Factor graphs for inventory label scanning in warehouse

Darshan Kalyanasundaram

Technische Universiteit Delft

TU Delft

Factor graphs for inventory label scanning in warehouse

by

Darshan Kalyanasundaram

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday November 28, 2023 at 12:45 PM.

Student number:	5351170	
Project duration:	March 1, 2022 – November 28, 2023	
Thesis committee:	Prof. Dr. G.C.H.E De Croon,	TU Delft, supervisor(AE)
	Prof. Dr. J. Alonso Mora,	TU Delft,supervisor(3mE-CoR)
	Ir. Michaël Ozo,	Xosight B.V., supervisor
	Prof. Dr. H. Caesar	TU Delft, external(3mE-CoR)

TU Delft

XOSIGHT

Preface

The end of my master's thesis concludes my three-year-long journey at TU Delft. I am grateful to TU Delft for this opportunity. I have learnt a lot in the past three years which would not be possible if not for TU Delft. I want to thank my parents and brother for believing and supporting me. I am grateful to Dr.Guido de Croon and Ir.Michaël Ozo for their guidance throughout the project. The motivation and support I obtained from them have inspired me to realize the importance of scientific research. I would also like to thank Dr.Javier Alonso Mora for supporting me in this work. I would like to thank my colleagues Navanith and Avinash at Xosight for helping me with the dataset. I am grateful to my friends Godwin, Stephy and Gopalan for supporting me during my thesis. I would also like to thank Chinmay, Denzil and Akshath for the fun times during the 1st year of my studies.

Darshan Kalyanasundaram
Delft, November 2023

Contents

1 Scientific Paper **1**

1

Scientific Paper

Factor graphs for inventory label scanning in warehouse

Darshan Kalyanasundaram (5351170)

***Abstract*—A key important part of a warehouse operation is to keep track of the products in the warehouse. Traditionally, handheld scanners are used to scan the products to perform a stock count. The advancements in robotics have paved the way for new technologies that can improve the scanning process. This work shows how prior knowledge and warehouse structure can be used to perform scanning operations. The method uses a localized camera in the warehouse whose estimates drift over time and the knowledge about the environment to estimate the correct location of the product using factor graphs. The proposed method shows by exploiting the structure of the warehouse the drift in the VIO(Visual Inertial Odometry) can be reduced and the position estimation of the location labels can be improved.**

***Index Terms*—factor graphs, warehouse, inventory scanning**

I. Introduction

Warehouses play a key part in storing and transporting materials from different locations to the customers. It is important to keep track of all the materials that are currently present in the warehouse. This is achieved by inventory counting, a process to take account of all the stock currently present in the warehouse. The most common method of inventory scanning is using a handheld barcode scanner. While this method is widely used, it leads to inventory counting errors. An error can lead to a misplaced or a lost package which might be extremely hard to find thus increasing the time of delivery of the product. XOsight, a startup in Delft, aims to tackle the errors caused by inventory counting by using computer vision and machine learning methods.

The stocks in the warehouse are identified by a unique barcode which we refer to as a location label (Fig. 1). During inventory counting, the location label is scanned using the handheld scanner to perform stock counting. An error occurs when the label is scanned incorrectly due to a lapse in the human operator's attention. Handheld scanning for large warehouses becomes time-consuming and requires a large workforce. The handheld scanner has a limited range which makes it difficult to deal with large packages or difficult-to-reach areas.

In this work, we try to address the limitations caused by a handheld scanner using a camera-based scanning approach which can localize the location labels in the warehouse. XOsight already uses a camera-based system to localize the location labels in the warehouse. The localization of the labels is achieved using a VIO (Visual Inertial Odometry) [1] algorithm. When the drift in the VIO is significantly high, the localized labels have incorrect positions. These incorrect position estimates are undesirable as they can lead to misplaced packages. In this work, we address this problem by using factor graphs [2]. We use Gaussian priors and known constraints by exploiting the warehouse structure to estimate correct positions of the location labels. The developed method is validated using a real-world dataset.

Fig. 1. A typical location label in the warehouse.

II. Related Work

In recent times, the use of robots in stock counting has increased due to advancements in robotics. [5] proposes a LiDAR-based system to map and localize inside the warehouse at high speeds using an aerial drone. The method uses RFID tags and April Tags to perform stock counting. The stock position is estimated based on the RFID, April tag measurements and the drone position. A prior map of the warehouse is obtained from manual flight which is then used to localize the drone in the warehouse. The high speeds of the drones are ensured by the A* algorithm which acts as a local planner and reactive obstacle avoidance method to avoid dynamic obstacles. In addition to this, a visibility constraint is added such the path generated is always in the field of view of the LiDAR.

A CNN-based barcode detection was proposed in [7] to perform inventory counting. This work builds upon [6] which proposes a heterogeneous approach to localize a UAV (Unmanned Aerial Vehicle) in a warehouse-like environment using a UGV(Unmanned Ground Vehicle). In [7], the barcodes are detected using CNN which is used to optimize the trajectory of the drone. The optimized trajectory is then used to scan the barcodes using a laser scanner. Both the approaches [6] and [7], rely on the UGV to get the global position of the UAV.

A sensor fusion technique based on multiple sensors using cameras, a 2D laser scanner, a 1D range sensor and an IMU was proposed in [8] to perform autonomous navigation using

UAVs in the warehouse. The proposed method also uses April tags whose positions are stored in an XML map. The measurements from the sensors are used to estimate relative and absolute 6D pose and altitude. The April tags are detected to estimate absolute 6D pose since their prior position is known which can be used to counter the drift in the system. The estimated quantities are then fused with an extended Kalman filter (EKF) to get a better estimate of the position of the UAV. The proposed method in [8] explains only the navigation part and leaves out the inventory counting process.

[10] presented SPM-SLAM algorithm that can be used to map large environments using markers. The system uses two initialization techniques to initialize the system based on marker observations. If the pose of the marker is detected without ambiguity(where there are two possible rotations for a single frame) then the system uses one frame initialization method. If there is a marker detected with ambiguity then it uses two two-frame initialization methods to initialize the system. The system uses keyframes to create a map. The markers are added to the keyframe only if they have the pose detected without ambiguity. The growth of the map is limited by removing keyframes that are redundant. When a new keyframe is added to the map the system performs local optimization where the new keyframe and the neighbouring keyframes are optimized. When no markers are observed the system enters relocalization mode until a new marker is observed. The system performs loop closure when a previously observed marker is observed again. All the keyframes are stored in a graph structure where the edges represent the constraints between the keyframes which here is the relative transformation between the frames. Before adding a loop closure, the drift accumulated is corrected by minimizing the error across all the edges. This is done to avoid incorrect estimation of the pose of a new marker which may be observed in the loop closing frame. When all the markers are observed, the graph containing all the keyframes and the poses along with the camera intrinsic parameters are subjected to global optimization.

[11] proposed a new marker-based SLAM approach which uses odometry information from a VIO algorithm to create a map with the markers. The odometry data is combined with the tag measurements from the camera which detects the corners. The combined data is used to construct a factor graph. TagSLAM considers all the frames with tags as a keyframe. To exploit the sparsity in the keyframes, iSAM2 is used to estimate correct tag poses. TagSLAM can incorporate priors in two forms. If the exact pose of the tag is known then this can be added as an absolute prior. If the distance between two tags is known this can be added as a relative pose prior. The incoming measurements are stored in a full graph which contains all the past measurements to the latest measurement. When the measurements in the full graph can be initialized then they are stored in the optimized graph which forms a well-constrained problem. TagSLAM uses subgraph discovery where the full graph is traversed till all new variables become determinable. When the discovery of all the new variables is completed using subgraph discovery, these subgraphs are subjected to optimization without using ISAM2. Based on the error threshold, the subgraph is then transferred to the optimized graph which is optimized using ISAM2. TagSLAM is able to reduce the drift occurring in the VIO with the help of tags. The main reason for the reduction in the drift is the loop closure with the help of the tags.

[12] is an extension of SPM-SLAM where it uses keypoints in addition to tags. Keypoints are specific points in an image which can be used to represent an image. These keypoints are extracted from the image using feature descriptors. UcoSLAM can also work only with keypoints or only with tags when either keypoints or tags are available. This provides great flexibility in its operational environment. When only tags are available, UcoSLAM behaves as SPM-SLAM. UcoSLAM uses a connection graph where the vertices are keyframes and the edges are weights. These weights are given based on how many keypoints and tag corners are present in a connected frame. If there are ten keypoints that are seen in two keyframes then a weight of ten is assigned to the edge connecting those keyframes. Similarly, if a tag is visible in two keyframes then a weight of four is assigned to the edge connecting the two keyframes. The weight of four represents the four corners of the tags. The bag-of-words approach is used for keyframe recognition. UcoSLAM runs a map manager on a parallel thread which performs loop closure based on keypoints. The map manager also performs keyframe culling to keep the system from using high computational resources. The tag-based loop closure and culling are the same as SPM-SLAM. The global optimization is performed when no more keypoints or tags are available to process.

The method proposed in [5] requires special RFID tags and April tags to estimate the product locations which makes it difficult to adapt to warehouses which do not use RFID tags. Also, a prior map of all the April tags is required to perform proper localization. While the method [7] uses the barcodes in the warehouse to optimize the trajectory the results presented in the paper are tested for a very small trajectory. The method also requires additional ground robot for proper localization in the warehouse which makes the system complex. The method proposed in [8] showcased the sensor fusion approach to perform inventory counting in the warehouse. However, a prior map of April tags is required to perform proper localisation. The use of multiple sensors also makes the system heavy and expensive. The methods [10], [11] and [12] can be used to perform inventory counting and reduce the drift in the system if some prior knowledge is known about them. The use of marker-based SLAM in the warehouse might sound like a good idea. But the modern warehouses already have a standardized system which uses location labels to perform inventory counting. The marker-based SLAM would complicate the existing infrastructure that is already in place. This work aims to perform inventory scanning using a camera and estimate the position of location labels in the warehouse without any additional sensors and markers as discussed in the above methods. This method shows

how the position of the location label can estimated from a drifting pose estimates over time and also how the environment structure can be used in estimating the position of the location label. In the following sections, the methodology, the dataset and results will be explained in detail.

III. Factor Graphs

A factor graph (Fig. 2) is a bipartite graph which can be mathematically expressed as $\mathbb{F} = (\mathbb{U}, \mathbb{V}, \mathbb{E})$. There are two types of nodes in a factor graph which are known as variables $x_i \in \mathbb{V}$ and factors $\phi_i \in \mathbb{U}$. A factor node ϕ_i is connected to a variable node x_j through edge $e_{ij} \in \mathbb{E}$. The variable node x_i represents the quantity in the problem that needs to be modelled. In our case, this quantity is a pose in 3D space. The factor node ϕ_i contains the constraints connecting two variable nodes x_i and x_j. In our case, this factor would be a relative pose constraint between two poses. A factor graph

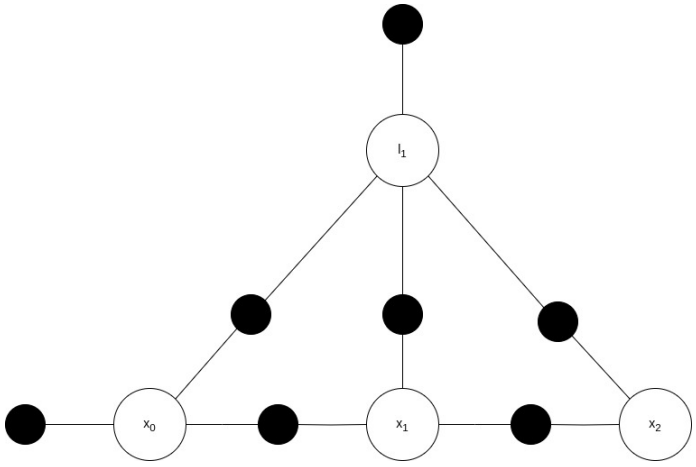

Fig. 2. An example of a factor graph. Here the white circles are the variable nodes and the black circles are the factor nodes. Note that the variable nodes x_0 and l_1 have additional factor nodes which are not connected to any other variable node. These are called unary factors. An example of an unary factor is a prior factor node which encodes the prior knowledge of the variable node connected to itself.

provides a graphical representation of the factorization of a function. The factorization of a function X is given as,

$$\phi(X) = \prod_i \phi_i(X_i) \tag{1}$$

To understand the above equation let's look at the equation below,

$$f(a, b) = g(a) * h(b) \tag{2}$$

In the equation 2, the right-hand side contains two individual functions $g(a)$ and $h(b)$ whose product gives a new function $f(a, b)$. The functions $g(a)$ and $h(b)$ are the factors of the global function $f(a, b)$. The factors are used to understand the local properties of the global function. Similarly, in equation 1, the right-hand side contains the factors that model the local properties whose product gives the global function $\phi(X)$. The global function $\phi(X)$ represents the problem that needs to be solved.

The ability to model the local properties of a complex problem makes factor graphs attractive to use in the robotics domain. It also allows to use domain domain-specific knowledge which can be added as factors to the graph structure which can allow to produce better and faster results. The concept of factors in the graph structure allows to understand and explain the problem clearly as the factors contain the necessary constraints to solve the problem [3]. In this work to construct and work with factor graphs, we use GTSAM (Georgia Tech Smoothing and Mapping) [4], an open-source library which allows to construct factor graphs and perform optimization on it. The library also provides several factors that can be used directly in SLAM, Structure-from-Motion, collision avoidance and trajectory optimization problems. The library also allows the users to create custom factors for specific applications.

IV. Methodology

A. Overview

In this work, the position of the location labels is estimated using camera measurements. The system uses a VIO algorithm which localizes the camera in the world frame. The position of the location labels is estimated using the camera pose. The pose estimates of the VIO algorithm experience drift which leads to incorrect position estimates which causes the estimated position of the location label to be incorrect. The incorrect position of the location labels and the drifted pose of the camera is added to a factor graph which contains known constraints and Gaussian priors. The factor graph is subjected to optimization to obtain a corrected pose of the camera and position of the location label (Fig. 3). The following sections will explain the system in detail.

B. System setup

The system consists of a stereo camera, an IMU and a monocular camera which is mounted on a forklift. The stereo camera and IMU localize the camera in the world frame using VIO. The monocular camera's pose is estimated using the relative position between the stereo camera and the monocular camera. The location labels are detected using a monocular camera. The monocular camera's pose is then used to estimate the location label in the world frame by using the pixel coordinates. The pose estimated by the VIO drifts over time leading to inaccurate pose estimates. The inaccurate pose estimates also cause the position of the location label to be inaccurate. To correct the drift and find the correct position of the location labels, the incorrect pose of the camera and the incorrect position of the location label are added to the factor graph. The corrected pose of the camera and the position of the location label are estimated using known constraints and Gaussian priors which are added to the factor graph. A detailed explanation of the system is discussed in the following sections.

C. Visual Inertial Odometry

Visual Inertial Odometry (VIO) is the process of estimating the camera pose in a global frame by fusing the measurements from the camera and an IMU. One can also obtain velocity measurements of the camera from the algorithm. Based on the type of visual sensor used in the system there are different VIO algorithms. [14], [15] and [16] are few examples of VIO algorithm.

The VIO algorithm estimates the pose of the camera by tracking features in the consecutive image frames which are known as keypoints. These keypoints are triangulated to obtain the camera pose. The features in the image are obtained by feature detectors like FAST, SIFT or BRIEF. The detected keypoints along with the IMU measurements are fused together using a filtering-based approach or optimization approach to obtain the pose and velocity of the camera. In the filtering-based approach [15], the keypoints and IMU measurements are fused together using an extended Kalman filter to estimate the pose of the camera. In the optimization-based approach [16], the keypoints and IMU measurements are added to a graph which is optimized to obtain the pose of the camera. Both filtering and optimization-based approach has their own advantages and disadvantages, please refer [13] to know more. This work uses an optimization-based approach, VINS-Fusion an extension of VINS-Mono [16] to localize the camera. Although [13] shows how the optimization-based approach performs better localization than the filtering-based approach, our work does not rely on the type of VIO approach used since it only uses the final pose estimates from the algorithm.

D. Location label detection

The location labels are detected using the monocular camera. The image frames from the monocular camera are fed to a CNN-based neural network which detects the location labels and gives an OCR output of the numbers on the label along with a bounding box around the label(Fig. 4). When the location label is not viewed properly, the OCR output produced by CNN is incorrect. A preprocessing step is applied to select only the correct OCR output. This is done by ensuring the OCR has the desired length of characters in the output. The correct OCR output is then stored for estimating the 3D position of the location label.

E. Location label 3D position estimation

The 3D position of the location label is estimated using the pose estimates from the VIO. The bounding box provides the position of the location label in the image. The bounding box is used to extract the center pixel coordinates (u_c, v_c) of the location label. The camera intrinsic matrix K is used to convert the pixel coordinates (u_c, v_c) in the image to the camera frame T_c. This gives the relative position of the location label's center p_{rel} with respect to the camera frame. The camera's global pose ${}^{c}T_{w}$ information is available from the VIO pose estimates which are used to covert the location label's position with respect to global frame p_{global}.

F. VIO Drift and Incorrect location label position estimation

As the camera moves over a period of time, the VIO algorithm accumulates errors in the system leading to inaccurate pose estimates which is known as drift. Since the VIO pose estimates are used to estimate the position of the location labels, the drift in VIO pose estimates is propagated into the position estimation of the location label leading to an inaccurate position estimate of the location label. The incorrect pose estimates and incorrect 3D position of location labels are used to model the factor graph along with Gaussian priors which represent the knowledge about the environment.

G. Factor graph structure

The drift in the VIO cannot be completely eliminated unless a ground truth is available for the trajectory. However, the drift can be reduced by adding prior knowledge and adding information about the environment. Factor graphs are used to create a graphical model of the trajectory and the detected location labels which is optimized to reduce the drift in the trajectory thereby estimating the correct position of the location label.

When the VIO is initialized, the global frame of reference T_{origin} is defined. The factor graph is also initialized with the same global frame of reference by adding a Gaussian prior Σ. When the camera moves, the VIO provides the pose estimates of the camera whose references are with respect to T_{origin}. These pose estimates are added to the factor graph $\mathbb{F}$ as variable nodes $x_i \in \mathbb{X}$. In factor graphs, the variable nodes are treated as unknown quantities so the added pose estimates will be considered as an initial estimate during the optimization process. The pose variable nodes $\mathbb{X}$ are connected to factor nodes ϕ. The factor node contains the relative pose between two variable nodes (add image). The relative pose is estimated based on the current camera pose estimate and the previous pose estimate from the VIO. The factor node also contains information about the uncertainty in the relative pose as a Gaussian noise. Prior knowledge about the camera's height is added as a Gaussian prior to all the pose variable nodes to avoid any unnecessary drift in the camera's height.

As the camera moves through the warehouse, the location labels are detected. The detected location labels are added to the factor graph as new variable nodes $\mathbb{L}$. Based on the OCR output, for every new location label, a new variable node l_j is created. The variable node $l_j \in \mathbb{L}$ represents the unknown 3D position of the location label which is to be estimated during the optimization. The estimated 3D position from the VIO pose is used as an initial estimate during optimization. The location label variable node l_j is connected to a projection factor node ϕ_p. The projection factor node ϕ_p contains the pixel measurement of the location label with a Gaussian noise defining the uncertainty in the pixel measurement. The projection factor node ϕ_p is connected to a pose variable node x_i. The projection factor minimizes the reprojection error based on the camera pose and the pixel measurement. Since the height of the location labels and the camera are known, this information is added as a Gaussian prior for all the location

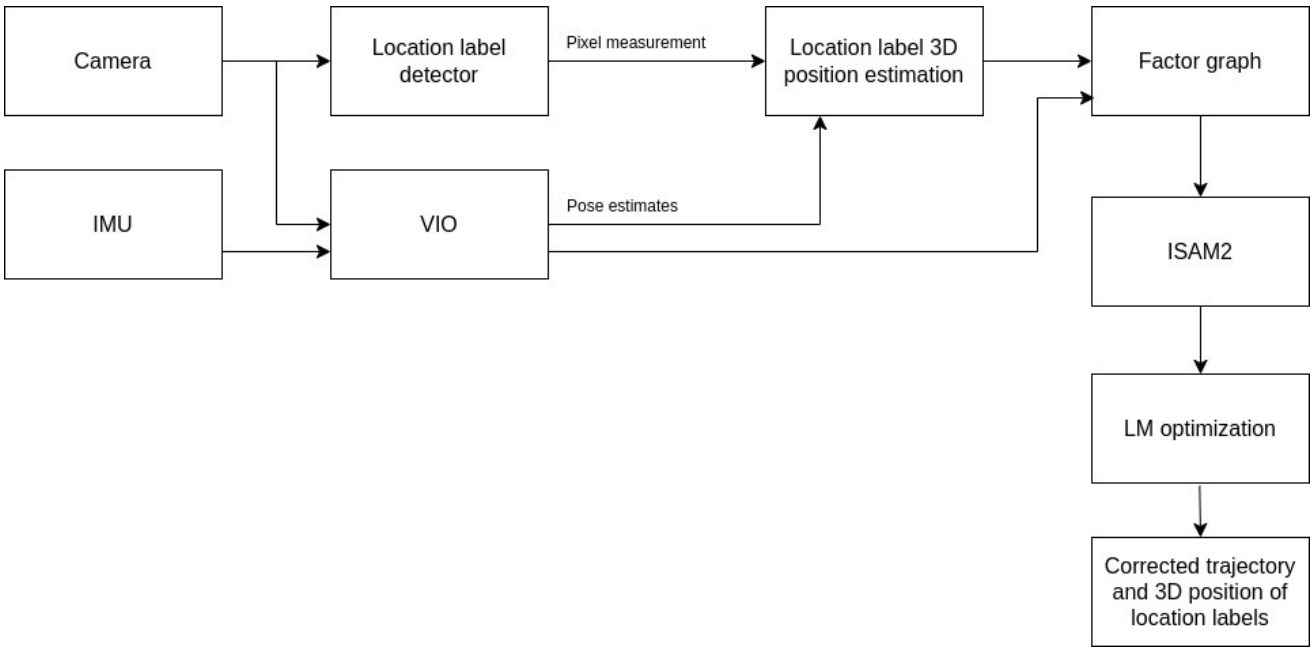

Fig. 3. System overview. The images from the camera are used to estimate the pose of the camera by the VIO algorithm. The image is also used to detect the location label whose 3D position is estimated. The pose estimates and the estimated 3D position of the location label are added to the factor graph. The factor graph is subjected to optimization by ISAM2 and the Levenberg-Marquardt algorithm to obtain the corrected trajectory and 3D position of the location labels.

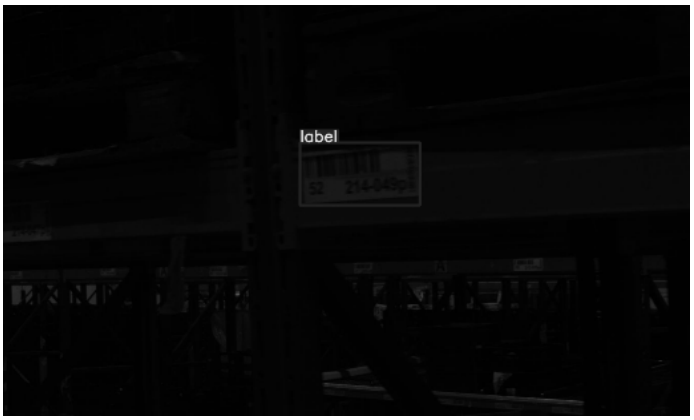

Fig. 4. A detected location label from the dataset

label variables and pose variables. Fig. 5 shows the factor graph structure used in optimization.

H. Line factor

The location labels in the warehouse are located on the lower rack of an aisle. The racks are aligned such that they lie in a straight line. This straight-line information can be modelled in the factor graph as a line factor. The line equation is calculated at the early stage of the scanning process. During the beginning of the scanning process, the drift in the VIO is assumed to be negligible. When there are detections of two different location labels, their 3D position becomes available. A line equation is constructed using their x and y coordinates using a point-slope formula. The height of the location label is not considered since it is already known as a prior. The constructed line equation models the line in which all the location labels should lie. In GTSAM, the Jacobians need to be calculated based on the error function used in the custom factor. The Jacobians provide the information about the gradients of the error function to the optimizer which is necessary to converge to an optimal solution.

The error function in the line factor reduces the perpendicular distance between the line equation and the location label coordinates in the 2D plane. The perpendicular distance between a point in a 2D plane and a line is given by,

$$e = \frac{|Ax + By + C|}{\sqrt{A^2 + B^2}} \tag{3}$$

where x and y are coordinates of the location label in 2D. A and B are coefficients of x and y. C is a constant term. The Jacobian for the error function with respect to x and y coordinates is given by,

$$J = \begin{bmatrix} \frac{\partial e}{\partial x} & \frac{\partial e}{\partial y} & \frac{\partial e}{\partial z} \end{bmatrix} \tag{4}$$

$$\frac{\partial e}{\partial x} = \frac{\text{sgn}(Ax + By + C) \cdot A}{\sqrt{A^2 + B^2}} \tag{5}$$

$$\frac{\partial e}{\partial y} = \frac{\text{sgn}(Ax + By + C) \cdot B}{\sqrt{A^2 + B^2}} \tag{6}$$

The Jacobian contains z coordinate which is not present in the error function (eq. 3). This is required by GTSAM implementation since the location labels are in 3D position but only the x and y coordinates are optimized which makes

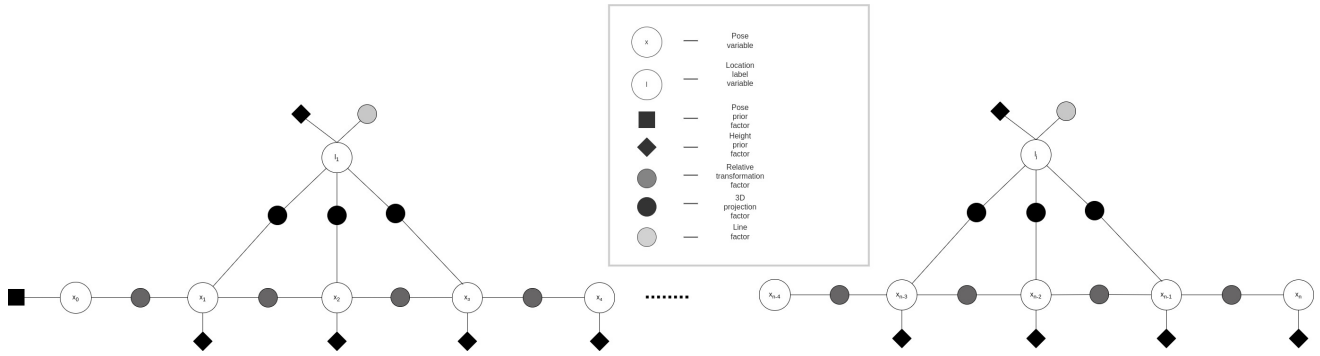

Fig. 5. Factor graph structure used in optimization. Here it can be seen that a location label variable node is connected to multiple pose views using the projection factor. Every pose and landmark has a height prior factor. The location label variable node also has an additional unary factor which is the line factor. The pose variable nodes are connected to the adjacent pose variable nodes using a relative transformation factor.

the z gradient of the Jacobian vector to be zero. The final Jacobian vector is given as,

$$J = \left[\frac{\text{sgn}(Ax + By + C) \cdot A}{\sqrt{A^2 + B^2}} \frac{\text{sgn}(Ax + By + C) \cdot B}{\sqrt{A^2 + B^2}} 0 \right] \tag{7}$$

where $sgn(Ax + By + C)$ is given as,

$$\text{sgn}(Ax + By + C) = \begin{cases} -1, & \text{if } Ax + By + C < 0 \\ 0, & \text{if } Ax + By + C = 0 \\ 1, & \text{if } Ax + By + C > 0 \end{cases} \tag{8}$$

I. Optimization

The unknown values in the variables nodes of the factor graph are estimated by performing MAP (Maximum A Posteriori) inference. The goal of MAP is to maximize the product of all the factors in the graph (1). [19] shows performing MAP inference on a factor graph is equivalent to solving the nonlinear least squares. The optimization problem becomes,

$$\phi(X) = \underset{X}{\text{argmin}} \sum \|h_i(x_i) - z_i\|^2_{\Sigma_i} \tag{9}$$

Here $\phi(X)$ can be treated as X^{MAP} since both refer to maximizing the product of the factors. In 9, $h_i(X_i)$ is the measurement function that needs to be estimated, z_i is the measurement from the sensor and Σ_i is the covariance which contains the noise model of the sensor measurement.

The factor graph is optimized in two steps using ISAM2 (Incremental Smoothing and Mapping 2) [18] and Levenberg-Marquardt optimization. ISAM2 optimizes the factor graph in an incremental step. ISAM2 converts the factor graph into a Bayes net and solves the Bayes net using an incremental matrix factorization. ISAM2 uses a technique called fluid relinearization which linearizes the Bayes net only when needed rather than linearizing at every time step. ISAM2 reduces the computational cost by updating only the affected variables in the Bayes net due to relinearization and leaves out the unaffected part of the Bayes net. This can be understood with the following example. When the camera moves around the warehouse, the location label present in one aisle does not affect the estimates obtained for the previous location label in a different aisle. So solving only for the current location label would reduce the computational cost greatly. This is what ISAM2 does. Similarly, when a loop closure happens, only the affected leaves are optimized which reduces the computational cost. When the detection of all location labels is completed, a final global optimization is performed using Levenberg-Marquardt optimization to remove any errors present from the ISAM2 estimates.

During experiments, it was found that using purely a Gaussian model in the optimization does not converge properly to produce better results (see Section VI). Robust estimators are used to deal with the problem caused by pure Gaussian noise models. GTSAM provides error functions like Cauchy [20], Dynamic Covariance Scaling(DCS) [21] and Huber [22] which can be used in the optimization to produce better estimates of the variables in the factor graph. Unlike least squares which become unstable when outliers are encountered, the robust error functions are capable of dealing with outliers which makes them suitable for real-world datasets. When the Cauchy or Huber error function is used in optimization, the problem becomes,

$$\min_x \sum_i \rho\left(\|h_i(x_i) - z_i\|_{\Sigma_i}\right) \tag{10}$$

DCS [21] error function is primarily used to deal with incorrect loop closures in the factor graph. The error function uses a scalable variable which is also minimized during the optimization. The scalable variable is a scalar value which is multiplied by the covariance matrix. This allows the covariance matrix to be changed during the optimization allowing it to deal with the outlier more efficiently.

V. DATASET

The proposed method is validated using a real-world dataset. The real-world dataset is recorded in a warehouse using the camera setup mounted on a forklift. The forklift moves back and forth in an aisle and records the images. These images are used to detect the location labels and estimate their 3D position. Fig. 6 shows a typical scanning trajectory used to estimate the position of the location labels.

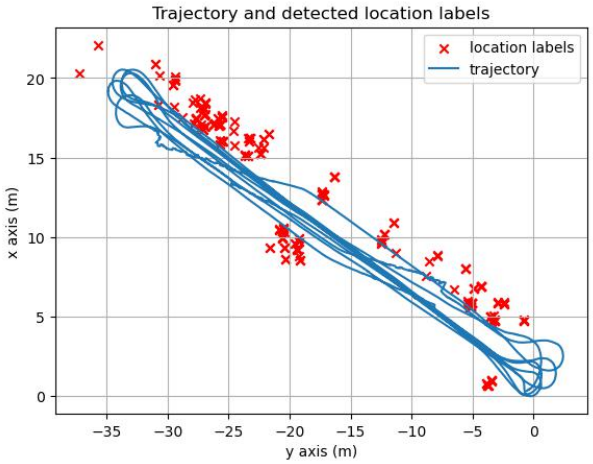

Fig. 6. The trajectory is the pose estimates from the VIO and the red markers are the location labels. The drift in the VIO can be observed from this image as the camera moves around the aisle, and the location labels that are previously detected are detected in a different place. Ideally, if there is no drift there should only be a straight line of location labels.

VI. Experimental Results

The proposed method is tested on a laptop with an i7-10750H @ 2.6GHz processor and 16GB RAM. The proposed method does not require any GPU hardware. The scripts were developed in Python. In this section, the results obtained from the real-world dataset are discussed.

A. Real world dataset

The dataset captured from the forklift consists of incorrect location label detections (Fig. 6). From the image, it can be observed the same markers when it is revisited do not appear in the position in which it was previously estimated. The incorrect position of the location labels is caused by the drift in the VIO. The incorrect VIO pose and 2D measurement (pixel coordinates) of the location labels are added to the factor graph. The uncertainty in the VIO pose and 2D measurement of the location labels are modelled purely as Gaussian noise and optimized. The graphs are optimized with different noise models to understand the influence of the noise model on the optimized result.

Fig. 7 shows the result obtained after optimization with the uncertainty of 0.005m in translation and 0.034 rad in rotation. The results shown in the figure, are obtained with prior only on their height and without any loop closure constraints or line factor constraints. From the image, it can be seen, that the results are worse than the actual measurements added to the graph.

The uncertainty in the translation is changed to 0.05m and the rotation uncertainty is kept at 0.034rad. Fig. 8 shows the result after optimization. Comparing to fig. 7, fig. 8 has improved results. However, the trajectory is curved which is not the actual case since the forklift moves in a straight line during the scanning. Also, the location label estimates are not aligned in a straight line.

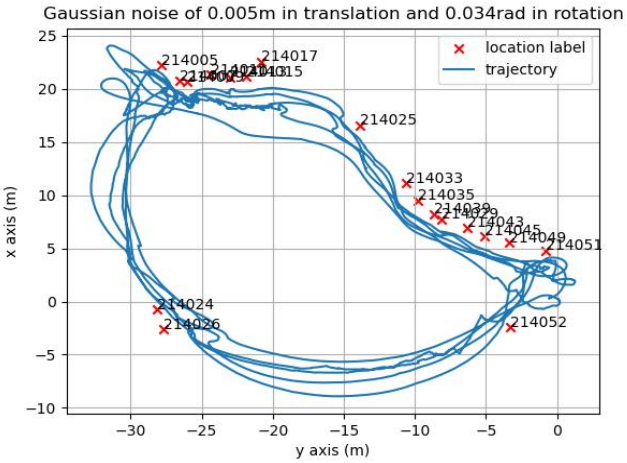

Fig. 7. Gaussian noise with 0.005m and 0.034rad

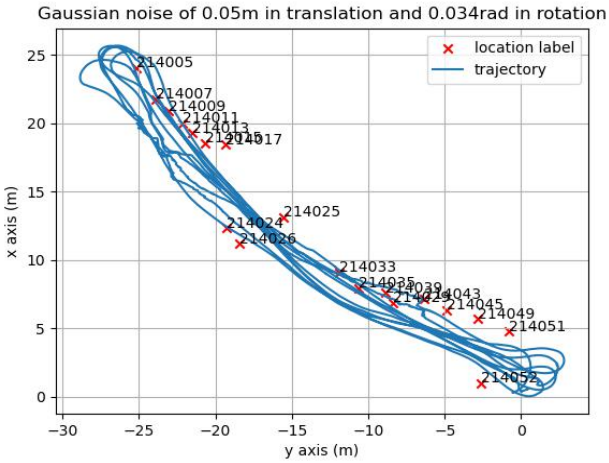

Fig. 8. Gaussian noise with 0.05m and 0.034rad

The uncertainty in the translation is further changed to 0.5m while keeping the uncertainty at 0.034rad. Fig. 9 shows the trajectory after optimization. The trajectory produced after the optimization is still not correct as there is an overlap between trajectory and location labels. Fig. 7, fig. 8 and fig. 9 show that using pure Gaussian noise models to reduce the drift is difficult.

Loop closure is performed in the factor graph in an attempt to reduce the drift in the VIO pose. When a location label is revisited, based on the OCR output, the current viewing pose is connected to the nearest previously visited pose to close the loop. The uncertainty in the loop closure is modelled using a Gaussian noise model and the graph is optimized. Fig. 10 shows closing the loop with the Gaussian noise model does not provide better results. Instead of reducing the drift, the loop closure produces instability in the optimization. The instability might be due to the Gaussian noise model. The results from fig. 10 show that the fixed Gaussian noise model might not be

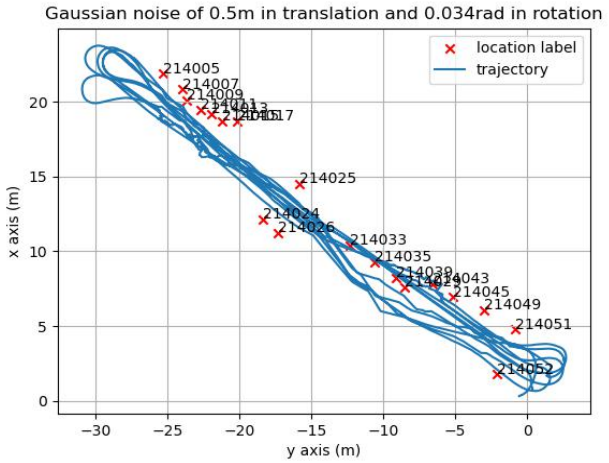

Fig. 9. Gaussian noise with 0.5m and 0.034rad

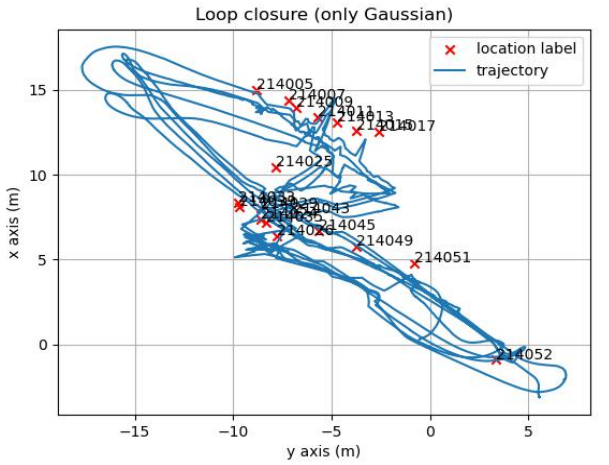

Fig. 10. Loop closure

enough to reduce the drift. This fixed Gaussian noise model problem is solved by using the DCS error function in the GTSAM.

The robust error function DCS is included in the factor graph for the loop closure edges to deal with the outliers. Fig. 11 shows that adding the DCS error function to the loop closure edges provides better trajectory estimates when compared to using purely Gaussian noise models. However, the estimated location labels can seen overlapping on the trajectory and still, the trajectory does not align with the real-world scenario.

The straight-line information is exploited by using the line factor in the factor graph. Fig. 12 shows the results obtained after adding the line factor. From the figure, it can be seen that the location labels align in a straight line which is to be expected after the addition of the line factor to the factor graph. The use of line factor in the factor graph shows that exploiting the known structures in the warehouse can be useful

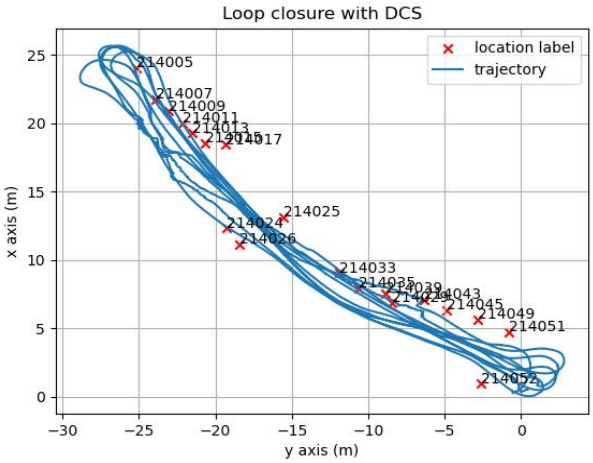

Fig. 11. Loop closure with DCS

in the estimation of the position of the location labels. After the addition of the line factor into the factor graph, the influence of the loop closure on the optimized results was tested. The results produced only with line factor and not without any loop closure produced the same results as seen in fig. 12. The results with only the line factor show that loop closures are often not necessary when the structure of the environment is known. The use of only line factor also constitutes reduced computational cost. The time taken to optimize with loop closure including global optimization was **6.99s** and using only line factor was **5.75s**. The reduced computational cost is due to the fact there are no loop closures in the factor graph that affect the previous estimates which are not needed to be optimized again.

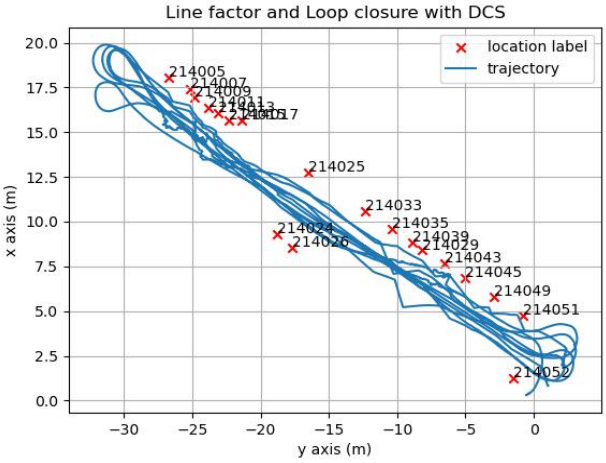

Fig. 12. Line factor

The real-world dataset does not have a ground truth reference to calculate the error after optimization. Here the absolute average positional error in each individual axis is calculated between the trajectory in the raw data and the trajectory

obtained after line factor constraints. The mean difference in the x-axis is **1.592 m** and the mean difference in the y-axis is **0.246 m**. Combining the observations made from fig. 12 and the mean difference in the x and y axes, it can be observed that the drift in the raw dataset has been reduced. Similarly, the absolute average positional error of the location labels in the x and y axes are **1.09 m** and **0.11 m** respectively.

Also, the line factor was tested using a Gaussian noise model and using Cauchy error function. The results were again the same as seen in fig. 12. It is interesting to note that the position of the location labels can be estimated only using pure Gaussian noise models when the structure of the environment is known. The computational time when using the robust error function is **5.75s** and only the Gaussian noise model is **5.9s**. Clearly, using a robust error function allows for faster convergence during optimization.

The experimental results give insight into how the structure of the environment can be modelled using factor graphs to estimate the position of location labels in the warehouse while trying to reduce the drift in the VIO. The results also show that the trajectory lies between the aisle without any overlap on the location labels which resembles the real-world scenario. However, the trajectory estimates are not smooth when compared to the raw data. Since the main goal lies in estimating the position of the location labels this result is acceptable.

VII. Conclusion

This work provides insight into the location label position estimation using cameras. This method shows how the known structures in the warehouse can be used in the factor graphs to improve the position of the location labels. The use of the environment structures in the factor graphs showed that loop closures can be avoided which can reduce the computational time required to perform optimization. In future work, the new factors should be investigated based on the warehouse structure to improve the estimation process. For example, the distance between two racks can be converted into a factor to improve the estimation process. Also, the length of the racks can also be added as a factor to improve the position of the locations labels in that particular rack. However, it should be noted that the new factors do not affect the sparsity in the factor graph or else it will lead to increased computational cost. It is also worth investigating the drift model in the VIO as fewer attempts have been made to model the drift. One interesting approach would be to use time series machine learning models to investigate the drift modelling in the VIO.

References

[1] T. Qin, P. Li and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004-1020, Aug. 2018, doi: 10.1109/TRO.2018.2853729.

[2] Frank Dellaert; Michael Kaess, Factor Graphs for Robot Perception , now, 2017.

[3] Dellaert, Frank. "Factor graphs: Exploiting structure in robotics." Annual Review of Control, Robotics, and Autonomous Systems 4 (2021): 141-166.

[4] F. Dellaert and G. Contributors, borglab/gtsam. Georgia Tech Borg Lab, 2022. doi: 10.5281/zenodo.5794541.

[5] M. Beul, D. Droeschel, M. Nieuwenhuisen, J. Quenzel, S. Houben and S. Behnke, "Fast Autonomous Flight in Warehouses for Inventory Applications," in IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3121-3128, Oct. 2018, doi: 10.1109/LRA.2018.2849833.

[6] I. Kalinov, E. Safronov, R. Agishev, M. Kurenkov, and D. Tsetserukou, "High-Precision UAV Localization System for Landing on a Mobile Collaborative Robot Based on an IR Marker Pattern Recognition," IEEE Xplore, Apr. 01, 2019. https://ieeexplore.ieee.org/abstract/document/8746668 (accessed Dec. 08, 2022).

[7] I. Kalinov et al., "WareVision: CNN Barcode Detection-Based UAV Trajectory Optimization for Autonomous Warehouse Stocktaking," in IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 6647-6653, Oct. 2020, doi: 10.1109/LRA.2020.3010733.

[8] W. Kwon, J. H. Park, M. Lee, J. Her, S. -H. Kim and J. -W. Seo, "Robust Autonomous Navigation of Unmanned Aerial Vehicles (UAVs) for Warehouses' Inventory Application," in IEEE Robotics and Automation Letters, vol. 5, no. 1, pp. 243-249, Jan. 2020, doi: 10.1109/LRA.2019.2955003.

[9] I. Kalinov, D. Trinitatova and D. Tsetserukou, "WareVR: Virtual Reality Interface for Supervision of Autonomous Robotic System Aimed at Warehouse Stocktaking," 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 2021, pp. 2139-2145, doi: 10.1109/SMC52423.2021.9659133.

[10] Rafael Muñoz-Salinas, Manuel J. Marín-Jimenez, R. Medina-Carnicer, SPM-SLAM: Simultaneous localization and mapping with squared planar markers, Pattern Recognition, Volume 86, 2019, Pages 156-171, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2018.09.003.

[11] B. Pfrommer and K. Daniilidis, "TagSLAM: Robust SLAM with Fiducial Markers," arXiv.org, Oct. 01, 2019. https://arxiv.org/abs/1910.00679

[12] R. Muñoz-Salinas and R. Medina-Carnicer, "UcoSLAM: Simultaneous localization and mapping by fusion of keypoints and squared planar markers," Pattern Recognition, vol. 101, p. 107193, May 2020, doi: https://doi.org/10.1016/j.patcog.2019.107193.

[13] C. Chen, H. Zhu, M. Li, and S. You, "A Review of Visual-Inertial Simultaneous Localization and Mapping from Filtering-Based and Optimization-Based Perspectives," Robotics, vol. 7, no. 3, p. 45, Aug. 2018, doi: 10.3390/robotics7030045.

[14] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," The International Journal of Robotics Research, vol. 34, no. 3, pp. 314–334, Dec. 2014, doi: https://doi.org/10.1177/0278364914554813.

[15] M. Bloesch, S. Omari, M. Hutter and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015, pp. 298-304, doi: 10.1109/IROS.2015.7353389.

[16] T. Qin, P. Li and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004-1020, Aug. 2018, doi: 10.1109/TRO.2018.2853729.

[17] R. Jiang, R. Klette and S. Wang, "Modeling of Unbounded Long-Range Drift in Visual Odometry," 2010 Fourth Pacific-Rim Symposium on Image and Video Technology, Singapore, 2010, pp. 121-126, doi: 10.1109/PSIVT.2010.27.

[18] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 3281-3288, doi: 10.1109/ICRA.2011.5979641.

[19] F. Dellaert and M. Kaess, Factor Graphs for Robot Perception. Hanover, MA: Now Publishers, 2017.

[20] Gim Hee Lee, Fraundorfer, F. and Pollefeys, M. (2013) 'Robust pose-graph loop-closures with expectation-maximization', 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems [Preprint]. doi:10.1109/iros.2013.6696406.

[21] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss and W. Burgard, "Robust map optimization using dynamic covariance scaling," 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 2013, pp. 62-69, doi: 10.1109/ICRA.2013.6630557.

[22] Zhang, Z. (1997) 'Parameter estimation techniques: A tutorial with application to conic fitting', Image and Vision Computing, 15(1), pp. 59–76. doi:10.1016/s0262-8856(96)01112-2.