately determine the rigid-body motion of the camera. However, such systems suffer from scale ambiguity and drifts among several other issues. By combining the measurement from multiple sensors, we can overcome these shortcomings to achieve large scale and robust localization of the system. This project proposes a tightly coupled GNSS-VSLAM system that fuses the GNSS measurement in the graph-based global bundle adjustment to remove the scale ambiguity of the monocular SLAM and correct the noisy GNSS measurements. The geo-referenced positioning of the camera and the estimation of attitude in this manner eliminates the need for global sensors like - magnetometer and gyroscope, which are very noisy and highly susceptible to environmental disturbances.

The proposed GNSS-VSLAM system results in smooth camera trajectory compared to noisy standard GPS measurement and better map scale compared to monocular VSLAM system. However, it lacks the robustness and tracking stability required to support real-time applications, such as - outdoor Augmented Reality, navigation, etc. It is mainly due to the high variance and low-frequency updates of the RAW GNSS data. The system can be further improved by fusing with high-frequency inertial measurements or adopting a different re-localization based approaches.

CONTENTS

1. INTRODUCTION

Augmented Reality (AR) has come a long way, from early AR display system developed in 1968 by Ivan Sutherland [1] which takes up the entire room space, to the current state of the art mobile AR systems (ARCore and ARKit) which fits conveniently into the users pocket. The latest mobile AR systems include features such as - real-time depth maps, body tracking, physically based rendering (occlusion, shadows and lighting), scene understanding, etc. These systems are actively being adopted for mainstream applications, like - medical training, retail shopping, interactive data visualization, entertainment applications etc.

The use of augmented reality for indoor application is highly explored, however lesser work has been done for the outdoor applications. It is mainly because the mapping of the static indoor environment is relatively easy and deterministic compared to an outdoor location. The mapping of large-scale physical environment and maintaining a consistent map for user localization for a longer period is a highly challenging problem. However, with the development of computer vision algorithms and optimized sensor fusion strategies along with the growing computational power, it has become an achievable task. The large-scale outdoor tracking for AR will open up new possibilities for users. For example, in tourism, an augmented reality application can be used to create a virtual guided tour of any physical place of attraction. It can also be used for smart navigation systems or linked to a digital twin of the city (smart-city projects) for interactive experiences.

The motivation for this research is derived from my project work on mixed reality-based platform for collaboration in geo-referenced environment. The collaboration platform supports user on both the augmented reality (AR) and virtual reality (VR) systems by using a GPS-based (Global Position System) based coordinate system defined above the local euclidean coordinate system. The GPS world coordinate system allows to localize the outdoor AR user on a real-world geographical coordinate system, which can be used for a range of real-world applications. On the VR platform, the user can virtually experience the digital twin of the real world with an additional layer of interactive contents representing the shared space (including user avatars, 3D models, videos, etc.). The application supports real-time collaboration between users across the platform by connecting the two different spaces, which allows the user to share their experience with remote users in a seamless way.

This collaboration platform provided a novel method of interaction between users using mixed reality technologies. It successfully demonstrates basic collaboration between users, including features such as - voice conferencing, using Avatar to represent users, action synchronization, freehand drawing in virtual 3D space and virtual content placement at run-time. However, the state of the art AR systems are not ideal for outdoor experiences [2][3] and pose multiple issues like - mapping of a dynamic environment, inconsistent global brightness (due to environmental factors), the scale of the environment etc. These issues became the challenges for the platform and resulted in limited robustness and use cases.

This research project is motivated from these limitations and investigate the possibility of geo-referenced localization for the outdoor AR user on a global scale with a few centimetres level of accuracy. There have been several attempts to address the problem of large-scale outdoor tracking for AR, however without having much success. The previous solutions/literature can not be either generalized or requires complex and costly instruments for the process, as we will see in more details in the background chapter. In this project, we use a simple monocular camera for vision-based simultaneous localization and mapping (SLAM) and geographical navigation satellite system (GNSS) sensor for global positioning and tracking of the user in an outdoor environment. Usually, both of these two sensors are available on a consumer mobile phone, as such the proposed solution can be easily extended for the mobile platform as well.

The main objective of this project is to develop a generalized framework for the integration of GNSS measurements into the visual SLAM (VSLAM) system to achieve the geo-referenced positioning and attitude for the sensor system. There exist multiple solutions for the integration of GNSS measurements and inertial data into VSLAM systems using either filtering-based or graph-based approach. But to my knowledge, there's no research on using just the GNSS measurement in graph-based bundle adjustment process of visual SLAM. In this report, we will understand how to approach the integration of global sensor measurements into visual SLAM bundle adjustment and what are the challenges to achieve geo-referenced position and attitude of the user. Other contribution of this project includes - estimation of map scale from GNSS measurement to overcome the scale ambiguity issue of monocular VSLAM and approaches for estimating the transformation between different coordinate systems.

This project is the preliminary work, performed as the starting point for further research on the topic of geo-reference AR system. Along with this report, complete source code and software builds used in the project are uploaded on Github (Please refer to the appendix for code listings and user manual). In the next chapter, we will cover the general theory and previous works related to this research. Next, the analysis and design chapter introduces the GNSS-VSLAM systems' architecture and dependencies, along with the decisions made to limit the scope of this project. The implementation chapter provides the problem formulation and its solution used in the project. Finally, the testing and results of the system are documented in the following chapters, and the remaining technical challenges are discussed.

2. BACKGROUND

In this chapter, we will start with understanding visual SLAM and global positioning system, used for localizing the user on local and a global scale respectively. Next, we will cover the relevant research and previous works related to this project.

I. VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING (VSLAM)

The problem of localizing the robot (system of sensors) is a traditional topic in the field of robotics and is a well-studied problem in computer vision. There has been decades of work on the implementation of localization of robot and mapping of the physical environment using a single or a set of sensors. Generally, this problem is referred to as the simultaneous localization and mapping (SLAM) or structure from motion (SFM). Different kinds of visual and ranging sensors such as - monocular or stereo cameras, Light Detection and Ranging (LiDAR), Sound Navigation and Ranging (SONAR) etc. are used to perform the mapping of the environment. The sensors can detect interesting features in the image/frame (keypoints) or measure the time of flight to determine the three-dimensional position of a point in the physical world (map-point). The collection of such map-points will eventually grow into a big set of points in 3D space (these 3D points are also called the point cloud).

In Visual simultaneous localization and mapping (VSLAM) method, the robot simultaneously performs the task of mapping the environment and self-localization using vision-based sensors (i.e. cameras). Multiple visual sensors can be arranged in different configuration like - monocular, stereoscope, 360 degrees, etc. This problem of simultaneous mapping and localization is referred by different names in different disciplines, like - Visual Odometry, Photogrammetry, Multi-view reconstruction, Structure from motion etc. However, there is a slight difference between these methods. For example, in Photogrammetry the reconstruction of the 3D structure is achieved from an unordered set of images using frame-to-frame continuity. It is targeted for offline usage and processes images captured from an uncalibrated camera as well. Visual odometry is used in problems where the retrieval of robot position is more important than generating a detailed 3D map of the environment. There is no distinct separation between the use of these terminologies, but the common factor is that they all perform the task of simultaneous localization and mapping (SLAM).

To better understand the process, let's take a close look into the process of VSLAM. We will understand how the 3D map point positions and camera poses are estimated from a sequence of images, and what are the requirement and challenges at every step of the process.

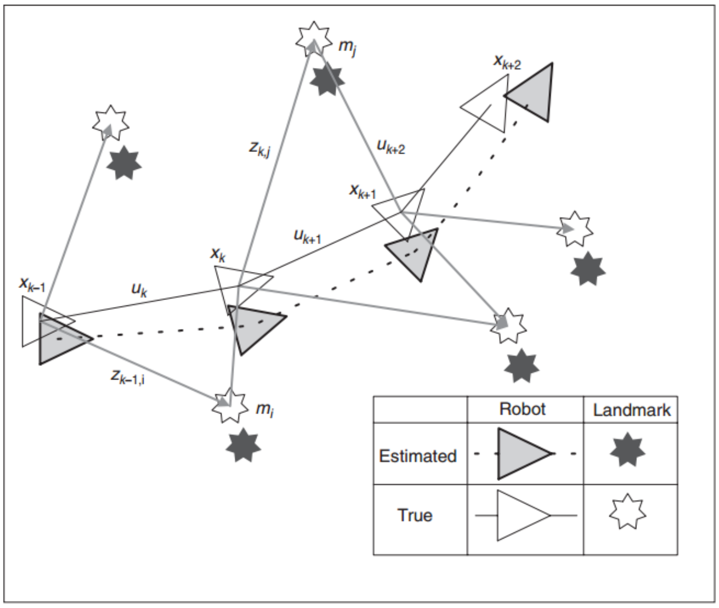


Figure 2.1: The simultaneous estimate of both robot (i.e. camera) and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations. Credits: [4]

Consider a rigid-body setup where a calibrated camera (i.e. robot) is moving through an environment. The starting point of the camera is fixed, as the point of reference (i.e. origin) for the SLAM world. The virtual coordinate system for the camera is defined as per the left-hand rule, where x and y axes lie in the camera's image plane, and the z-axis is orthogonal to this plane, pointing in the outwards direction. Figure 2.1, shows the moving camera about the static points capturing N keyframes. The keyframe is defined for a set of image frames capturing at least M point features taken from distinct views of the scene. Each keyframe has a corresponding reference frame, which is aligned with the camera frame at the time instant when the image was captured. At any time instant k, the following quantities are defined: u_k the state vector describing the location and orientation of the camera; X_{ki} the vector describing the location of the i^{th} landmark whose true location is assumed to be time-invariant; x_{ki} the 2D observation taken from camera image for the location of i^{th} landmark at time k.

The visual SLAM is an inverse graphics problem compared to computer graphics, where we deal with the task of rasterizing a given 3D models and camera configuration into a 2D image. Consider a 3D world point **X** expressed in homogeneous form

$$\mathbf{X} = [X \quad Y \quad Z \quad W]^T \tag{2.1}$$

and a 2D image point **x**,

$$\mathbf{x} = [x \quad y \quad d]^T \tag{2.2}$$

where d is for the depth of the pixel (x,y). In computer graphics, the projection of 3D point onto the 2D screen is given by

$$\begin{aligned} \lambda\boldsymbol{x} &= \boldsymbol{PX}, \\ \mathbf{P} &= \boldsymbol{M_{projection}} * \boldsymbol{M_{view}} * \boldsymbol{M_{model}} \end{aligned} \tag{2.3}$$

here λ is the inverse depth of the 3D point and **P** is a 3x4 projection camera matrix. It is important to note that, here we assume the image is free from any type of lens or geometric distortions. In the case of the inverse graphics, the projection matrix **P** is defined as follows

$$\begin{aligned} \boldsymbol{P} &= \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}], \\ \boldsymbol{K} &= \begin{bmatrix} \alpha f_x & s & c_x \\ 0 & \alpha f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.4}$$

here **K** is the matrix comprising of camera intrinsic parameters, containing - focal length f, camera optical center c, and skew factor s. **R** and **t** are the corresponding rotation and translation of camera referred to as the extrinsic parameters. The matrix **K** can be determined from the calibration process of the camera. The common way of calibration is by using a known pattern of image (like a checkerboard or an asymmetric circle pattern). For a stable VSLAM, it is very necessary to obtain the precise intrinsic camera parameters for matrix and distortion coefficients for camera lenses. The OpenCV library used in this project provides in-built methods for the calibration of the camera. However, to use a custom image resolution for VSLAM, we will need to determine new camera parameters and distortion coefficients. Thus I have created c++ executable to calibrate the camera with custom width and height, and save the parameters into a YAML file (Check appendix for usage).

The point to point correspondences between image frames is required to estimate the camera trajectory from a sequence of images. With the help of epipolar geometry, we can determine the extrinsic parameters of the camera (i.e. rotation and translation) between image pairs. The estimation of the point to point correspondences between image pair is a key step in the VSLAM process. However, in an average image resolution of 640x480, there are 0.3 M pixel points. Due to limited computation power, we can not estimate the correspondences for every pixel in the image and thus, we try to select a few key points and respective key descriptors in every frame. For general use, we can limit the size of keypoints to 2000 for any single frame, which is the value used in [5]. There are several feature-based methods for the detection and matching of keypoints, like - FAST (Features from Accelerated Segment Test), SIFT (Scale Invariant Feature Transform), SURF (Speeded-Up Robust Features); ORB: Oriented FAST and Rotated BRIEF (Binary Robust Independent Elementary Features), etc.

SIFT uses the maxima from a difference-of-Gaussian (DOG) pyramid as features, which makes it robust for varying scale conditions. To make SIFT rotation-invariant, the keypoint descriptor can be rotated to fit the orientation. In case of SURF, the DOG is replaced with a Hessian matrix-based blob detector. SURF computes for the sums of gradient components and the sums of their absolute values. SURF is computationally faster than SIFT but comes with a drawback of lesser accuracy in tracking feature. The advantage of using FAST corner detection is its computational efficiency which makes it very suitable for real-time applications.

The OpenVSLAM used in this project uses ORB-feature detection [6] for point correspondences and matching. ORB method uses binary features that are invariant to rotation and scale (up to a limit), resulting in a fast recognition of feature points with good invariance to camera viewpoint. Raul Mur-Artal et al. showed the superior performance of ORB for place recognition in [7].

The feature-based method extracts the corner points from the image using the high change in intensity along the x and y direction of the image. The image gradient (∇I) is defined in terms of its partial derivative in x and y direction and is represented as following

$$\nabla I = \begin{bmatrix} I_x & I_y \end{bmatrix}^T \tag{2.5}$$

The partial derivative I_x and I_y are calculated by the convolution of the source image I by the partial Gaussian kernel $G_{\sigma x}$ and $G_{\sigma y}$ respectively. Then we define matrix $\boldsymbol{M_I}$ as follows

$$\boldsymbol{M_I} = \nabla I \nabla I^T = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.6}$$

The eigenvalues of M_I, λ_1 and λ_2 can define the type of feature for the region. If both λ_1 and λ_2 are large positive value then there's a corner; if either one is large and the other is very small ($\approx$ 0) then the point lies on the edge, and if both the values are very small then point lies in a flat region of constant intensity. The reason for using the distinct feature is to enable the tracking of unique image region across images to solve the correspondence problem.

The advantage of using a feature-based tracking and matching is that it can handle large baseline cases during tacking and the relocalisation is possible in case of losing track, unlike the optical flow-based methods. Now the geometric constraints can be established for a pair of images taken from either different cameras (i.e. stereo) or single-camera (i.e. monocular). However, when the motion between the two images is not known, the epipolar geometry is undetermined. Without the known inter-camera distance, the scale of locally constructed map portions and the corresponding motion estimates is liable to global drift over time [8].

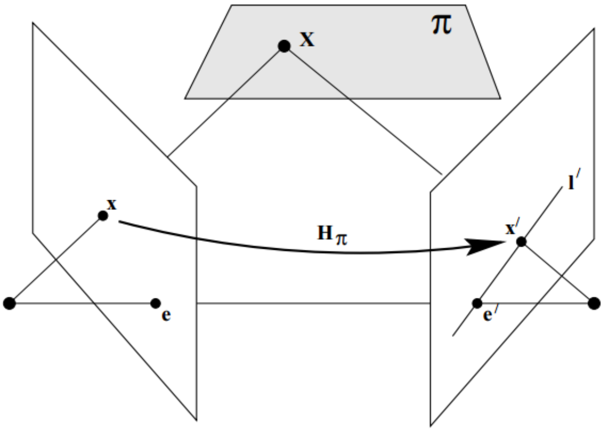


Figure 2.2: A point x in one image is transferred via the plane π to a matching point x' in the second image. The epipolar line through x' is obtained by joining x' to the epipole e'. Credits: [9]

Figure 2.2, shows the typical epipolar geometry for a two-view scenario. The 3D point, X is lying on a plane π in the space and x and x' are the pre-image of point X in left and right image respectively. Since X lies on a ray corresponding to x, the projected point x' must lie on the epipolar line l' corresponding to the image of this ray. The point e and

e' are the corresponding epipoles of left and right image respectively, determined from the point of intersection between the image plane and a line connecting the two camera centres. The left and right image are connected by some rigid-body transformation of rotation R and translation T. Then by the principals of epipolar geometry, we have the following constraints

$$x'\hat{\boldsymbol{T}}\boldsymbol{R}x = 0 \tag{2.7}$$

where $\hat{T}$ is the skew symmetric matrix of size 3x3. This constraint (equation 2.7) is also known as epipolar constraint, which is derived from the triple product of $o\vec{X}, \vec{o'o}, \vec{o'X}$ (complete derivation of this can be found in [9]). The matrix product of $\hat{T}$ and R is also generally referred to as the essential matrix (E = $\hat{T}R$) belonging to the essential space defined as

$$\varepsilon \equiv \{ \quad \hat{\boldsymbol{T}}\boldsymbol{R} \quad | \quad \boldsymbol{R} \in SO(3), \quad \boldsymbol{T} \in \mathrm{I\!R}^3 \quad \} \subset \mathrm{I\!R}^{3x3} \tag{2.8}$$

The camera motion belongs to rigid-body motion SE(3) group and has 6 degrees of freedom (3 for translation and 3 for rotation). Given enough point to point correspondences between image pairs, the essential matrix can be obtained by standard 8-point algorithm [9]. The same concept can be generalized for an uncalibrated camera as well by using the Fundamental matrix as follows

$$\begin{gathered} \boldsymbol{x'}^T \boldsymbol{F}\boldsymbol{x} = 0, \\ \boldsymbol{E} = \boldsymbol{K^T}\boldsymbol{F}\boldsymbol{K} \end{gathered} \tag{2.9}$$

It is important to note that in a real-world application the image data will include random noise and unexpected structures and repeating patterns. This degrades the estimation and will result in outliers, which needs to be filtered out to improve the robustness of the algorithm. A common outlier rejection algorithm such as Random Sample Consensus (RANSAC) [10], can be used to improve the efficiency. Even after the rejection of outliers, the small calculation errors will get accumulated over time and will throw the camera off-trajectory and result in an incorrect map point locations. The bundle adjustment (BA) methods are used to minimize the cost function of the graph and improve the tracking of the system. It was originally conceived for photogrammetry in the 1950s and then later widely adopted in computer vision tasks. The BA algorithm is used to jointly process the 3D pose of the camera and map points to optimize the reprojection cost function (equation 2.11) that best predict the location of the observed points.

$$E(R, T, \boldsymbol{X_1, ..., X_N}) = \sum_{j=1}^{N} |\boldsymbol{\tilde{x}_1^j} - \pi(\boldsymbol{X_j})| + |\boldsymbol{\tilde{x}_2^j} - \pi(R, T, \boldsymbol{X_j})|^2 \tag{2.10}$$

where $\boldsymbol{\tilde{x}_1^j}$ and $\boldsymbol{\tilde{x}_2^j}$ are the observed corresponding point in image first and second respectively. The function π is the reprojection of 3D map point $\boldsymbol{X_j}$ on the image plane and $\pi(R, T, \boldsymbol{X_j})$ denotes the perspective projection after rotation and translation. For a general case of m images, we have

$$E(\{R_i, T_i\}_{i=1...m}, \{\boldsymbol{X_j}\}_{j=1..N}) = \sum_{i=1}^{m} \sum_{j=1}^{N} \theta_{ij} |\boldsymbol{\tilde{x}_i^j} - \pi(R_i, T_i, \boldsymbol{X_j})|^2 \tag{2.11}$$

with $T_1 = 0$ and $R_1 = I$; $\theta_{ij} = 1$ if point j is visible in the image i, else $\theta_{ij} = 0$.

It is often that corner features between images are incorrectly matched, which can affect the robustness of the system. Raul Mur-Artal et al. in ORB-SLAM [11] proposed a parallel computation model to estimate the relative pose between two frames using RANSAC (Random Sample Consensus) approach. RANSAC is used to solve the location determination problem (LDP), where the objective is to determine the points in space that project onto an image into a set of landmarks with known locations [12]. Generally, the BA is used as the last step in the reconstruction pipeline, because the optimization of the pose graph is a highly non-convex problem that requires a good initialization. The minimization of non-convex cost function is a challenging problem and are typically solved by iterative non-linear least-squares algorithms, such as - Levenberg-Marquardt or Gauss-Newton algorithms.

II. GEOGRAPHICAL NAVIGATION SATELLITE SYSTEM (GNSS)

Global Navigation Satellite System (GNSS) is a generic term that refers to any global satellite-based system which can pinpoint the geographical location of a user anywhere on the surface of the Earth. Localizing oneself on the surface of the earth is crucial for many practical applications, like - navigation, mapping, surveying, defence, etc. The GPS technology was developed by the U.S. Department of Defense in 1973 and opened for public use a few years later. Still, it wasn't until 1995 that the system was able to reach a fully functional state. The GPS consists of up to 32 medium Earth orbit satellites in six different orbital planes, with the exact number of satellites varying as older satellites are decommissioned and replaced. It works by receiving radio signals from satellite containing the ephemeris data, which is used to estimate the position of the receiver. As of July 2020, there exist six different satellite constellations - the United States' Global Positioning System (GPS), Russia's Global Navigation Satellite System (GLONASS) and China's BeiDou Navigation Satellite System (BDS) are fully operational, with the European Union's Galileo, Japan's Quasi-Zenith Satellite System (QZSS) and India's Indian Regional Navigation Satellite System (IRNSS) in partial operation and different dates for becoming fully operational systems. If the GNSS receivers supports multiple constellations, it may be possible to improve the positioning in terms of accuracy and speed due to more satellites being available. The accuracy of measurement can typically vary from few tens of meters (in case of normal GPS) to few centimetres (in case of Differential GPS) for real-time systems [13].

The receiver's location is calculated from the distance measurements (pseudo-range) between the receiving antenna and the position of geosynchronous satellites in orbit. Instead of triangulation, we use trilateration (measuring distances, see figure 2.3) which requires at least three satellites to estimate the position. However, in a practical situation, these measurements are always noisy, and we need at least four satellites to be able to accurately determine the receiver's position. As the radio signals travel from satellite, it is prone to many sources of errors, like - ionospheric delay, time relativity, clock bias, multipath reflections, etc. which needs to be taken into account. For a precise measurement of the position, we need to model all these sources of errors and apply it to pseudo-range measurements at the receivers end, which evaluates the satellite signal and processes the ephemeris data, subsequently.

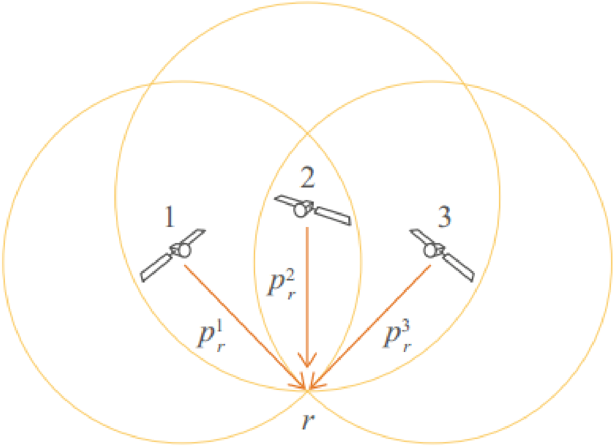


Figure 2.3: Intersecting sphere in trilateration of position. Image credits: [14]

For a code-based pseudorange (PR_c) we have the following (non-linear) equation

(2.12) taking into account the satellite clock bias (dt^S), the delay caused by the ionosphere (d_{ion}), the delay caused by the troposphere (d_{trop}) and the receiver noise (ϵ).

$$PR_c = \rho + \delta t_R - \delta t^S + d_{ion} + d_{trop} + \epsilon \tag{2.12}$$

The above equation is non-linear, because of the geometric distance (ρ) between the receiver and the satellite. Since we have the approximated position of the receiver ($X_0Y_0Z_0$), we can linearize it from the first-order Taylor series expansion. This pseudorange (PR) information is required by the PVT solution, which provides the user's position and time anywhere on the globe.

The PVT estimators are algorithms which take as input satellite positions and pseudorange to those satellites and aim to estimate the receiver's Position, Velocity and Time (PVT). In some cases, we can use the signal characteristics to improve the estimation of the receiver's velocity (e.g. using Doppler measurements), thus increasing the accuracy of the position estimations. Additionally to the position related parameters, we also need to estimate the receiver clock bias with respect to a certain GNSS time frame. This is handled by having the clock bias as one of the parameters to be estimated alongside with the position and velocity.

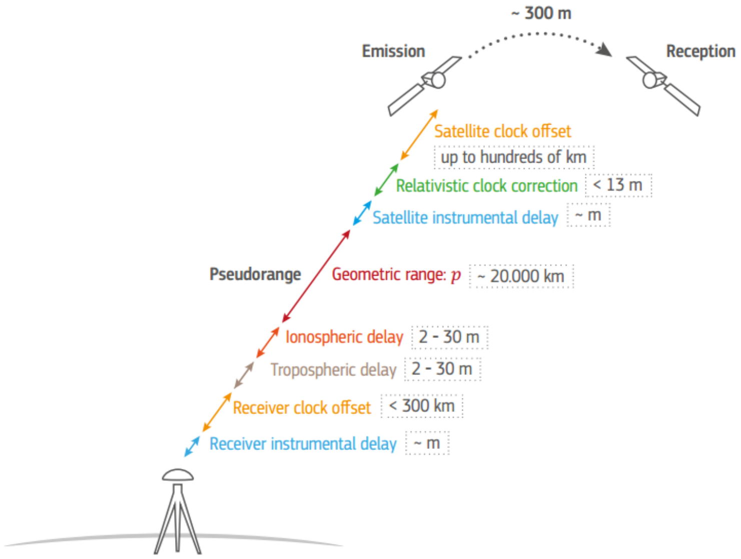


Figure 2.4: Sources of GNSS pseudorange measurement errors in an open area. There would be additional reflection and refraction errors in urban environments. Credits: [14]

The satellite signal is subject to various sources of error before it reaches the receiver (see figure 2.4). These error sources degrade the signal, resulting in poor accuracy of the pseudorange and position estimate. The errors can be systematic in that the error resulting from these sources are more or less a constant bias and the effect persists over a longer period of time, or they can be random sources of error, contributing to signal noise and changing rapidly. Various methods and models are used to mitigate these effects, to a varying degree of success depending on their environment.

- **Multipath errors**: The environment factors such as buildings, mountains, or dense foliage at the receivers' end might reflect or diffract the incoming signal. This type of errors are called multipath errors and is very common to experience in an urban area or tropical forests. There are different methods to reduce or detect the multipath errors. One of the ways is to use antennas that only record incoming signals from where they are expected to arrive, mainly from the sky at angles near or above the horizon, and not from below.

- **Geometric Dilution of Precision (GDOP)**: The geometry of how the satellites are arranged in their orbits relative to the receiver has an effect on the position estimate on the Earths surface. Satellites clustered together in space will yield nearly same pseudorange estimates and will not provide necessary precision. Increasing the number of satellites in sensors view can improve the GDOP value, either by using more than the four required satellites to solve the position equations or allow the receiver to select a more favourable subset of the satellites.

- **Ephemeris errors**: The satellites are affected by solar radiation and the gravitational pull of the sun and the moon, gradually altering their orbits. These changes are continually observed by the monitoring stations on Earth and any model-based corrections of the orbit is updated to the satellites ephemeris data. However, the average pseudorange error due to ephemeris prediction error is 0.8 meters for real-time operations.

- **Atmospheric errors**: Environmental factors like the refraction index or airmass of the medium, that the signal must pass through effects the speed of the satellite signal. The amount of airmass is greater for satellites nearer the horizon relative to the receiver, compared to satellites at larger angles. Provided a rough position of the receiver and some meteorological parameters, this error can be modelled and largely compensated for in the pseudorange estimation.

- **Satellite clock errors**: Even though the atomic clocks onboard the satellite is extremely accurate, but due to small drifts and relativistic time effects, it needs to be accounted and modelled for the estimation. The residual clock error after correction at the receiver ranges from 0.3-4 meters depending on satellite and time since the last clock drift update.

The precision of the GNSS receiver will also vary depending upon their build type and cost. Some of the high-end receivers such as - Trimble AgGPS 332 or iNAT receivers can be accurate up to centimetre level of accuracy, while the low-cost GNSS receivers such as those installed onboard mobile devices can vary somewhere in the range between 4.9 m to 100 m [14]. It is not necessary that the GNSS receiver will support all the active satellite constellations. Like in this project, my Android GNSS receiver supports only GPS and Galileo constellation of satellites. This may affect the coverage or the availability of service in some locations.

III. OUTDOOR AUGMENTED REALITY

Since the time augmented reality was first conceptualized by Wellner et al. in 1993 [15], there has been tremendous development in the field over the last three decades. This progress has led to the development of multiple user-friendly AR systems like - mixed reality-based head-mounted displays (for example - Microsoft Hololens, Magic leap, etc.) and smart wearable technologies (like - Google Glass, Vuzix blade, etc.). Despite the progress, the field of augmented reality still struggles with technical challenges for making a compelling use case for any real-time outdoor applications. Some of the main challenges for AR pointed out by DWF Van Krevelen and R Poelman in [16] are - Portability and outdoor use; Tracking and (auto)calibration; Display fidelity; Depth perception, etc. The tracking of an AR system in an unprepared environment remains a challenge. The mobile AR devices have a limited amount of visual and tracking with limited computational power, which are unsuited for real-time outdoor applications. Mobile AR systems for large scale outdoor environment still remains an open challenge and an area of active research. In this research, we will restrict the scope to the outdoor tracking with geo-referenced tracking systems only.

Feiner et al. developed one of the earliest geo-referenced AR system, it was developed in 1997 at Columbia University to aid tourists in exploring urban environments [17]. Their AR system comprised of a backpack with a computer, a GPS receiver, a pair of see-through goggles for the display with an inbuilt IMU sensor, and a hand-held 2D pad with stylus for interacting with the system. The systems display the information about the points of interest, by overlaying the text labels on the see-through glasses. The text labels are positioned corresponding to the geo-location and aid the user in navigating to these locations. Another AR system similar to this was created and tested by Behzadan et al. [18][19] at the University of Michigan for the visualization of the construction work-flow. Initially, the AR system used a standard positioning service (SPS) coupled with only a gyroscope for the attitude solution but later was upgraded with differential GPS (Trimble AgGPS 332 Receiver) and a full inertial navigation system (INS). Although these papers provide designs for setting up an outdoor AR system, but requires a lot of costly hardware equipment and doesn't provide quantitative analysis or the performance results of the system. Most of this can now be substituted with mobile computing resources and advanced localization algorithms.

Roberts et al. at the University of Nottingham, developed a mobile AR system for the visualization of subsurface and under-ground structure such as - cables, pipes, and other utility lines [20][21]. The proposed solution uses kinematic GPS (RTK GPS) and INS data for precise localization and orientation of the AR system. The paper claims to achieve centimetre level precision, however, no details of the implementation or test results have been provided or accessible at the time of this writing. Also, the hardware requirement of their system is similar to the ones used by Feiner et al. and Behzadan et al. and it strictly depends upon the GNSS measurements, which means it can not be used in GPS denied conditions. The stability of the solution is also questionable for long period usage and user dynamics are not clear from the report.

IV. VSLAM SYSTEMS

The concept of SLAM was first proposed by Randall et al. in 1986 [22]. In the early years, filtering-based approaches were popular and used to process visual measurements. For example, Davison et al. proposed MonoSLAM in 2003 [23], in which an extended Kalman filter (EKF) was used to filter the camera pose and 3D map points (MPs). It was the first real-time monocular SLAM system, which was seminal in the formulation of the problem and basis for much of the later research. After that, G. Klein and D. Murray presented Parallel Tracking and Mapping (PTAM) in 2009 [24]. PTAM used a two threaded system for parallelizing the task of mapping and tracking in different threads for estimating the ego-motion of the handheld camera in real-time. For the optimization of camera poses and map points, the PTAM used the bundle adjustment approach after being inspired by its success in structure from motion (SFM) and visual odometry problems. The bundle adjustment optimizes the camera poses and 3D map point positions jointly by minimizing the reprojection error function. The standardization of SLAM problems and concepts like keyframe structure and optimization using bundle adjustment led to the further development in the field resulting in the current state of the art VSLAM systems, such as ORB-SLAM [11], Direct Sparse Odometry [25], LSD-SLAM [26].

For the global positioning of the AR system, a majority of the previous work is focused on the integration of global sensor devices with the vision-based systems for navigation. These visual navigation systems can be broadly categorized under two categories of - filtering based methods [27][28] and graph-based methods [29][30][31]. While both the approaches are capable of optimizing the localization and mapping problem, the graph-based approach is considered to be superior [32]. The sliding window approach of graph-based method allows it to easily integrate outdated measurements and perform delayed data association, and optionally output lagged pose estimation. Experimentally as well, the graph-based localization gives higher accuracy compared to particle-based methods. This motivates our approach of using the graph-based optimization strategy for the pose-graph problem and fusion of GNSS measurement.

The state of the art AR API (Application Programming Interface) frameworks, such as - Google ARCore, Apple ARKit, and Microsoft Hololens are available for mobile AR systems. However, they all have limitation of localization in a large scale environment. Tobias Feigl et al. in [33] concludes that the SLAM techniques of these AR systems are a showstopper for an area of as small as 1600m^2 or straight distance of 120m. Tobias Feigl et al. points out two typical problems of these AR systems: (a) They tend to crash when a user moves while a surrounding object, e.g., a vehicle, synchronously moving with the camera. (b) There are system instabilities when workers randomly enter or leave the field of view of the AR system and temporarily occlude known features (as the system can hardly distinguish between self-motion and feature motion). The study shows that AR systems accumulate a linear scaling error of 6.65cm per meter, on average. The best system only yields the mean absolute error of 17.28m per 120m of distance and scaling error of up to 14.4cm per meter, which is clearly too much for a stable AR experience.

V. GEO-REFERENCED VSLAM SYSTEMS

Daniel Kiss-Illes et al. [34] proposed a GPS-SLAM system that combines both the GPS and IMU measurements to estimate the relative sensor poses to be used in ORB-SLAM [11]. The GPS-SLAM system implements a constant velocity model to estimate the relative camera position and augments the existing camera poses in the local map to improve the tracking. This resulted in fewer lost frames and a denser map reconstruction. However, it also resulted in poor tracking in case of a larger rotation and used a combined GNSS-IMU measurement to estimate new camera poses for correction.

Most of the fusion of GPS/GNSS based inertial navigation system involves the integration of IMU data with the visual SLAM. This is mainly due to the high-frequency input of the IMU data which can match the high frame rate of the visual sensors. The IMU data provides relatively higher measurement accuracy compared to RAW GNSS measurements over a shorter period. While some have even suggested the coupled visual SLAM and inertial navigation as an alternative to GPS based navigation [35]. In our case, we want to avoid the processing of IMU and use the RAW GNSS measurements only for a tightly coupled with visual SLAM system.

Xiao Chen et al. proposed a tightly coupled system of low-cost GNSS and monocular camera for global localization of the sensor [36]. Their proposed system GNSS-Visual-ORB-SLAM (GVORB) works by the data fusion into SLAM graph-based optimization instead of a filter-based approach. In original ORB-SLAM [11], it has three processing threads for - tracking, local mapping and loop closing. In addition to these threads, GVORB system uses two additional threads for big local bundle adjustment (BLBA) and global bundle adjustment (GBA). The BLBA is used to integrate GNSS measurement into local bundle adjustment optimization and GBA is used to optimize the keyframe poses and map points with all the GNSS measurement and visual observations. The GVORB system has an average positioning error of approx. 5 meters for 2 minutes of an online system. Although their work seems promising on the surface, the paper doesn't properly explain the characteristics of new systems and the testing results are calculated from a synthetic GNSS data with random generated Gaussian noise.

Tong Qin et al. in [37] presents a more general approach for fusing multiple sensor data into visual SLAM process. The paper proposes an integration framework for the fusion of global sensors (i.e. GPS, magnetometer and barometer) and local sensors (i.e. camera, IMU and LiDAR) measurements in a pose graph optimization step of the SLAM process. The different properties of global and the local sensors help in overcoming the shortcomings of other sensor measurements, which helps in creating a more robust localization and mapping system. Their localization results are impressive and better than ORB-SLAM on the publicly available dataset of KITTI sequence [38]. However, we can not directly use their approach as it relies upon multiple sensor inputs, and the pose graph optimization with just the GNSS measurement wouldn't be sufficient for global convergence of the system.

Our approach of GNSS fusion in this project is largely based on the framework proposed by Tong Qin et al. in [37]. Along with the updated global bundle adjustment for the GNSS fused pose graph, this project presents a strategy for estimating map scale for monocular VSLAM. Further, we propose methods for the alignment of GNSS and VSLAM coordinate systems and scaling of the global map without the loop closure detection.

3. ANALYSIS AND DESIGN

From the background research, it became clear that there doesn't exist any solution/research that allows for stable geo-referenced localization in real-time which is suitable to create an augmented reality experience, more so to create a collaborative experience between AR and VR platform. Thus, this research may be considered as the beginning step to combine the state of the art systems from the field of robotics and sensor fusion, to create a consistent platform that accurately tracks the user on geo-referenced scale and support heterogeneous platforms. Given the ambitious objective of the research and some preliminary proof of concepts of the platform, I had to limit the scope of this project and still have a meaningful contribution towards the larger objective.

Due to the nature of the research and time constraints, I have focused primarily on the objective of tracking and localization of the camera using visual SLAM and GNSS measurement. Although this project does not directly implement any augmented reality system, the selection of tools and libraries are strictly in line with the development of the larger system. Similarly, the approach of tightly coupled GNSS and visual SLAM is to optimize the computation cost and save on performance to be compatible with the mobile devices. For the same reason of performance, the integration of any inertial measurement data is not considered.

The geo-referenced position of the system is obtained by the processing of satellite's ephemeris data by the GNSS receiver. For the processing of ephemeris data, we are using the open-source project called GNSS compare [39], which was developed as part of the European Space Agency (ESA) competition called Galileo Smartphone App Challenge. The satellites' ephemeris data contains - Eccentricity; Semimajor axis; Inclination; Longitude of the ascending node; Argument of periapsis; and True anomaly. The android application currently supports Galileo and GPS satellite constellations for the estimation of Position, Velocity and Time (PVT) of the system. It then estimates following GNSS measurements - timestamp; latitude; longitude; altitude; speed; accuracy; the number of visible satellites; multipath indicator; accumulated delta range uncertainty in meters; and type of constellation. The position updates, speed and range uncertainty measurements are shared with the VSLAM system over a TCP-based connection.

There are many formats for representing geographical coordinate system and geodetic datum for defining the position of the system. For the input and output coordinates, I have used WGS-84 standard [40], as specified by the United States Department of Defence. This coordinate system is most commonly used in the satellite-based navigation, geodesy and cartography including GPS. Although global, this coordinate system is not suitable for directly integrating in visual SLAM system. The VSLAM world is a Euclidean coordinate system with orthogonal basis vectors representing a flat/planar world. Thus, normally the GPS coordinates are converted to Earth Centered Earth Fixed (ECEF) coordinate system or Universal Transverse Mercator (UTM) coordinate system to use in 3D applications. I have decided to use the UTM coordinate system, as it utilizes a conformal projection of the global map onto a planar surface by dividing the surface into local projections called "grid zones". Unlike ECEF, the UTM doesn't involve spherical mapping and the transformation between the SLAM world coordinate system and UTM can be easily defined. The advantage of the UTM coordinate system is that the distances and angles can be calculated using Euclidean geometry over short distances and the distance units are in meters, which makes it intuitive to interpret.

The current state of the art visual SLAM systems [11][26][25] can accurately map and localize the camera using local sensors (like - camera or IMU) while global sensors (like - GNSS or magnetometer) can provide locally noisy but globally drift-free positions. The complementary nature of these sensors makes it ideal for them to be combined, to correct for the uncertainty of the other sensor. Fusing the measurements from these sensors can help in achieving geo-referenced 6 DOF (degree of freedom) pose estimation. Although, the conventional filter-based methods can fuse the local estimates into the global pose graph required for the convergence, but is not ideal for performance when compared to graph-based BA. The cost of BA is linear in N, whereas the cost of filtering is cubic [41], owing to the fact uncertainties are explicitly represented in filtering-based methods. For filtering-based method in SLAM, the computational cost of propagating joint distributions scales poorly with the number of variables involved during optimization, and the number of features in the map will be severely limited. Whereas the graph-based method retains poses to be in a sliding window of the most recent camera positions, which are a set of intelligently or heuristically chosen keyframes. The graph optimization remains relatively efficient, even if the number of features in the graph and measurements from the keyframes is very high. That is why graph-based methods are preferred over filter-based method for this type of optimization tasks.

For visual SLAM, I have used OpenVSLAM project, developed by Shinya Sumikura et. al. in 2019 [5]. OpenVSLAM project provides a visual SLAM framework that builds on top of the existing state of the art systems used in SLAM, like - using ORB (Oriented FAST and Rotated BRIEF) [6] for feature detection and tracking; DBoW2 (Dynamic Bag of Words) [42] for indexing and converting the images into a bag-of-word representation; G2O (General Graph Optimization) [43] for optimizing graph-based nonlinear error functions. Figure 3.1 shows the OpenVSLAM architecture with all the functional blocks included in three threads of tracking, mapping and global optimization that run in parallel.

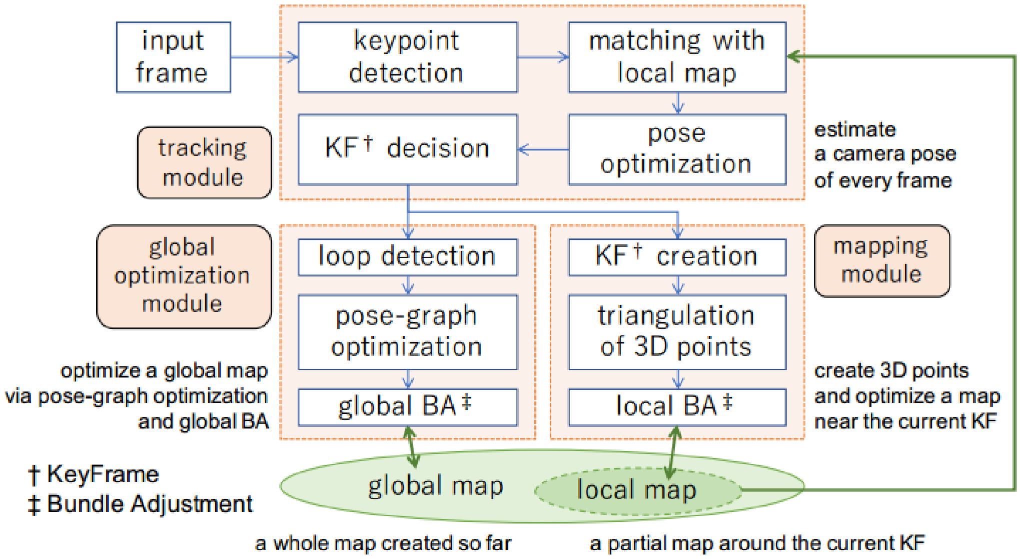

input frame
keypoint detection
matching with local map
tracking module
KF † decision
pose optimization
estimate a camera pose of every frame
global optimization module
loop detection
pose-graph optimization
global BA ‡
KF † creation
triangulation of 3D points
local BA ‡
mapping module
optimize a global map via pose-graph optimization and global BA
create 3D points and optimize a map near the current KF
† KeyFrame
‡ Bundle Adjustment
global map
local map
a whole map created so far
a partial map around the current KF

Figure 3.1: OpenVSLAM architecture. image credit: [5]

There are three different stages of optimization for the global convergence of the system. The first optimization is done, at the frame level in the tracking thread (referred to as the pose optimization in Fig 3.1). The camera pose optimization matches the current frame with the previous frame and optimizes the pose using motion only bundle adjustment (BA). The second stage of optimization is done in the local BA step under the mapping module. This optimization is done, over the connected neighbouring keyframes that have shared landmarks in the field of view. The new correspondences for unmatched ORB in current keyframe are searched in the neighbouring keyframes from the covisibility graph to triangulate new map points. This step achieves an optimal reconstruction of the environment and camera poses using the local BA. The optimization of local bundle adjustment is contained within the $SE(3)$ group, and hence the camera pose is restricted to 6 DOF (Rotation and Translation). The third optimization is the global BA step under the global optimization module. The global bundle adjustment is used when a loop closure is detected in the new keyframe. A similarity transform $SIM(3)$ is calculated to estimate the accumulated drift of the system.

$$Sim(3) = \left\{ \begin{bmatrix} s\boldsymbol{R} & \boldsymbol{T} \\ 0 & 1 \end{bmatrix} with \quad \boldsymbol{R} \in SO(3), \quad \boldsymbol{T} \in \mathbb{R}^3, \quad s \in \mathbb{R}_+ \right\} \tag{3.1}$$

Compared to local BA the global BA is optimized in 7 DOF (Rotation, Translation and scale). This results in a drift aware and scaled global optimization of the graph in the VSLAM system.

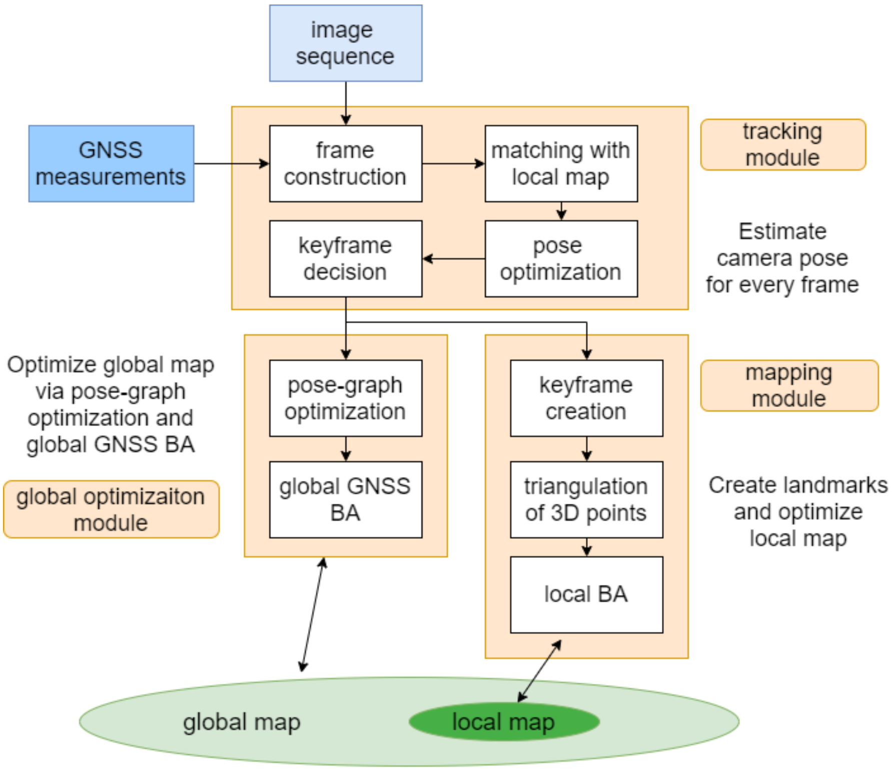


Figure 3.2: Modified VSLAM system architecture

For a standalone VSLAM system, the loop closing is an important step to correct for the accumulated drift over time. However, it’s not useful in case of a real-time application that doesn’t necessarily involve moving in a closed path. In our case, it is assumed that the mobile user may arbitrarily walk and not necessarily revisit the same place again. Thus I have replaced the global bundle adjustment step containing the loop closure test with the global GNSS bundle adjustment process (see figure 3.2). The global GNSS BA is called every time after the addition of a fixed number of keyframes (30 keyframes in our case) to the global map. The GNSS measurements are used to complement the systems drift and scale the global map accordingly.

In the current design, the complete system includes a GNSS receiver installed on-board the android device, Logitech C920 webcam and a windows laptop with i7 2.6 GHz (8 core CPU). The android device is connected with the PC via a USB cable, and a socket-based TCP connection is established between the android application and SLAM system on the PC. The Android application handles the processing of the ephemeris data and support for satellite constellations, and the GNSS measurements are sent over the socket connection to the client application. This system design was opted to be inline with the larger goal of testing the solution on mobile. Most of the current mobile phones that supports AR framework are already equipped with a high-resolution camera and GNSS receiver with onboard computation power equivalent to 2+ GHz octa-core processors.

4. IMPLEMENTATION

This chapter explains about the implementation of the core systems and presents the formulation of the problem.

I. TIGHTLY COUPLED GNSS AND VISUAL SLAM SYSTEM

For a tightly coupled GNSS-VSLAM system, the carrier phase or pseudo-range measurement is directly fused with the visual information. In the tightly coupled system, even if only one satellite is tracked the system can still make use of the GNSS information [44]. While in a loosely coupled system, the position calculated by the GNSS receiver is fused externally with visual measurements as proposed in [36] and [37]. The tightly coupled system make sense in our case for two reasons - first to get a globally drift free localization without using the loop closure; and second, it doesn't require additional processing step for the fusion and optimization, thus saving on performance.

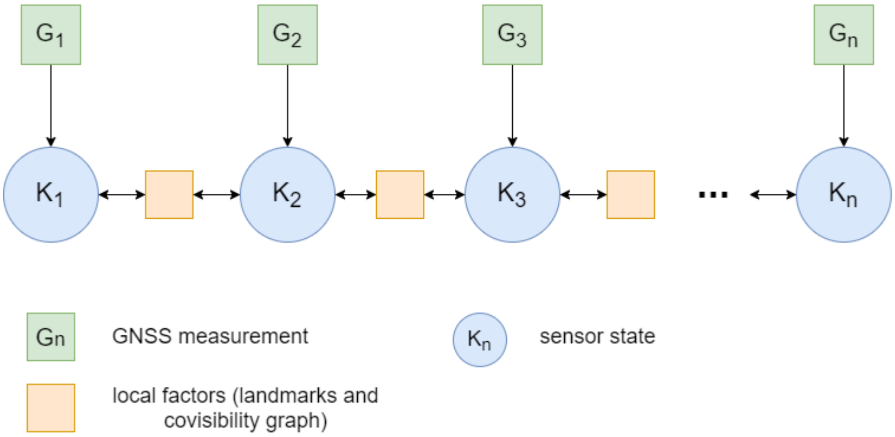


Figure 4.1: GNSS measurement in global pose-graph structure

The most of the visual SLAM process is same as implemented in OpenVSLAM [5] and is already discussed briefly in the introduction. The major change from regular VSLAM is the global GNSS bundle adjustment step under the global optimization module, instead of global bundle adjustment 3.2. The GNSS BA has two main jobs - first to estimate and scale the map if necessary, and second to optimize the global reprojection error with the GNSS measurement as a prior unitary edge of the graph. Figure 4.1 shows the abstract representation of GNSS fused global pose graph structure in the global GNSS BA. The keyframes and the local connectivity of the graph is estimated from the local BA and stored in the local map database with the covisibility graph. This connectivity and landmark points are collectively represented by the orange square. The blue spheres represent the sensor state, that contains the estimated pose (translation and rotation) of the sensor in the SLAM world coordinate system. Each edge of the graph has the cost function and constraints the relative pose from one node to another.

The first concern in the fusion of GNSS measurement is the low frequency updates ($\approx$ 1Hz), while the VSLAM system runs at roughly 10 frames per second (in case of OpenVSlam). To get the GNSS measurement for every frame, I have used a linear interpolation of the measured GNSS position. Thus to estimate GNSS position at time t for the received measurement at time t_1 and t_2, we can linearly interpolate as follows

$$\begin{aligned} \delta t &= t_2 - t_1, \\ \delta g &= G_{t2} - G_{t1}, \\ G_t &= G_{t1} + \frac{(t - t_1)}{\delta t} \delta g \end{aligned} \tag{4.1}$$

here G_{ti} represents the GNSS measurement at timestamp t_i where $t_1 <= t_i <= t_2$.

The global pose graph optimization is a Maximum Likelihood Estimation (MLE) problem, which consists of the joint probability distribution of sensor poses. This problem can be formulated as follows - given a good initial estimate of sensor poses $\chi = \{k_0, k_1, ..., k_n\}$, where $k_i = \{{p_i}^w, {q_i}^w\}$ and ${p_i}^w, {q_i}^w$ denotes the position and rotation under the SLAM world. With the assumption that all the measurement probabilities are independent, the problem can be derived as following

$$\hat{\chi} = \underset{\chi}{\operatorname{argmax}} \prod_{t=0}^{n} \prod_{k \in S} p(z_t^k | \chi), \tag{4.2}$$

where S is the set of measurements including the local factors and GNSS measurements. The uncertainty of measurements are assumed to be Gaussian Distribution with mean and covariance, that is $p({z_t}^k | \chi) \approx N({\hat{z}_t}^k, \Omega_t k)$. Thus the equation 4.2 can be derived further as follows

$$\begin{aligned} \hat{\chi} &= \underset{\chi}{\operatorname{argmax}} \prod_{t=0}^{n} \prod_{k \in S} exp(-\frac{1}{2}||z_t^k - h_t^k(\chi)||^2_{\Omega_t^k}), \\ &= \underset{\chi}{\operatorname{argmax}} \sum_{t=0}^{n} \sum_{k \in S} ||z_t^k - h_t^k(\chi)||^2_{\Omega_t^k} \end{aligned} \tag{4.3}$$

where $||z_t^k - h_t^k(\chi)||^2_{\Omega_t^k}$ is the Mahalanobis norm and converts the state estimation to a non-linear least squares problem. This is solved using graph-based bundle adjustment process.

The UTM coordinate can be represented by a 3D vector for Easting, Northing and Altitude of the system. Here, the system position is defined relative to the starting point, i.e. the first UTM coordinate is fixed as the origin point. Thus, the relative position and error function is defined as follows:

$$\begin{aligned} P_t^{GNSS} &= U_t^{GNSS} - U_0^{GNSS} \\ z_t^{GNSS} - h_t^{GNSS}(\chi) &= z_t^{GNSS} - h_t^{GNSS}(x_t) \end{aligned} \tag{4.4}$$

here, U^{GNSS} is the recorded measurement and $h_t^{GNSS}(x_t)$ is the transformed GNSS position vector in SLAM world coordinates. The GNSS measurements directly constrain the camera pose of every node, and the covariance is set from the accumulated delta range uncertainty (in meters) received from the GNSS sensor. The range uncertainty is inversely proportional to the number of active satellites. Larger the number of visible satellites, smaller is the covariance.

Once the pose graph is constructed, the Levenberg-Marquardt algorithm is used in an iterative way to optimize the graph. The goal of the optimization is to estimate the best node configuration that minimizes the cost function and matches the graph edges as much as possible.

After the optimization, we can transform the estimated camera poses from VSLAM world to GNSS coordinate system. This way we can get the geo-referenced positioning of the camera from the VSLAM system. We can use a large number of keyframes for the global pose graph optimization to get an accurate estimation, however, the computation complexity increases with the large number of keyframes. Thus here I have maintained a sufficiently large window for optimization and discard the old poses and measurements from global bundle adjustment. I have used the last 100 keyframes from the global map for global pose-graph optimization.

An important step before the global GNSS BA is the scaling of the global map. The test for map scale is done after a fixed number of keyframes have been added to the map database. This limit is set to 25 keyframes in this project. The above-mentioned pose graph optimization is constrained within the $SE(3)$ space with 6 DOF (position and rotation). To fix the scale ambiguity, we need to estimate the map scale from external sensor measurements, which in this case is GNSS data. The code snippet: 8.2 provides the algorithm used for the estimation of scale factor from the "gps_pos" and "cam_pos" list for all the keyframes used in current global optimization.

Listing 4.1: Estimation of scale factor for global map

```
Vec3_t mean_gps;
Vec3_t mean_cam;
mean_of_eigen_vec(gps_pos, mean_gps);
mean_of_eigen_vec(cam_pos, mean_cam);

double sum_gps_diff = 0.0;
double sum_cam_diff = 0.0;
for (size_t i = 0; i < gps_pos.size(); ++i) {
    sum_gps_diff += (gps_pos[i] - mean_gps).squaredNorm();
    sum_cam_diff += (cam_pos[i] - mean_cam).squaredNorm();
}
const double scale = std::sqrt(sum_gps_diff / sum_cam_diff);
```

The estimated scale factor is used to uniformly scale the global map by multiplying it with all the keyframe and map points from the global map database.

II. COORDINATE SYSTEMS AND INITIALIZATION

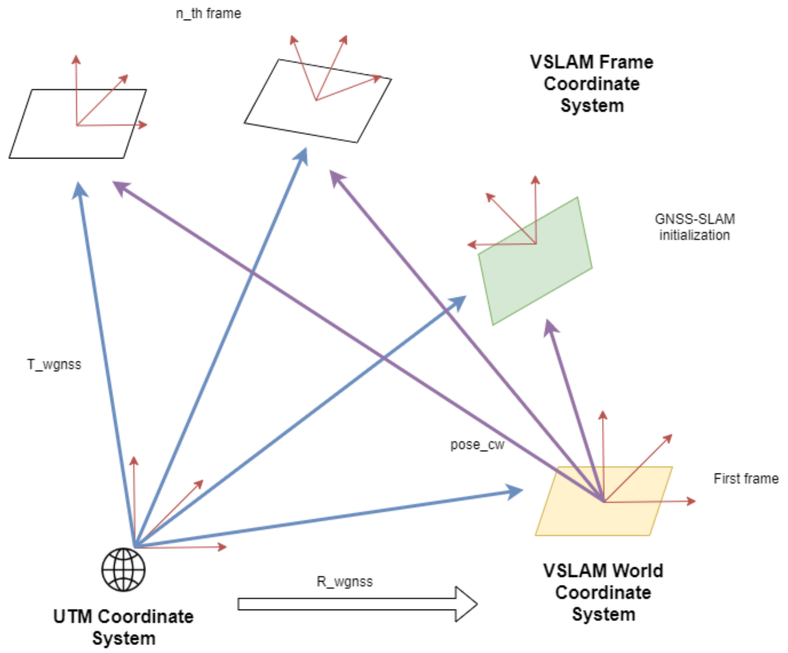


Figure 4.2: Different coordinate systems in the GNSS-VSLAM system

In the combined GNSS and VSLAM system there are 3 different coordinate systems in place (see figure 4.2). In order to get the geo-referenced position using VSLAM, we need to estimate the transformation between the VSLAM coordinate systems and the UTM coordinate system. The first is the UTM coordinate system, which is defined for the GNSS receiver and is obtained by converting GNSS measurements to UTM coordinates. The second coordinate system is the VSLAM world coordinate system, which is defined for the first frame of the image sequence that is used to initialize the tracking of the VSLAM system. The third coordinate system is defined for the individual/continuous frames of the VSLAM system. From figure 4.2, R_wgnss represents the rotational transformation from UTM to VSLAM world coordinate system; T_wgnss represents the transformed GNSS measurement into SLAM world coordinate; and $pose\_cw$ represents the estimated camera pose in the VSLAM world.

The VSLAM world and frame coordinate systems have the same scaling and they differ only in their orientation. The UTM coordinate system has both the different orientation and scale compared to the other two coordinate systems. The VSLAM world coordinate system has its origin at the first tracked frame (represented by the yellow frame in Fig 4.2). This frame remains fixed throughout the process and never gets deleted or changed during the execution. The VSLAM frame coordinate system is defined for the current camera pose and changes with every new frame update. The keyframe poses are always defined in SLAM world coordinate system.

The UTM Easting and Northing measurements from the sensors are mapped to the X and Z values, while the Y-axis is used for representing the height. In the VSLAM world coordinate system, the horizontal movement of the camera is restricted to the XZ-plane, while the vertical movement of the camera is represented on the Y-axis. This commonality between the two coordinate system leads to the hypothesis that the linear transformation can be represented by a 3x3 rotation matrix (R_wgnss) along the Y-axis. As the VSLAM frame coordinate system changes with every new frame, we cannot determine the rotation matrix that describes the general transformation from the GNSS coordinate system to the VSLAM coordinate system. Therefore it only makes sense to have calculated the rotation matrix from the GNSS to the VSLAM world coordinate system. These coordinate systems are fixed, and the resulting rotation matrix can be consistently used for transformation.

The rotational transformation between the two coordinate systems is estimated from two direction vectors representing the sensor position in their local coordinate systems. Once we have enough confidence in the locally estimated direction, we can calculate the rotation matrix between the two coordinate systems that represents the required transformation. Detailed code snippet is included in Appendices 8.1. As the GNSS measurement is susceptible to noise, this initialization step becomes slightly complicated. During the testing, I have constraint the initialization of the GNSS position based on either of following two conditions- (a) either the movement distance is greater than 15 meters (calculated from GNSS positions); (b) at least 30 keyframes have been added to the global map. This rotational transformation is estimated during the GNSS initialization stage, and no global GNSS BA optimization is done without this initialization. Once the transformation is estimated, all the previous GNSS measurements are aligned for the keyframes stored in the global map. The previous GNSS measurement (T_{gnss}) can be converted to SLAM world (T_{wgnss}) as following

$$\begin{aligned} \boldsymbol{T}_{wgnss} &= \boldsymbol{R}_{wgnss}\boldsymbol{T}_{gnss}, \\ \boldsymbol{T}_{gnss} &= \boldsymbol{R}_{wgnss}^{-1} pose_{wc} \end{aligned} \tag{4.5}$$

here $pose_{wc}$ represents the camera position in SLAM world and $\mathbf{R}_{wgnss}^{-1} = \mathbf{R}_{wgnss}.transpose()$ as $\mathbf{R}_{wgnss} \in SO(3)$ group.

Once we have estimated the transformation between the GNSS and VSLAM coordinate system, we can update the previously set GNSS-position of keyframes in the map database. It ensures that we have a consistent GNSS measurements during the global pose-graph optimization after the initialization step.

5. TESTING

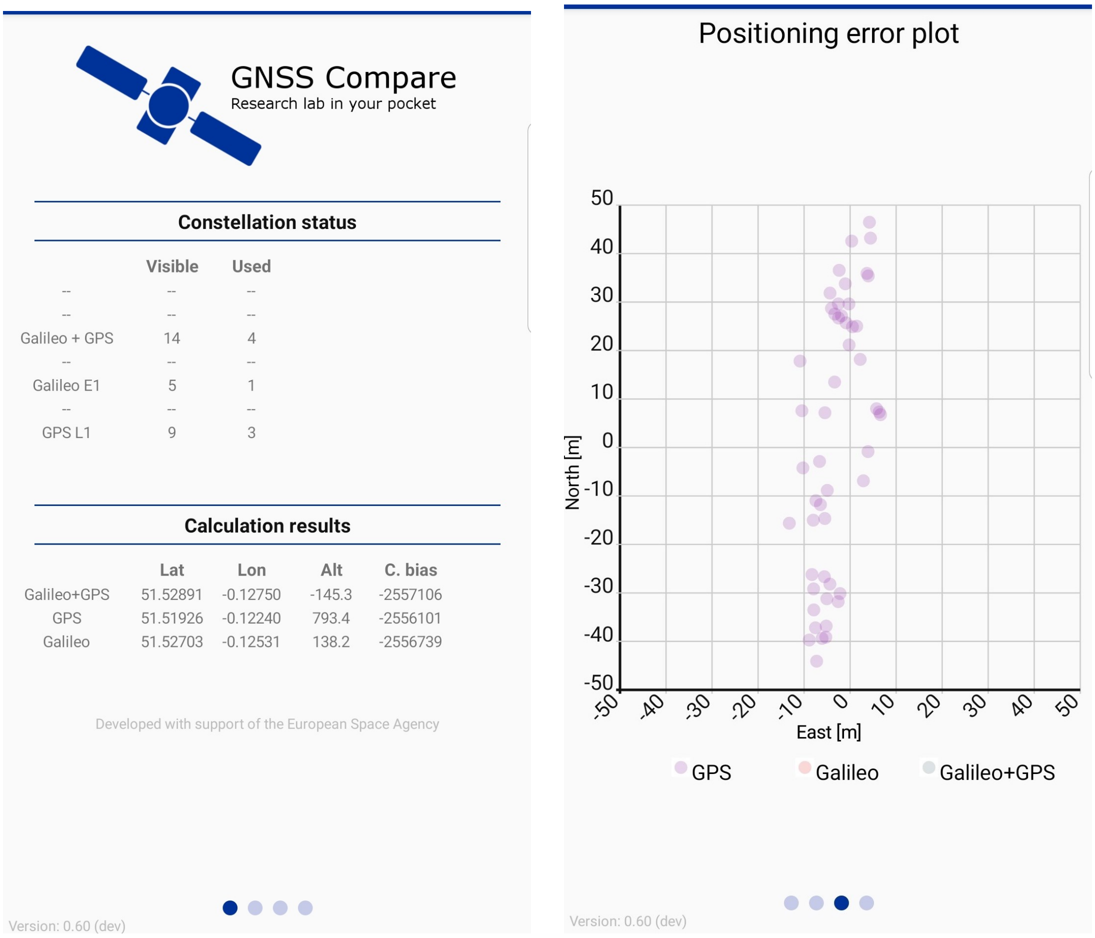
Two systems are created for testing the tightly coupled GNSS VSLAM - online system and offline system. The online system uses a connected live camera and a GNSS server. The received GNSS data from the android device is fed directly into the VSLAM system. The offline system runs with the pre-recorded video data and timestamped GNSS measurement text, stored locally on device.

Two tests were planned, for testing of stability and the accuracy of the GNSS-VSLAM system. The first test is for the correction of noisy GNSS measurement of camera trajectory. As the GNSS measurement is assumed to be a Gaussian distribution with the standard deviation of averaged position error of 9.71 m [45], it cannot be directly used for precise localization of the user. The second test is for the estimation of scale factor for the GNSS-VSLAM system. This test will measure the accuracy of similarity between the physical world and the virtual VSLAM coordinate system of the global map.

To test the GNSS fused VSLAM system, I have captured the video sequence of the environment from my calibrated Logitech C920 web-camera and synchronously recorded the corresponding live GNSS measurements with timestamps. The system's timestamps was saved for every new record. This saved data is then used in offline system, and subsequent results are recorded and presented in the next section. Similarly, for the online system, the complete setup of a laptop with a webcam and a paired android device was carried on the streets, and subsequent input and output records are serialized to the file for evaluation.

Since I have used the low-cost GNSS receiver and mobile system, it's hard to get the ground truth data. For the test of camera trajectory and noise correction of GNSS measurement using VSLAM, we can do a qualitative analysis by manually plotting the estimated positions over the 2D map. We can visually compare the input and output graph plot for smoothness of the traversed trajectory for the camera. For the evaluation of scale factor estimate, we can do the quantitative test by calculating the absolute distance between the starting and ending point of the travelled trajectory and compare this distance in meters for input GNSS measurement and estimated sensor position in GNSS-VSLAM system. I have used Google map to get the ground-truth value for starting and ending location.

Figure 5.1, presents the screenshot of the android server application for GNSS measurement running on a mobile device.



GNSS Compare
Research lab in your pocket

Constellation status

	Visible	Used
--	--	--
--	--	--
Galileo + GPS	14	4
--	--	--
Galileo E1	5	1
--	--	--
GPS L1	9	3

Calculation results

	Lat	Lon	Alt	C. bias
Galileo+GPS	51.52891	-0.12750	-145.3	-2557106
GPS	51.51926	-0.12240	793.4	-2556101
Galileo	51.52703	-0.12531	138.2	-2556739

Developed with support of the European Space Agency

Version: 0.60 (dev)

(a) home page with constellation and GNSS estimate information

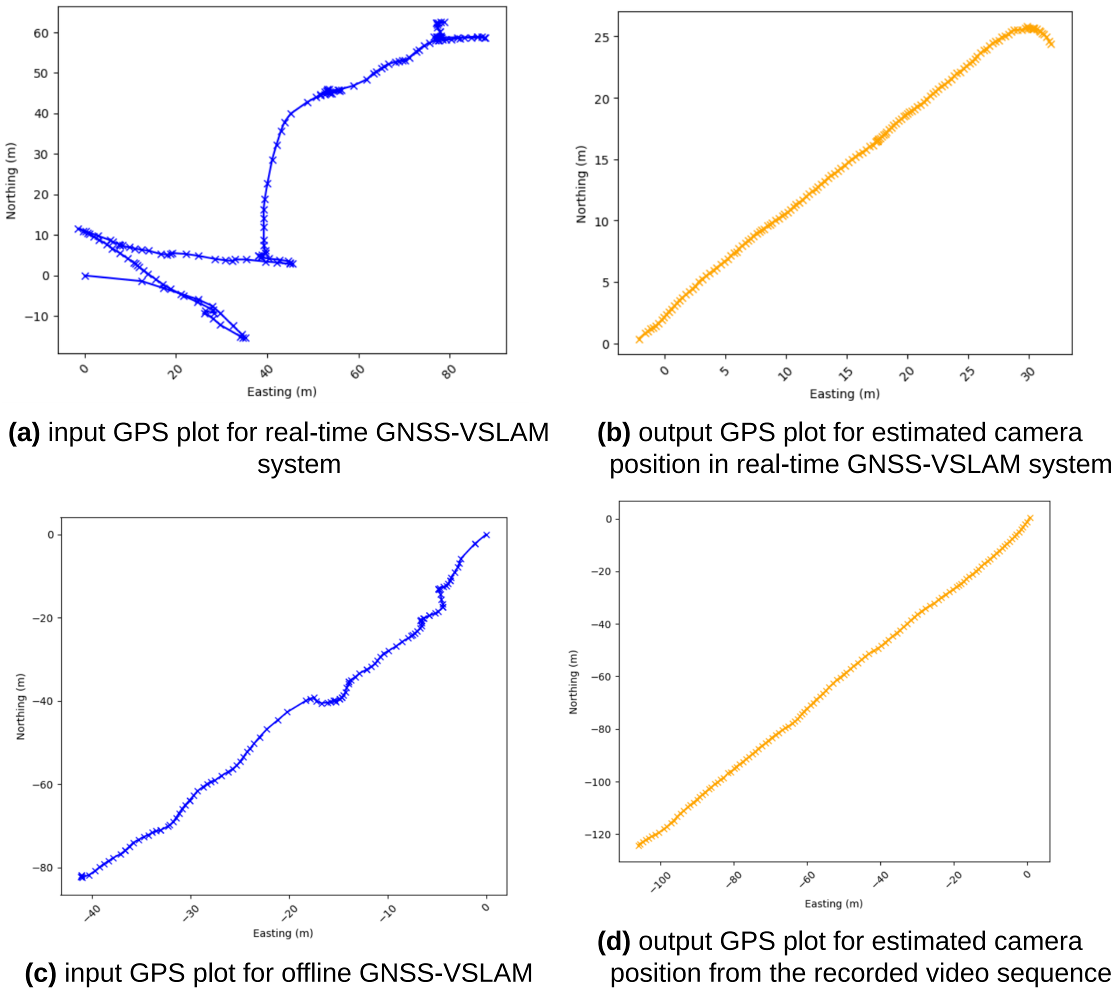
(b) GNSS measurement position error plot, while standing still before the start of testing

Figure 5.1: Android GNSS server application

From figure 5.1b, we can observe that the receiver has a maximum positional error of approx. 50m in the North (z-axis) and approx. 30m in the East (x-axis). Such high variance would increase the chances of poor initialization of rotational transformation and is not ideal for a stable GNSS-VSLAM system. And since no global bundle adjustment is done before the initialization, it would also adversely affect the scale factor estimation immediately after the initialization step.

6. RESULTS

This section presents the qualitative and quantitative analysis of the GNSS-VSLAM system for noise correction and scale estimation respectively.



(a) input GPS plot for real-time GNSS-VSLAM system

(b) output GPS plot for estimated camera position in real-time GNSS-VSLAM system

(c) input GPS plot for offline GNSS-VSLAM system

(d) output GPS plot for estimated camera position from the recorded video sequence in offline GNSS-VSLAM system

Figure 6.1: Graph plot of input (blue line) and output (yellow line) camera trajectory in UTM coordinates relative to starting point

Figure 6.1, presents the graph plot of the input and output GPS trajectory for the GNSS-VSLAM system. The visual observation of the graph shows that the estimated camera trajectory in the output is much more smoother and continuous compared to the input GPS positions. From input graph 6.1a, we can observe how the bundle adjustment process has smoothed out the outlier input observation resulting in smoother output path in graph 6.1b. In case of offline testing, the recorded GPS measurements are relatively smooth in graph 6.1c, but it still contains slight bumps and measurement inconsistencies, which are smoothed out in output graph 6.1d.

For the quantitative test of scale, we got the following results during the online and offline testing of the GNSS-VSLAM system.

	Starting GPS (lat,lon)	Ending GPS (lat,lon)	Distance [meters]
Input	51.528391, -0.126144	51.527671, -0.126782	91.42
Output	51.528395, -0.126134	51.527313, -0.127742	163.86
Ground truth	51.528395, -0.126134	51.527870, -0.127332	101.37

Table 6.1: Input, output and ground truth GPS coordinates in offline testing

	Starting GPS (lat,lon)	Ending GPS (lat,lon)	Distance [meters]
Input	51.527960, -0.126910	51.528494, -0.125741	100.33
Output	51.527964, -0.126941	51.528168, -0.126438	41.54
Ground truth	51.528046, -0.127030	51.528377, -0.126160	70.55

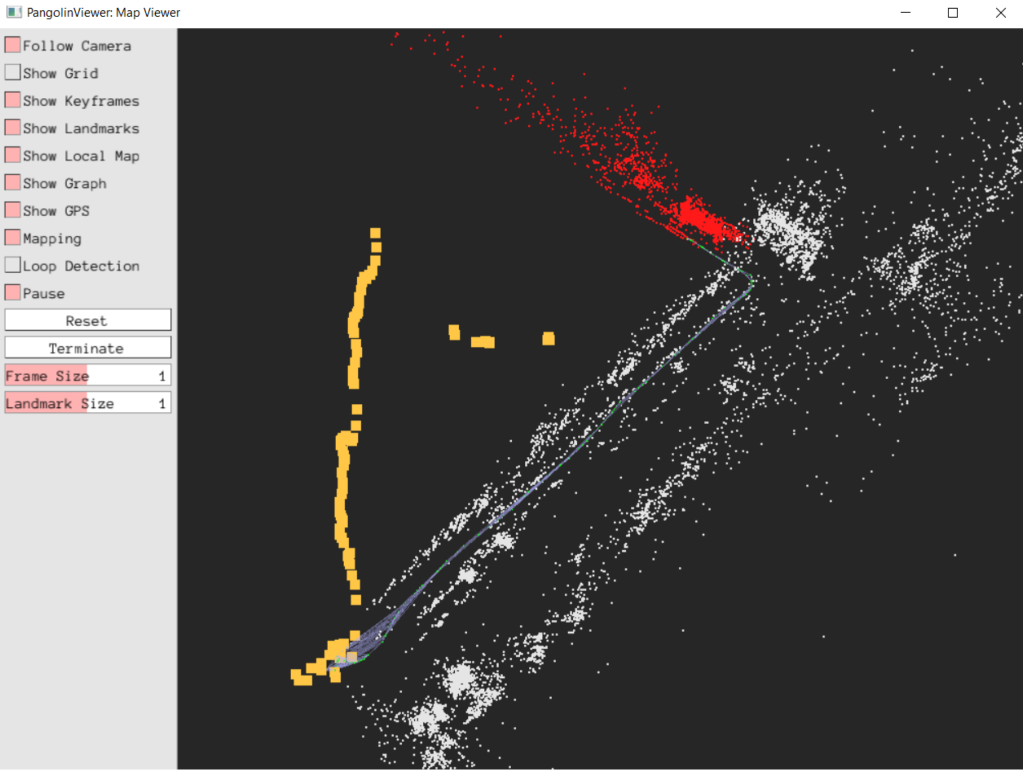
Table 6.2: Input, output and ground truth GPS coordinates in online testing

Table 6.1, shows that the GNSS-VSLAM system overly estimate the world scale, while from Table 6.2, we observer that the scale factor is underestimated. In my understanding following two factors are responsible for this

- The GNSS VSLAM initialization step. The estimation of rotation transformation between GNSS coordinate and VSLAM world coordinate system is highly prone to the noise factor in GNSS measurement.
- The estimation of the scale factor is based on the fact that the GNSS measurements are assumed to be normally distributed (Gaussian). However, that is not true especially in case of a moving robot [46].

In the current estimation of rotational transformation, the system depends upon the sensor distance (of 15m) and the number of keyframes (at least 30) to estimate the GNSS direction vector. The position vector is calculated by subtracting the current

GNSS measurement from the starting GNSS measurement (Appendices 8.1). While theoretically, it is correct but in practice, it almost always results in some amount of angular offset between the alignment of the SLAM world coordinate system to GNSS world coordinate System. Figure 6.2 shows the transformed GNSS positions plot in the 3D SLAM world. While there doesn't exist any elegant solution to automatically address this problem (to my knowledge), this can be fixed by adopting the manual alignment process. Like, placing fiduciary markers in the physical world or to manually define the rotational transformation during the process.



☑ Follow Camera
□ Show Grid
☑ Show Keyframes
☑ Show Landmarks
☑ Show Local Map
☑ Show Graph
☑ Show GPS
☑ Mapping
□ Loop Detection
☑ Pause

Frame Size 1
Landmark Size 1

Figure 6.2: Scaled GNSS-VSLAM system. The red dots represent local map feature point and white dots for global map points; The transformed GNSS measurements are represented in orange; The camera's trajectory is represented in thin blue line.

The incorrect estimation of the scale factor during the global GNSS bundle adjustment process also results in poor localization of the robot over the large distance or for a longer period of time. This severely affects the tracking of the VSLAM system and results in losing track of the map after a certain amount of time. In my testing, I have found that current GNSS-VSLAM system loses track after moving roughly a distance of 100 meters.

7. CONCLUSION

In this project, I have investigated on directly fusing the GNSS measurements into visual simultaneous localization and mapping process. The necessity for this research is motivated by the requirement of creating the large scale outdoor Augmented Reality experience. The current state of the art VSLAM systems use multi-sensory fusion with monocular sensors to estimate the position of the robot and the locally accurate map scale. However, no previous work exists in the public domain, that uses just the GNSS measurement and the VSLAM for a geo-referenced localization of the robot. As such in this research, I have attempted to address this problem and propose my preliminary work and results.

The key contribution of this project includes - an android application that can stream RAW GNSS measurements over a TCP-based socket connection. The current GNSS-VSLAM system is inefficient but provides a possible approach for the automatic estimation of transformation between the GNSS world coordinate system and VSLAM world coordinate systems. The estimation of map scale with monocular visual SLAM without using the loop closure, and a new graph-based global bundle adjustment process for the optimization of GNSS measurements and camera poses.

From the test results, we can conclude that the current GNSS-VSLAM system is very brittle and susceptible to the noise in GNSS measurements which affects the initialization process of the system. This adversely affects the tracking of the GNSS-VSLAM over a large distance and may even lose track in the worst case. Some other constraints or challenges of the project are as follows

- The initialization of rotational transformation between the GNSS world and VSLAM world is only valid until the current tracking session is active. So if we reset the map or start a new one, it needs to be re-initialized.

- High computation and multi-threaded processing. While the current system works smoothly on a laptop (with an update rate of approx. 10 Hz), it would not be ideal for it to be deployed on a mobile platform for real-time AR experience.

For future work, I plan to use the re-localization based methods to estimates the cameras 6 DOF poses and delegate the heavy computation of maintenance of the global map and localization to a networked system. The pose estimation, the optimization of the global map, and multiple bundle adjustment are essential for the task of VSLAM. However, this task is computationally heavy and non-linear to be carried, on an edge mobile device. Some of the recent works such as Robust Hierarchical localization at large scale by Paul-Eduardo Sarlin et al. [47] and ORB-SLAM3 by Carlos Campos et al. [48] looks very promising for the field of visual SLAM. The HF-net proposed by Paul-Eduardo Sarlin et al. can compute the keypoints as well as global and local feature descriptors in a single shot, and localize the system with great accuracy (less than 0.5m with 80% recall rate) for real-time application (approx. 20 FPS). The ORB-SLAM3 system supports visual-inertial SLAM which is robust and 2 to 5 times more accurate than it's predecessors, along with multiple map SLAM, which allows to gradually merge two maps upon losing track.

8. APPENDICES

This chapter includes list of all the code snippets, applications used, and the user manual for testing them. All the project source codes and test data is made public either on github or shared via link with this report.

I. GNSS SERVER APPLICATION

The complete source code for the project can be found at: https://github.com/nfynt/GNSS_Compare/tree/socket_conn

To run the server install the apk as usual and from the applications menu select start/stop the GNSS server 8.1.

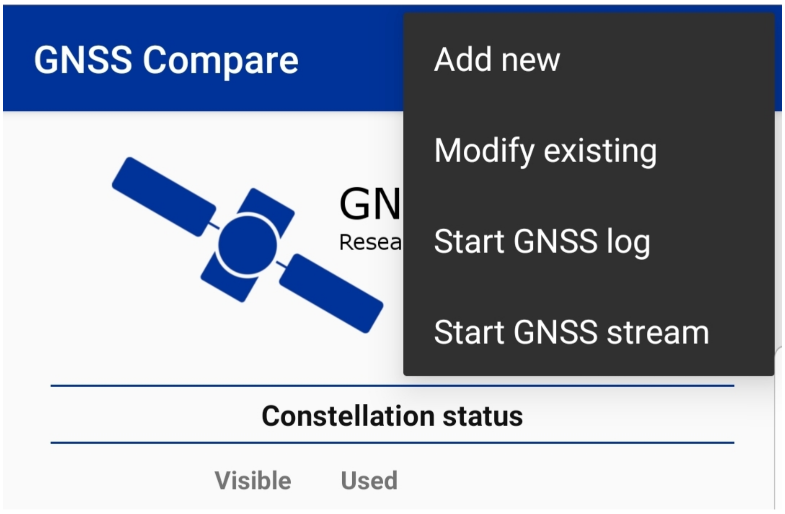


Figure 8.1: Start/stop GNSS server

By default the applications uses port 50000 for configuring TCP server and waits for inbound connection to stream the RAW data. To access this RAW data on a PC, connect the android device via a USB connection and use following steps:

- Obtain the ADB and install on the PC. This can be obtained from the Android SDK or in stand-alone packages.

- Ensure the USB drivers for your mobile are installed on the PC.
- Ensure that USB debugging is enabled on the mobile device.
- Ensure the ADB can detect your mobile by running the following command:
  adb devices
- Use ADB to forward mobile tcp port (50000 in this case) to PC tcp port (20175 in this case). Use command:
  adb forward tcp:20175 tcp:50000

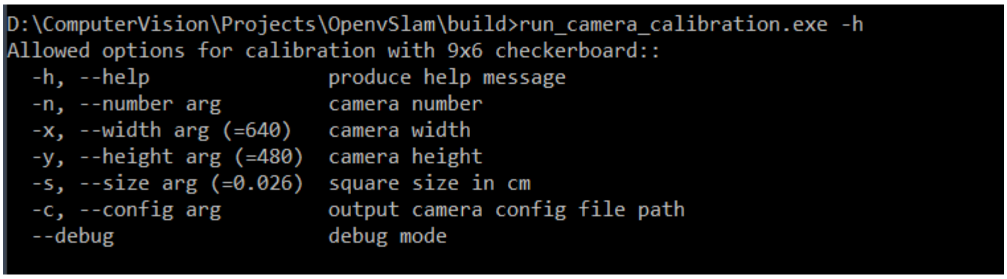
Test the RAW values from terminal on PC using Netcat or any other TCP/IP client.

II. VISUAL SLAM AND UTILITY APPLICATIONS

The complete source code for the project can be found at: https://github.com/nfynt/openvslam_gps_fusion/tree/develop

II.1. Camera calibration

Before running the main application we need to have the YAML configuration file containing the camera parameters. To calibrate any monocular camera, use the following utility application:

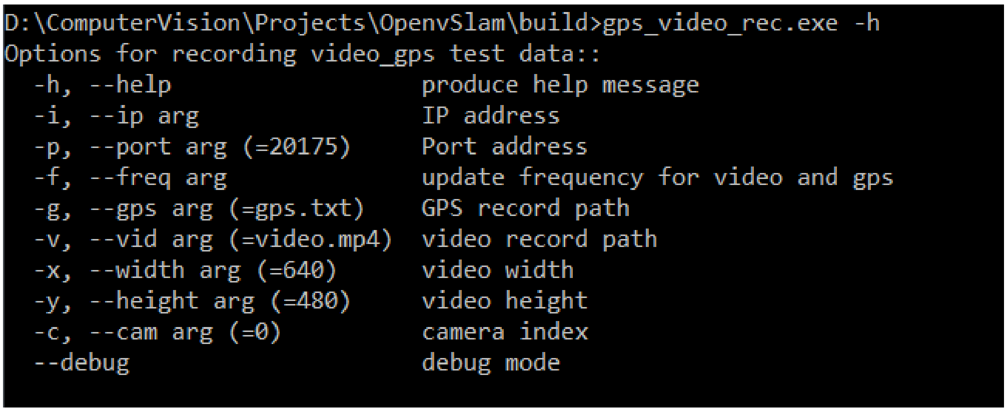


```
D:\ComputerVision\Projects\OpenvSlam\build>run_camera_calibration.exe -h
Allowed options for calibration with 9x6 checkerboard::
  -h, --help              produce help message
  -n, --number arg        camera number
  -x, --width arg (=640)  camera width
  -y, --height arg (=480) camera height
  -s, --size arg (=0.026) square size in cm
  -c, --config arg        output camera config file path
  --debug                 debug mode
```

Figure 8.2: camera calibration utility

II.2. Recording Video and GNSS measurement

In order to record the video and RAW GNSS measurement to be used for offline, you can use the following utility application:



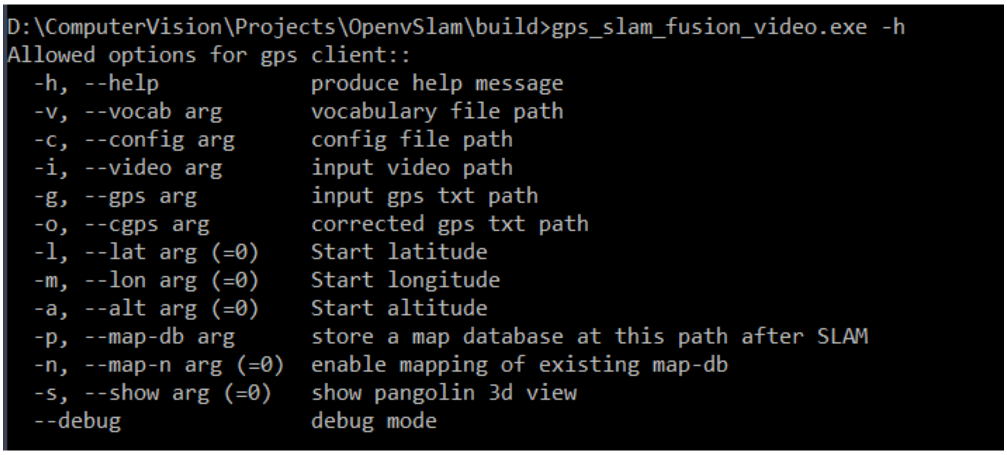
```
D:\ComputerVision\Projects\OpenvSlam\build>gps_video_rec.exe -h
Options for recording video_gps test data::
  -h, --help                  produce help message
  -i, --ip arg                IP address
  -p, --port arg (=20175)     Port address
  -f, --freq arg              update frequency for video and gps
  -g, --gps arg (=gps.txt)    GPS record path
  -v, --vid arg (=video.mp4)  video record path
  -x, --width arg (=640)      video width
  -y, --height arg (=480)     video height
  -c, --cam arg (=0)          camera index
  --debug                     debug mode
```

Figure 8.3: record GNSS and video

Alternatively, you can also use any other recorded video and RAW GNSS file, given the format of text file containing the measurements is in following order and csv format: timestamp, latitude, longitude, altitude, sensor speed, measurement accuracy, satellite count, multipath indicator, accumulated delta range uncertainty in meters, constellation type

II.3. Offline GNSS-VSLAM system

The offline GNSS-VSLAM system requires a video file from the calibrated camera and recorded text file with GNSS RAW measurement.

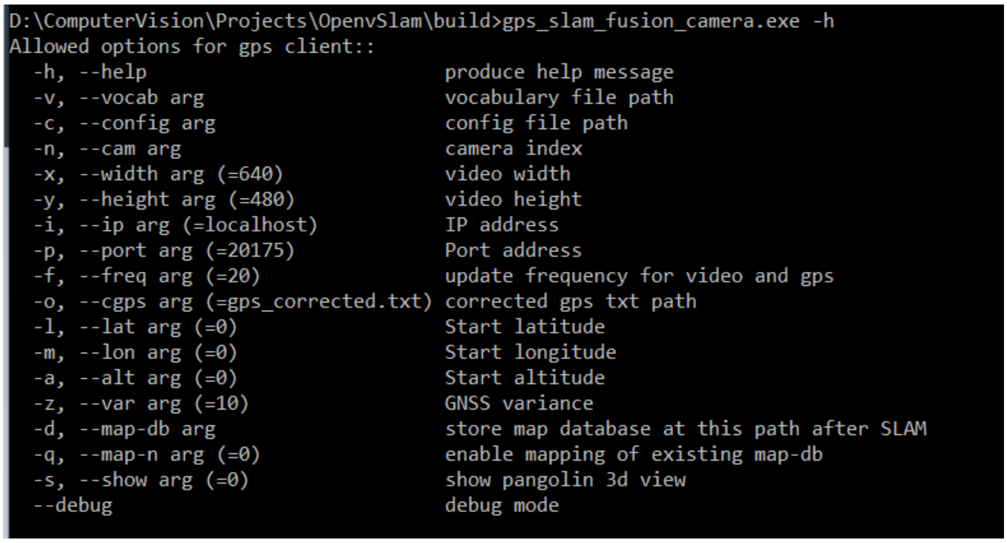


```
D:\ComputerVision\Projects\OpenvSlam\build>gps_slam_fusion_video.exe -h
Allowed options for gps client::
  -h, --help             produce help message
  -v, --vocab arg        vocabulary file path
  -c, --config arg       config file path
  -i, --video arg        input video path
  -g, --gps arg          input gps txt path
  -o, --cgps arg         corrected gps txt path
  -l, --lat arg (=0)     Start latitude
  -m, --lon arg (=0)     Start longitude
  -a, --alt arg (=0)     Start altitude
  -p, --map-db arg       store a map database at this path after SLAM
  -n, --map-n arg (=0)   enable mapping of existing map-db
  -s, --show arg (=0)    show pangolin 3d view
  --debug                debug mode
```

Figure 8.4: offline GNSS-VSLAM system

II.4. Online GNSS-VSLAM system

The online GNSS-VSLAM system runs in real-time using available camera and connected android device.



```
D:\ComputerVision\Projects\OpenvSlam\build>gps_slam_fusion_camera.exe -h
Allowed options for gps client::
  -h, --help                      produce help message
  -v, --vocab arg                 vocabulary file path
  -c, --config arg                config file path
  -n, --cam arg                   camera index
  -x, --width arg (=640)          video width
  -y, --height arg (=480)         video height
  -i, --ip arg (=localhost)       IP address
  -p, --port arg (=20175)         Port address
  -f, --freq arg (=20)            update frequency for video and gps
  -o, --cgps arg (=gps_corrected.txt) corrected gps txt path
  -l, --lat arg (=0)              Start latitude
  -m, --lon arg (=0)              Start longitude
  -a, --alt arg (=0)              Start altitude
  -z, --var arg (=10)             GNSS variance
  -d, --map-db arg                store map database at this path after SLAM
  -q, --map-n arg (=0)            enable mapping of existing map-db
  -s, --show arg (=0)             show pangolin 3d view
  --debug                         debug mode
```

Figure 8.5: Online GNSS-VSLAM system

III. CODE SNIPPETS

III.1. Rotational transformation

Listing 8.1: Rotational transformation matrix between GNSS and VSLAM coordinate system

```
//Estimate the camera position in SLAM world cs by inverse tranformation
Eigen::Vector3d w_pos = -camera_pose.block(0, 0, 3, 3).transpose() * camera_pose.block(0, 3, 3, 1);

 w_pos(1, 0) = 0; //ignore y position: assumed to be in same direction as UTM alt
Eigen::Vector3d gnss_pos = gps_parser::get_direction_vector(*start_geo , *curr_geo);
std::cout << "\nCamera pos " << w_pos.transpose()
                << "\nGPS pos " << gnss_pos.transpose() << std::endl;

//Normalize the world and gps direction vector
w_pos.normalize();
gnss_pos.normalize();
```

```
spdlog::info("Coordinate Basis\nWorld: {}\nGPS {}", w_pos.transpose(),
    ↪  gnss_pos.transpose());

//Estimate the rotation matrix to transform from gnss_pos to w_pos
Eigen::Vector3d rotation_axis = gnss_pos.cross(w_pos);
rotation_axis.normalize();
double angle = std::acos(gnss_pos.dot(w_pos));

// degenrate case when vector a and b point's in opposite direction
if (cos(angle) == -1) {
    spdlog::info("degenerate transformation case, will retry");
    continue;
}

Eigen::Quaternion<double> t_quat(Eigen::AngleAxisd(angle,
    ↪ rotation_axis));
t_quat.normalize();
R_wgnss = t_quat.toRotationMatrix();

std::cout << "Rotation matrix (gnss->slam_w)\n" << R_wgnss
         << "\n rotation angle: " << angle << "\n";
```

III.2. Estimation of map scale

Listing 8.2: Estimation of scale factor for global map

```
Vec3_t mean_gps;
Vec3_t mean_cam;
mean_of_eigen_vec(gps_pos, mean_gps);
mean_of_eigen_vec(cam_pos, mean_cam);

double sum_gps_diff = 0.0;
double sum_cam_diff = 0.0;
for (size_t i = 0; i < gps_pos.size(); ++i) {
    sum_gps_diff += (gps_pos[i] - mean_gps).squaredNorm();
```

```
        sum_cam_diff += (cam_pos[i] - mean_cam).squaredNorm();
    }
    const double scale = std::sqrt(sum_gps_diff / sum_cam_diff);
```

REFERENCES

[1] Ivan E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), page 757–764, New York, NY, USA, 1968. Association for Computing Machinery.

[2] Julie Ducasse. Augmented reality for outdoor environmental education. In *Augmented Reality in Education*, pages 329–352. Springer, 2020.

[3] R. Cervenak and P. Masek. Arkit as indoor positioning system. In *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–5, 2019.

[4] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[5] Shinya Sumikura, Mikiya Shibuya, and Ken Sakurada. OpenVSLAM: A Versatile Visual SLAM Framework. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, pages 2292–2295, New York, NY, USA, 2019. ACM.

[6] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[7] R. Mur-Artal and J. D. Tardós. Fast relocalisation and loop closing in keyframe-based slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 846–853, 2014.

[8] Hauke Strasdat, J Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2(3):7, 2010.

[9] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[10] David Nistér. Preemptive ransac for live structure and motion estimation. *Machine Vision and Applications*, 16(5):321–329, 2005.

[11] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[12] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[13] Herbert Landau, Xiaoming Chen, Sören Klose, Rodrigo Leandro, and Ulrich Vollath. Trimble's rtk and dgps solutions in comparison with precise point positioning. In *Observing our Changing Earth*, pages 709–718. Springer, 2009.

[14] European Space Agency. *Using raw GNSS measurement on Android device. White paper*, 2019 (accessed July 20, 2020).

[15] Pierre Wellner, Wendy Mackay, and Rich Gold. Back to the real world. *Communications of the ACM*, 36(7):24–26, 1993.

[16] DWF Van Krevelen and Ronald Poelman. A survey of augmented reality technologies, applications and limitations. *International journal of virtual reality*, 9(2):1–20, 2010.

[17] Steven Feiner, Blair MacIntyre, Tobias Höllerer, and Anthony Webster. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. *Personal Technologies*, 1(4):208–217, 1997.

[18] Amir H Behzadan and Vineet R Kamat. Georeferenced registration of construction graphics in mobile outdoor augmented reality. *Journal of computing in civil engineering*, 21(4):247–258, 2007.

[19] Amir H Behzadan, Brian W Timm, and Vineet R Kamat. General-purpose modular hardware and software framework for mobile outdoor augmented reality applications in engineering. *Advanced engineering informatics*, 22(1):90–105, 2008.

[20] Gethin W Roberts, Andrew Evans, Alan Dodson, Bryan Denby, Simon Cooper, Robin Hollands, et al. The use of augmented reality, gps and ins for subsurface data visualization. In *FIG XXII International Congress*, pages 1–12, 2002.

[21] Gethin Roberts, Xiaolin Meng, Ahmad Taha, and Jean-Philippe Montillet. The location and positioning of buried pipes and cables in built up areas. In *Proceedings of XXIII Fig Congress: Shaping the Change, October*, pages 1–9, 2006.

[22] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[23] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *null*, page 1403. IEEE, 2003.

[24] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[25] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[26] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[27] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572. IEEE, 2007.

[28] Joakim Rydell and Erika Emilsson. Chameleon: Visual-inertial indoor navigation. In *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pages 541–546. IEEE, 2012.

[29] Gabriel Nützi, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Fusion of imu and vision for absolute scale estimation in monocular slam. *Journal of intelligent & robotic systems*, 61(1-4):287–299, 2011.

[30] Laurent Kneip, Stephan Weiss, and Roland Siegwart. Deterministic initialization of metric state estimation filters for loosely-coupled monocular vision-inertial systems. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2235–2241. IEEE, 2011.

[31] Todd E Humphreys and Glenn Lightsey. Fusion of carrier-phase differential gps, bundle-adjustment-based visual slam, and inertial navigation for precisely and globally-registered augmented reality. page 159, 2013.

[32] Daniel Wilbers, Christian Merfels, and Cyrill Stachniss. A comparison of particle filter and graph-based optimization for localization with landmarks in automated vehicles. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 220–225. IEEE, 2019.

[33] Tobias Feigl, Andreas Porada, Steve Steiner, Christoffer Löffler, Christopher Mutschler, and Michael Philippsen. Localization limitations of arcore, arkit, and hololens in dynamic large-scale industry environments. In *VISIGRAPP (1: GRAPP)*, pages 307–318, 2020.

[34] Dániel Kiss-Illés, Cristina Barrado, and Esther Salamí. Gps-slam: an augmentation of the orb-slam algorithm. *Sensors*, 19(22):4973, 2019.

[35] Clark N Taylor. An analysis of observability-constrained kalman filtering for vision-aided navigation. In *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pages 1240–1246. IEEE, 2012.

[36] Xiao Chen, Weidong Hu, Lefeng Zhang, Zhiguang Shi, and Maisi Li. Integration of low-cost gnss and monocular cameras for simultaneous localization and mapping. *Sensors*, 18(77):2193, Jul 2018.

[37] Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A general optimization-based framework for global pose estimation with multiple sensors. *arXiv:1901.03642 [cs]*, Jan 2019. arXiv: 1901.03642.

[38] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[39] *GNSS Compare application - Galileo Smartphone application at European Space Agency Challenge*, 2019 (accessed July 20, 2020).

[40] B LOUIS Decker. World geodetic system 1984. Technical report, Defense Mapping Agency Aerospace Center St Louis Afs Mo, 1986.

[41] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual slam: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[42] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[43] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[44] Andrey Soloviev and Donald Venable. Integration of gps and vision measurements for navigation in gps challenged environments. In *IEEE/ION Position, Location and Navigation Symposium*, pages 826–833. IEEE, 2010.

[45] Renfro Brent A., Terry Audric, and Boeker Nicholas. *An Analysis of Global Positioning System (GPS) Standard Positioning System (SPS) Performance for 2016*. The University of Texas at Austin, "2017 (accessed July 20, 2020)".

[46] Rudolph Van Der Merwe, Eric Wan, and Simon Julier. Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 5120, 2004.

[47] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019.

[48] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam. *arXiv preprint arXiv:2007.11898*, 2020.