PFG 2016 / 0, 0000 – 0001
Stuttgart, August 2016

Article

Sensor Fusion for Self-Localisation of Automated Vehicles

CHRISTIAN MERFELS & CYRILL STACHNISS

Keywords: sensor fusion, state estimation, chain pose graph, localization, automated driving

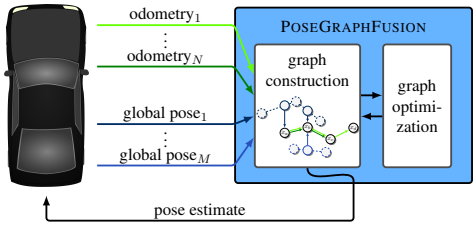
Summary:
Accurate pose estimation is key for a large number of real world applications. For example, automated cars require fast, recent, accurate, and highly available pose estimates for robust operation. Multiple redundant and complementary localisation systems are therefore installed on most automated vehicles. This work proposes a novel multi-sensor pose fusion approach for generically combining measurements from multiple localisation systems into a single pose estimate. We formulate our approach as a sliding window pose graph and enforce a particular chain graph structure, which enables efficient optimisation and a novel form of marginalisation. Our pose fusion approach scales from a filtering-based to a batch solution by increasing the size of the sliding window. It also adapts online to the available computational resources to guarantee high availability. We evaluate our approach on simulated data as well as on real data gathered with a prototype vehicle and demonstrate that our solution runs at 20 Hz, provides timely estimates, is accurate, and yields high availability.

Zusammenfassung: Eine genaue Lokalisierung ist zentral für viele reale Anwendungen. So benötigen beispielsweise automatisierte Autos häufige, aktuelle, präzise und hochverfügbare Schätzungen der Fahrzeugpose. Daher sind auf den meisten automatisierten Fahrzeugen mehrere redundante und komplementäre Ortungssysteme integriert. In diesem Artikel beschreiben wir eine neuartige generische Ortungsfusion zum Zusammenführen der Positionsmessungen von unterschiedlichen Ortungssystemen. Wir konstruieren dazu einen Posengraphen über das Zeitfenster der letzten Odometrie- und Positionsmessungen, so dass der Aufbau des Graphen einer Kettenstruktur entspricht. Diese ermöglicht ein effizientes Optimieren und eine neue Art der Marginalisierung. Unser Ortungsfusionsansatz skaliert von einer filter-basierten bis zur Batch-Lösung durch das Vergrößern des Zeitfensters. Er passt sich außerdem zur Laufzeit an die verfügbaren Rechenressourcen an, um eine hohe Verfügbarkeit der Posenschätzungen zu gewährleisten. Wir evaluieren unseren Ansatz mit Hilfe simulierter und auf einem realen Prototypen aufgezeichneter Daten und zeigen, dass unsere Lösung mit 20 Hz läuft, aktuelle Schätzungen bereitstellt, sowie präzise und hochverfügbar ist.

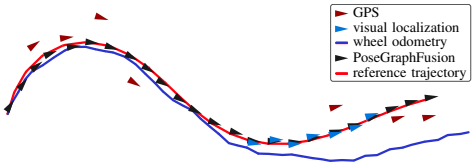
www.schweizerbart.de
DOI: 00.0000/0000-0000/2016/0000
0000-0000/16/0000 $ 0.00

1 Introduction

Automated vehicles and advanced driver assistance systems rely on a precise knowledge of the vehicle's pose, i.e., the position and heading of the vehicle. In the past, different localisation systems have been proposed to solve the localisation task that are usually tailored to a specific sensor set, which typically includes global navigation satellite system (GNSS) receiver, vision, or lidar sensors. Different types of localisation systems and sensors have individual failure modes such as GNSS-denied regions for GNSS receivers or darkness for visual systems, in which the performance degrades substantially or the system fails to provide a reasonable estimate of the vehicle's pose. We approach this problem by proposing a generic pose fusion approach, which is able to effectively merge several relative and global pose sources (see Fig. 1). This enables the combination of orthogonal or redundant pose sources and has the potential to increase the availability, reliability, and accuracy of the resulting localisation, adding also to its versatility and fail-safe behaviour.



(a) Multiple odometry and global pose sources are fused in a graph-based optimisation to provide a single pose estimate.



(b) Three pose sources are used to estimate the true trajectory (red) of a vehicle. The estimated poses are shown as black triangles: the goal is to approximate the unknown red line as closely as possible with the black triangles.

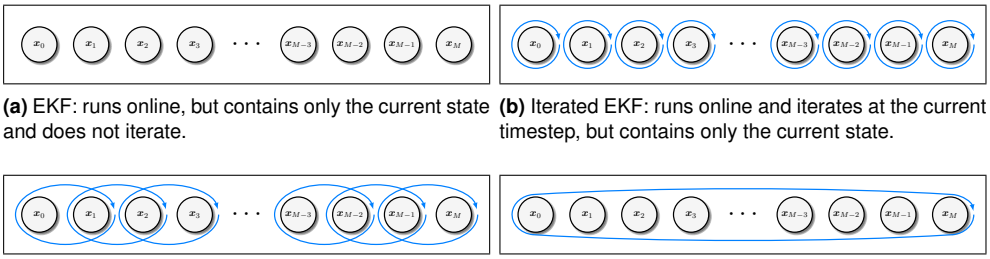
Fig. 1: Overview of the proposed multi-sensor data fusion.

Odometry and global pose sources have orthogonal strengths and weaknesses. On the one hand, odometry measurements are usually available at high frequencies and do not require a priori knowledge about the environment. On the other hand, they accumulate drift with growing distance and they are not globally referenced. The properties of estimates from global pose sources are typically converse to this. On the upside, they are globally referenced and their error is independent of the covered distance. On the downside, they are usually only available at low frequency and require a priori knowledge about or preparation of the environment (such as maps or satellite placement). Combining both types of measurements in a sliding window pose fusion allows one to estimate a trajectory which is both globally referenced and locally smooth.

Multi-sensor data fusion for navigation systems enables the integration of information from multiple sources to estimate the state of the system. This is conventionally achieved by using filtering-based approaches such as the Kalman filter and its variants or, alternatively, smoothing algorithms. Using all pose measurements and optimising the entire trajectory in a non-causal way is commonly referred to as batch estimation or bundle adjustment. It leads to a maximum likelihood (ML) estimate over the joint probability of vehicle poses and produces a statistically optimal result. A drawback of this approach is that the state vector grows unboundedly over time, thus limiting its online applicability. The Extended Kalman Filter (EKF) approaches this issue by restricting the state vector to the most recent state, hence collapsing the trajectory estimation into a single pose estimation problem. This, however, prevents relinearisation of previous states as they are already marginalised out. Also, the current state is not relinearised and the Jacobians are evaluated only once. The Iterated EKF (IEKF) solves this drawback by iterating and relinearising the solution of the current state, thus converging to the optimal state estimate. However, older

states are not explicitly available due to the involved marginalisation, which implies that they cannot be relinearised and refined anymore. The Sliding Window Filter (SIBLEY et al. 2010) weakens this drawback by extending the state vector to the set of the most recent states. It can be seen as an IEKF with an augmented state vector because the filter update of the IEKF is for many problems mathematically identical to the Gauss-Newton method (BELL & CATHEY 1993) when both the prediction and update steps are iterated (BARFOOT 2016).

Fig. 2 illustrates the relationship between the (Iterated) EKF, the Sliding Window Filter, and batch estimation. Motivated by the need for a powerful estimator and constrained by the requirement of an online solution, we design our pose estimation with the same key concepts as a Sliding Window Filter.



(a) EKF: runs online, but contains only the current state and does not iterate.

(b) Iterated EKF: runs online and iterates at the current timestep, but contains only the current state.

(c) Sliding Window Filter: runs online and iterates over the set of most recent states.

(d) Batch estimation: iterates over the entire trajectory, but runs offline and not in constant time.

Fig. 2: Comparison of iterative state estimation techniques. The figure is inspired by BARFOOT (2016, Fig. 4.17).

In this paper, we present an approach to multi-sensor data fusion that decouples the localisation from the fusion task and can be executed online. A key advantage of decoupling the fusion from the localisation is the ability to incorporate third-party localisation modules for which source code is unavailable. The *PoseGraphFusion* enables the combination of multiple global pose sources (e.g., GNSS receivers) with multiple odometry sources (e.g., wheel odometry) by formulating the fusion as a joint optimisation problem. The optimised trajectory then provides the fused estimate of the current pose, as illustrated in Fig. 1a and Fig. 1b. The contribution of this paper is an online pose estimation algorithm based on fusing multiple pose sources. The algorithm avoids overconfidence by performing delayed marginalisation. The pose estimation is formulated as a sliding window graph-based optimisation, which leads to the ML estimate over the joint probability of vehicle poses in the current window. It converges to the online batch estimate for increasing sizes of the sliding window, and provides the full batch estimate when executed offline. Our pose fusion combines exchangeable input sources in a generic way, regardless of what specific type of sensor its measurements originate from. It deals with multi-rate sources, nonconstant input frequencies, out-of-sequence estimates, and time-varying latencies in a straightforward manner. Efficiency is a major design criterion as the proposed system runs online in an automated vehicle. We therefore explicitly control the computation time by proposing a resource-adaptive optimisation scheme. Different parametrisations make it possible to scale from the (Iterated) EKF to the online batch solution and thus to balance runtime versus accuracy. Parts of this contribution were originally presented as two conference papers (MERFELS et al. 2016; MERFELS & STACHNISS 2016). This paper is a substantially revised version that additionally contributes an approach to resource-adaptive state estimation and further evaluations.

In brief, the key contributions of this paper are:

- the presentation of an efficient sensor fusion algorithm with generic odometry and global pose inputs for pose estimation, which handles different and unknown time behaviour of input sources;
- the description of a graph construction algorithm designed to exploit the problem structure and to produce a sparse block-tridiagonal structure of the system matrix, which therefore offers a fast solution;
- the design of a resource-adaptive state estimation concept to meet runtime requirements;
- the insight that marginalisation on such a matrix structure can exactly and efficiently be carried out without an additional fill-in, and that marginalisation can be interpreted as adding a prior node to the graph.

2 Related work

Conventional filtering-based approaches are usually based on some variant of the Kalman filter. For example, KUBELKA et al. (2015) use an error state EKF to fuse information from four different odometry sources: an inertial measurement unit (IMU), track encoders, visual odometry, and laser rangefinder scan-matching. The authors describe the modelling of the fusion strategy as a crucial issue mostly complicated by the significantly different update rates ranging from $0.3\,\mathrm{Hz}$ to $90\,\mathrm{Hz}$. WEISS et al. (2012) propose an EKF to fuse IMU with Global Positioning System (GPS) data and a camera-based pose estimate in the context of micro aerial vehicles. The propagation model is bound to the IMU, making its measurements indispensable, while other sensor measurements can be integrated in a modular way. The authors detail that integrating measurement delays makes it necessary to recompute all affected states. This is computationally feasible for the state vector but infeasible for the covariance matrix, which they therefore neglect in the case of measurement delays. Their work is generalised to a Multi-Sensor-Fusion EKF by LYNEN et al. (2013). It is based on the IEKF formulation where new states are created by high-frequency IMU readings. A lot of effort is taken to align all other, possibly delayed, pose readings with the states. The filtering-based approaches have in common that they rely on the Markov assumption at a very early stage and marginalise all older information, thus prematurely incorporating the linearisation error. STRASDAT et al. (2012) show that mainly because of that reason filtering performs suboptimal even for short time frames when compared to smoothing.

In contrast to filtering techniques, smoothing approaches compute the ML estimate by nonlinear least squares optimisation to a Bayesian network, Markov random field (MRF), or factor graph. Offline batch optimisation of this form assumes additive, white Gaussian noise with zero mean. It considers all available measurements. Online batch optimisation only takes into account all past states up to the current one. Although these operations are both computationally expensive, online batch optimisation becomes feasible through the usage of incremental smoothing techniques, such as iSAM2 (KAESS et al. 2012), that recalculate only the part of the graph that is affected by new measurements.

CHIU et al. (2013) combine a long-term smoother using iSAM2 and a short-term smoother using so-called Sliding-Window Factor Graphs to fuse pose sources. INDELMAN et al. (2012) use the incremental smoothing technique from KAESS et al. (2012) to fuse multiple odometry and pose sources. This implies that they choose a similar graph representation as proposed in this contribution, with the difference that they keep the full graph in memory over the entire trajectory, making the approach more memory-consuming. Long times of operation, as it is common for vehicles, or memory constraints might lead to problems of this approach. CUCCI & MATTEUCCI (2013) propose the graph-based ROAMFREE framework for multi-sensor pose tracking and sensor calibration. They keep the size of the graph bounded by simply discarding older nodes and edges, thus potentially obtaining overconfident estimates.

From a methodical point of view, the approach in this paper is closest to the approach of SIBLEY et al. (2010), who propose the concept of a sliding window filter in the context of robotics.

They apply it to planetary entry, descent, and landing scenarios, in which they estimate surface structure with a stereo camera setup. Our contribution is based on a similar methodology but applies it to our use case of pose fusion as described in our previous work (MERFELS & STACHNISS 2016).

3 Graph-based sensor fusion

Our approach combines pose estimates from multiple sources to compute the current best pose estimate. We consider pose fusion as multi-sensor data fusion for pose estimation. We view pose estimation as the optimisation problem of computing the set of recent poses that define the most likely trajectory given odometry and absolute pose estimates. To this end, we propose a nonlinear least squares estimation implemented in a sliding window fashion for efficient and effective pose estimation.

The input data to the sensor fusion are noisy pose estimates. We assume the noise to be additive, white, and normally distributed with zero mean. Many estimation and data fitting problems can be formulated as nonlinear least squares problems. Our approach exploits the state-of-the-art graph optimisation framework g2o (KÜMMERLE et al. 2011).

The key idea is that given the state vector $\mathbf{x} = (\mathbf{x}_1^\top, \dots, \mathbf{x}_m^\top)^\top$, where $\mathbf{x}_i$ represents the i-th estimated pose of the vehicle, and a set of measurements, where $\mathbf{z}_{ij}$ is the mean and $\mathbf{\Lambda}_{ij}$ the information matrix of a single measurement relating $\mathbf{x}_i$ to $\mathbf{x}_j$ (with $\mathcal{C}$ being all pairs of indices for which a measurement is available), least squares estimation seeks the state

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{\langle ij \rangle \in \mathcal{C}} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^\top \mathbf{\Lambda}_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) \tag{1}$$

that best explains all measurements given the ℓ_2 norm. The vector error function $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ measures how well the measurement $\mathbf{z}_{ij}$ is satisfied and we abbreviate it as $\mathbf{e}_{ij} = \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$. We increase the robustness against outliers by optionally applying a robust cost function, such as the Pseudo-Huber cost function (HARTLEY & ZISSERMAN 2004), on single or all constraints. The information matrices $\mathbf{\Lambda}_{ij}$ are the inverses of the measurement covariance matrices. The blockdiagonal matrix $\mathbf{\Lambda}$ is the result of stacking all $\mathbf{\Lambda}_{ij}$. Using the measurement covariances as weights gives rise to the probabilistic interpretation of the solution as the maximum joint likelihood of the individual estimates (BARFOOT 2016).

Linearising (1) around the initial guess $\breve{\mathbf{x}}$ leads to iteratively solving a linear system with the system matrix $\mathbf{H}$ and the right-hand side vector $\mathbf{b}$ such that

$$\mathbf{H} = \sum_{\langle ij \rangle \in \mathcal{C}} \mathbf{J}_{ij}(\breve{\mathbf{x}})^\top \mathbf{\Lambda}_{ij} \mathbf{J}_{ij}(\breve{\mathbf{x}}), \tag{2}$$

$$\mathbf{b}^\top = \sum_{\langle ij \rangle \in \mathcal{C}} \mathbf{e}_{ij}^\top \Big|_{\mathbf{x}=\breve{\mathbf{x}}} \mathbf{\Lambda}_{ij} \mathbf{J}_{ij}(\breve{\mathbf{x}}), \tag{3}$$

where $\mathbf{J}_{ij}(\breve{\mathbf{x}})$ refers to the Jacobian of the error function evaluated at $\breve{\mathbf{x}}$.

We briefly present the equivalent problem description and notation from the point of view of geodetic mapping to highlight the close relationship of this approach in both the robotics and geodetic community. The graph-based optimisation described above can be linked to the well-known normal equation matrices of parametric adjustment models. Here, $\mathbf{x}$ represents the parameter vector that we want to infer from the vector of observations $\mathbf{L}$. It has the associated covariance matrix $\mathbf{\Sigma}_{ll}$, which constitutes the stochastical model together with the cofactor σ_0^2 and the cofactor matrix $\mathbf{Q}_{ll}$, i.e., $\mathbf{\Sigma}_{ll} = \sigma_0^2 \mathbf{Q}_{ll}$. The inverse of the cofactor matrix is the above mentioned information matrix $\mathbf{\Lambda} = \mathbf{Q}_{ll}^{-1}$. In adjustment models, this matrix is typically called weighting matrix $\mathbf{P}$. The vector $\mathbf{l} = \mathbf{L} - \mathbf{L}_0$ is the difference between the observations $\mathbf{L}$ and

the approximated observation vector $\mathbf{L}_0$. The terms $\mathbf{J}_{ij}(\check{\mathbf{x}})$ are stacked into the design matrix $\mathbf{A}$. The least squares adjustment requires iteratively estimating the parameter corrections $\hat{\mathbf{x}}$ by solving $\mathbf{A}^\top\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^\top\mathbf{P}\mathbf{l}$. Note that we can identify $\mathbf{H}$ with $\mathbf{A}^\top\mathbf{P}\mathbf{A}$ and $\mathbf{b}$ with $-\mathbf{A}^\top\mathbf{P}\mathbf{l}$. For further information on the relationship of graph-based optimisation in the robotics community to geodetic mapping approaches, we refer the reader to AGARWAL et al. (2014a,b). In the remainder of this paper, we stick with the graph-based notion on the nonlinear least squares method.

3.1 *Sliding window chain pose graph fusion*

In contrast to general nonlinear least squares estimation, which is commonly taking into account all available information within the full pose graph, for an online state estimation system it is necessary to limit the considered information to keep the problem computationally tractable. The proposed approach achieves this by marginalising out mature state variables, and thus only considering a fixed amount of state variables. More formally, in a sliding window pose graph the state vector $\mathbf{x}$ is reduced to the M most recent poses $\mathbf{x} = (\mathbf{x}_{t-M+1}^\top, \dots, \mathbf{x}_t^\top)^\top$. Therefore, the size of the system matrix $\mathbf{H}$ is bounded by $\mathbb{R}^{3M\times 3M}$, because a pose $\mathbf{x}_t$ is represented as a three-dimensional vector. The optimisation result from the last time step provides the initial guess for the optimisation in the current time step. This leads to an effective and efficient solution in practice, because a single optimisation is usually sufficient to integrate the additional information of the current time step.

The state variables consist of the position and heading of the vehicle. They are defined in a two-dimensional Cartesian coordinate system for which we choose the Universal Transverse Mercator (UTM) coordinate system. We convert different but prevalent global coordinates (such as pairs of latitude and longitude in the World Geodetic System) to UTM to support a wide range of input sources. We refer to pose sources which measure poses in UTM as *global pose sources* (e.g., GPS), and to poses in this system as *global poses*. Pose sources which measure spatial transformations relative to the previous pose are dubbed *local pose sources* or simply odometry.

In the spirit of MRFs, we refer to nodes representing state variables as *hidden nodes*. In contrast, *global* pose constraints are encoded in so-called *observed nodes* and denoted by $\bar{\mathbf{x}}_i$. They are connected to hidden nodes to constrain them in the global coordinate frame. In the graph representation, these constraints are expressed as edges between the observed and the corresponding hidden nodes. An edge embodies a probability distribution over the relative transformation between the connected nodes, consisting of a mean vector $(\Delta x, \Delta y, \Delta\theta)$ and an associated uncertainty matrix.

Intuitively, the graph-based representation of the nonlinear least squares problem can be understood as a mass-spring model (GOLFARELLI et al. 1998; BARFOOT 2016) where the nodes represent masses and the edges represent springs. The optimal posterior solution of the graph corresponds to the minimum energy state of the system with respect to the energy stored in the springs. The observed nodes "pull" the hidden nodes towards them because the corresponding edges always contain a relative transformation to the hidden node of $(0, 0, 0)$. Their constraints therefore equal zero if and only if the pose of the hidden nodes are identical to the pose of the connected observed nodes. Similarly, we map odometry measurements to edges between hidden nodes. They "push" or "pull" the hidden nodes (depending on their current configuration) to relative poses which are equal to the relative transformation encoded in the edge.

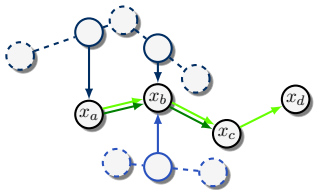
Our graph construction approach is designed such that it enforces a matrix structure for $\mathbf{H}$ (see (2)) that is particularly easy to solve, includes all measurement sources in a generic way independently of their specific output frequencies, and relates the number of state variables to the length of the interval of the sliding window. In contrast to related graph-based approaches (CUCCI & MATTEUCCI 2013; SIBLEY et al. 2010), we neither generate a hidden node every time a measurement arrives nor tie its generation to a specific pose source. Instead, we construct a hidden node every time step, i.e., after every Δt seconds (the *temporal resolution*). This is advantageous because the graph construction does not fail if this specific pose source fails to provide measurements,

and we can define the length of the sliding window without considering the output frequency of this specific pose source. For each hidden node, we additionally query all global pose sources for measurements and interpolate one observed node per source at the timestamp of the hidden node if measurements are available before and after the timestamp.

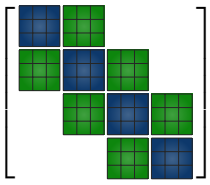
The integration of odometry information into the pose graph can be distinguished into two different approaches. A straightforward way consists in adding edges such that the timestamps of corresponding nodes conform as closely as possible to the timestamps of the measurements. We do not follow this approach as it means that edges can potentially connect hidden nodes that are not adjacent. This in turn results in an unrestricted graph structure, because with a growing number of odometry sources, any two hidden nodes can potentially be connected, and we will detail the downsides of this approach after describing ours. A simple example for such a non-chain pose graph is given in Fig. 4.

Instead of following the strategy just described, we propose to query each odometry source to interpolate the edges between each two successive hidden nodes. The interpolation has to take into account the law of propagation of uncertainty as detailed by SCHLEGEL et al. (2012). We refer to the resulting form of the graph as *chain pose graph*. Constructing edges from odometry measurements at most between successive hidden nodes via interpolation is crucial, as there are significant gains in terms of runtime and memory efficiency when using chain pose graphs. We derive these by analysing how the matrix structure of $\mathbf{H}$ is defined by our chain pose graph construction.

The block structure of $\mathbf{H}$ reflects the connections of poses and edges in the graph as it is its adjacency matrix (KÜMMERLE 2013). Its structure changes slightly with the availability of measurements. In general, the block structure of a chain pose graph is a block-tridiagonal matrix. An example graph in Fig. 3 illustrates how to integrate multiple hidden and observed nodes as well as odometry constraints. It additionally shows the resulting block-tridiagonal matrix structure of the corresponding system matrix. The diagonal entries in the system matrix are influenced by odometry and global pose constraints, while the off-diagonal entries are only affected by odometry constraints. Loop closures are not considered as they arise rarely when driving straight from destination to target. Moreover, they would also require to break with the treatment of input sources as black boxes, because the fusion cannot decide in place of a pose source (without knowledge of its internal functionality) whether it recognizes a place sufficiently well to warrant a loop closure edge.



(a) A chain pose graph.

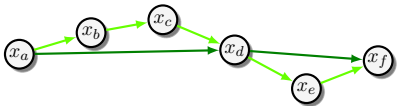


(b) Corresponding block-tridiagonal structure of the system matrix.

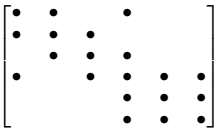
Fig. 3: A chain pose graph and the corresponding structure of the system matrix. a) The black circles are hidden nodes, the dashed blue circles are global pose measurements from two different sources, the non-dashed blue circles are observed nodes, and the green edges are odometry constraints. Note how the raw global pose estimates are interpolated (dashed blue lines) at the same timestamps as the hidden nodes to obtain the observed nodes. b) The structure of $\mathbf{H}$ for the graph in (a).

The block-tridiagonal structure is a consequence of the linear temporal ordering of the state variables combined with the fact that edges are constructed at most between successive nodes. We specifically design our solution to produce a block-tridiagonal matrix because this structure does

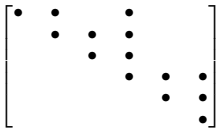
not produce fill-in in $\mathbf{H}$ after marginalisation of the oldest state variables (see Section 3.4). Beyond that, even the Cholesky factorisation $\mathbf{H} = \mathbf{R}^\top \mathbf{R}$, which we perform to solve the linear system, does not suffer from fill-in in its triangular matrix $\mathbf{R}$. In fact, $\mathbf{R}$ becomes a band matrix. As a consequence, costly variable reordering techniques (AGARWAL & OLSON 2012) are unnecessary as $\mathbf{R}$ already contains the minimum number of nonzero elements necessary to reconstruct $\mathbf{H}$. Fig. 4 and Fig. 5 visualise the different effects for factorisation and marginalisation of a non-chain pose graph compared to a chain pose graph. By comparing them it becomes apparent that $\mathbf{H}$ and $\mathbf{R}$ have less nonzero elements when constructing a chain pose graph. Moreover, $\mathbf{H}$ does not suffer from fill-in after marginalisation.



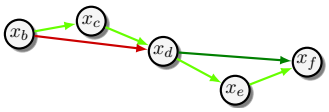
(a) An example graph that is not a chain pose graph.



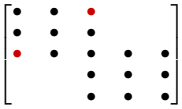
(b) Sparsity pattern of the system matrix $\mathbf{H}$ for the graph in (a).



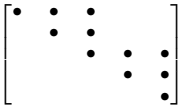
(c) Sparsity pattern of the Choleksy factorisation $\mathbf{R}$ for the matrix in (b).



(d) Graph structure after marginalising x_a. A new edge due to the marginalisation is displayed in red.



(e) Sparsity pattern of the system matrix $\mathbf{H}$ for the graph in (d). Fill-in due to the marginalisation is marked in red.



(f) Sparsity pattern of the Cholesky factorisation $\mathbf{R}$ for the matrix in (e).

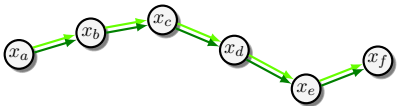
Fig. 4: Factorisation and marginalisation of a non-chain pose graph. Marginalising x_a leads to a new edge in the graph (namely from x_b to x_d) and thus to a denser system matrix. This reduces its sparsity and consequently decreases runtime performance. Also, it is necessary to apply a variable reordering strategy to reduce the number of nonzero elements in $\mathbf{R}$.

Furthermore, the computational complexity of the Cholesky factorisation of a block-tridiagonal matrix is $\mathcal{O}(n)$ (with n being the number of nonzero entries in $\mathbf{H}$). This is a substantial improvement as for arbitrary (dense) matrix structures their decomposition or inversion becomes as costly as approximately $\mathcal{O}(n^{2.4})$. For experiments concerning the runtime for general graph-based optimisation with g2o, we refer the reader to the runtime evaluations of**?**Fig. 8]Kummerle2011, where the time complexity for optimising the Manhattan3500 dataset (OLSON et al. 2006) is clearly higher than linear in the number of nodes although the same sparse matrix techniques as in our implementation were used. For our approach, we demonstrate the linear time complexity for a practical experiment in the evaluations section.

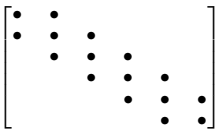
In summary, our chain pose graph approach prevents fill-in after marginalisation in $\mathbf{H}$ and during the factorisation in $\mathbf{R}$, makes common variable reodering strategies unnecessary, and is efficiently solvable in $\mathcal{O}(n)$.

3.2 *Time behaviour*

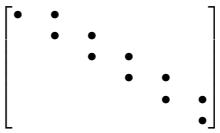
After detailing how and for which timestamps we construct hidden and observed nodes in Section 3.1, we turn our attention to the questions how our system handles the time behaviour of input



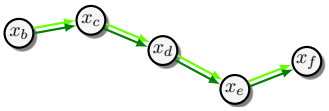
(a) A chain pose graph. Odometry measurements are broken down such that they create edges between adjacent hidden nodes.



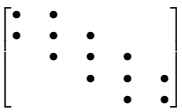
(b) Sparsity pattern of the system matrix $\mathbf{H}$ for the graph in (a).



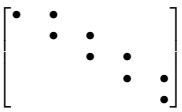
(c) Sparsity pattern of the Cholesky factorisation $\mathbf{R}$ for the matrix in (b).



(d) Graph structure after marginalising x_a. The marginalisation does not result in new edges.



(e) Sparsity pattern of $\mathbf{H}$ for the graph in (d). The marginalisation does not result in fill-in.



(f) Sparsity pattern of the Cholesky factorisation $\mathbf{R}$ for the matrix in (e).

Fig. 5: Factorisation and marginalisation of a chain pose graph. Marginalising x_a does not lead to new edges in the graph or fill-in in the system matrix. Variable reordering strategies are unnecessary as $\mathbf{R}$ contains the minimum number of nonzero elements necessary to reconstruct $\mathbf{H}$.

sources and how we design the time behaviour of the pose fusion. In this paper, we define time behaviour as the latency, frequency, and availability of estimates.

Integrating input sources with unknown time behaviour is difficult as we deal with multi-rate sources, nonconstant input frequencies, out-of-sequence estimates, and time-varying latencies. Our approach consists in buffering and preprocessing all incoming data. This does not introduce any delay as we do not need the measurements before the next graph construction phase. The preprocessing includes detecting missing pose estimates by estimating the recent input frequency of each source and comparing the number of estimates we should have received to the number of estimates we actually received. Sorting the data by time enables the integration of out-of-sequence data.

Instead of being data-triggered, the output of the PoseGraphFusion is time-triggered. Its output frequency f is decoupled from the temporal resolution Δt, i.e., the difference between the timestamps of two adjacent hidden nodes. Every $\frac{1}{f}$ the cycle of graph construction and optimisation is triggered. The time spent for constructing and optimising the graph is summed up as the computation time c_t. The most recent state is estimated with a sliding window pose graph over the current set of measurements. It is subsequently propagated into the future to the start of the next cycle as depicted in Fig. 6 with a constant turn rate and velocity model. The propagation ensures that at that point, the pose estimate just computed can be sent out directly. This in turn guarantees a low latency of the pose estimate, where we define the latency to be the difference of the time when the pose has been computed and the time for which it is valid. Depending on the application, a slightly higher latency might be tolerable in exchange for a propagation-free pose estimate, in which case the pose propagation is turned off.

A conventional Kalman filtering approach has difficulty to generically incorporate multiple input sources with unknown time behaviour. This is best seen for out-of-sequence measurements: to integrate such a measurement (which is valid for some point in time in the past), it has to propagate its state back in time to the time of the measurement, apply the measurement, and reapply all other stored measurements. This is described in more detail by BAR-SHALOM (2002).

Measurements from multiple sources with different delays cause the same effect for a data-

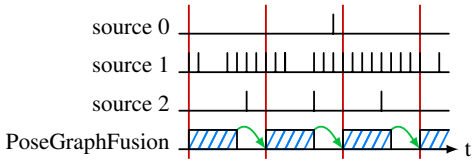


Fig. 6: Time behaviour of the input data for three examples sources 0 to 2 and the corresponding time behaviour of the output of the PoseGraphFusion. The input sources show occasional activity (source 0), data dropouts (source 1), and different data rates. The PoseGraphFusion is able to handle these characteristics and incorporates all input data by first buffering it. At the start of each cycle (directly after the red vertical line), the graph is constructed and optimised. The time necessary for that is the computation time c_t (blue hatched). After the optimisation, the estimated pose is propagated to the beginning of the next cycle (green arrows) so that it can be transmitted immediately.

triggered Kalman filter. This is because pose sources with a higher delay act like out-of-sequence measurements when compared to a pose source with a lower delay whose measurements have already been processed. A pose graph approach with the described characteristics admittedly introduces additional inaccuracies due to the interpolation of measurements, but it does not suffer from repeated backward-forward computations, and is able to elegantly resolve time behaviour issues and treat all sources in a homogeneous manner.

3.3 *Resource-adaptive state estimation*

In Section 3.1 we have shown that the PoseGraphFusion has a linear runtime complexity in the number of hidden nodes, and we have detailed its time behaviour in Section 3.2. We combine this knowledge to enable the algorithm to adapt its parameters at runtime such that it does not use more computational resources than available. This becomes especially crucial when deploying the software in multiple vehicles with different hardware environments such that one parameter set does not fit all of them, or when the hardware resources are shared with other software.

These resource constraints translate to the problem that it takes different amounts of time to solve the same problem on different hardware or when the available CPU time fluctuates over time. On the one hand, this potentially results in $c_t > \frac{1}{f}$, i.e., the computation time is longer than the available time in one cycle, which causes a decreased and fluctuating output frequency. On the other hand, the estimation quality is suboptimal if $c_t \ll \frac{1}{f}$ as more hidden nodes could have been considered in the optimisation process in the remaining cycle time (note that the estimation quality increases with the number of hidden nodes, see Section 4.4 in the evaluation section). Ideally, $c_t = \frac{1}{f}$ for all time steps t. In practice, $\Delta c_t = \frac{1}{f} - c_t \neq 0$ where Δc_t is the remaining cycle time (note that Δc_t can be negative).

The goal is to obtain on average $\Delta c_t \approx 0$ by letting the PoseGraphFusion adapt the number of hidden nodes M_t in its optimisation problem. This results in automatic adaptation to new hardware environments. Moreover, it also adjusts to temporary available or denied CPU resources at runtime.

Our solution consists of using concepts from linear control theory as our system has linear runtime complexity in the number of hidden nodes. We create a feedback loop by measuring the computation time c_t for the current time step with a known number of hidden nodes M_t, and subsequently deriving a new number M_{t+1} such that $\Delta c_{t+1} \approx 0$. We employ a proportional-integral-derivative (PID) controller for this task, which controls M_{t+1} according to

$$M_{t+1} = K_\mathrm{p} \Delta c_t + K_\mathrm{i} \sum_{i=1}^{t} \frac{\Delta c_i}{f} + K_\mathrm{d} (\Delta c_t - \Delta c_{t-1}) f, \tag{4}$$

where K_p, K_i, and K_d are the proportional, integral, and derivative gain constants. These are empirically determined and left constant during the runtime. Note that the sum $\sum_{i=1}^{t} \frac{\Delta c_i}{f}$ does not have to be recomputed from scratch in every time step. Instead, at time step t it is sufficient to add the term $\frac{\Delta c_t}{f}$ to the sum $\sum_{i=1}^{t-1} \frac{\Delta c_i}{f}$, which is known from the previous time step.

Our resource-adaptive state estimation is similar to anytime algorithms in that the quality of results improves gradually as available computation time increases and that we can control computation time. However, other than anytime algorithms, we have to control computation beforehand and cannot interrupt an optimisation step. The control of the number of hidden nodes leads to a better usage of computational resources, and thus to a higher estimation quality.

3.4 *Marginalisation*

As stated in Section 3.1, it is mandatory to limit the number of hidden nodes to maintain constant runtime complexity. Simply removing edges and nodes leads to information loss and is equivalent to conditioning, which potentially leads to overconfidence. Therefore, we marginalise the oldest hidden nodes. This truncates the graph but retains the same information (given the linearisation point). The common approach for that is computing the Schur complement on the system matrix $\mathbf{H}$. For non-chain pose graphs, the disadvantage of this operation is the introduction of conditional dependencies between state variables that are connected. This is equivalent to additional fill-in in $\mathbf{H}$.

As we design our problem structure to be a chain pose graph, we are able to retain the same sparsity pattern and do not suffer from a denser system matrix after marginalisation. This follows directly from the construction of the chain pose graph, which allows edges to be constructed only between successive nodes. As a consequence, removing the oldest hidden node does not lead to any additional conditional dependencies as it is solely connected to the penultimate hidden node (and possibly, to one or more observed nodes). Marginalising multiple hidden nodes amounts to iteratively marginalising the oldest hidden node. The fact that no new edges are introduced during marginalisation implies that the system structure for chain pose graphs will not become denser over time. Therefore, the memory usage and the time necessary to compute the Gauss-Newton iterations do not depend on the duration of the runtime.

As an alternative to performing the Schur complement, MERFELS & STACHNISS (2016) show how to replace the oldest hidden node with a *prior node* such that it affects the graph in the same way as the Schur complement does. This prior node is equivalent to performing exact marginalisation as it captures the additionally introduced conditional dependencies of the marginalised oldest hidden node. The advantages of using the prior node are, amongst others, that the user can visualise the prior information, adapt its uncertainty, or apply a robust cost function to it.

3.5 *Noise correlations between input sources*

We need to apply appropriate preprocessing if noise is correlated between input sources. This occurs when, for example, different input sources are based on the same sensor data or when the same algorithm runs on two physically different sensors (e.g., two GPS receivers). We explicitly account for correlated noise between sources to avoid overconfident estimates. As our approach is agnostic to the specific type of input source, we perform fusion under unknown correlation and employ Covariance Intersection (JULIER & UHLMANN 1997, 2007; REINHARDT et al. 2012) in a preprocessing step.

The main goal of Covariance Intersection is to find a parameter value $\omega \in [0, 1]$ such that the fused state $\mathbf{x}^\omega$ and covariance matrix $\mathbf{C}^\omega$ of two normally distributed states $\mathbf{x}_1, \mathbf{x}_2$ with associated

covariance matrices $\mathbf{C}_1, \mathbf{C}_2$ are optimal given a certain objective function. After determining ω, the fused result is computed by

$$\mathbf{C}^\omega = \left(\omega \mathbf{C}_1^{-1} + (1-\omega)\mathbf{C}_2^{-1}\right)^{-1}, \tag{5}$$

$$\mathbf{x}^\omega = \mathbf{C}^\omega \left(\omega \mathbf{C}_1^{-1}\mathbf{x}_1 + (1-\omega)\mathbf{C}_2^{-1}\mathbf{x}_2\right). \tag{6}$$

Fig. 7 provides an example of two values of ω for the application of Covariance Intersection on two covariance matrices. Further details on how to compute ω given different objective functions are described by MERFELS et al. (2016).

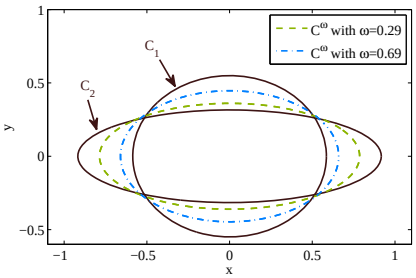


Fig. 7: Covariance Intersection on two covariances matrices $\mathbf{C}_1, \mathbf{C}_2$ with two different parameter values ω. Covariance Intersection provides a consistent uncertainty estimate despite the fact that the noises disturbing the measurements belonging to $\mathbf{C}_1$ and $\mathbf{C}_2$ have an unknown correlation.

In our approach, we identify groups of sources with correlated noise, buffer their measurements, and combine them into a single estimate per group using Covariance Intersection. We repeat this process if there are more than two sources in a single group. These estimates are subsequently entered into the pose fusion process. This results in a reduced number of input sources, in consistent covariance estimates, and avoids the potential threat of divergent or overconfident fusion estimates.

4 Evaluation

The experimental section is designed to support our claims that the presented sensor fusion approach is an efficient pose estimation algorithm, converges accurately towards the online ML estimate, scales from a filtering to a batch least squares solution, handles different time behaviours without difficulty, and adapts online to the available computational resources. We provide experiments on data gathered on a real prototype vehicle and on simulated data.

4.1 *Experiment based on a real prototype vehicle*

The following experiment is designed to show that the pose fusion is able to run online in a car and effectively generate pose estimates given a set of real-world input sources. The prototype vehicle is equipped with a front-facing monocular camera, a front-facing automotive-grade lidar scanner, and four fisheye top view cameras. Three localisation systems are incorporated as global pose sources and one system is considered for odometry. The first global pose source (referenced as *pose source 0*) is computed by matching coarse lidar scans to a globally referenced point cloud. The second global pose source is a GPS receiver (*pose source 1*), whereas the third source (*pose*

source 2) is a visual localisation system, which computes a pose based on comparisons of visual features with a globally referenced feature map. The odometry source is provided by wheel odometry.

The PoseGraphFusion takes these four sources as input and computes pose estimates for the planner. As the noises of the odometry source and the global pose sources are not correlated, we directly fuse them and do not need to apply covariance intersection in this setup. We use $M = 1000$ hidden nodes and set the temporal resolution to $\Delta t = 25\,\mathrm{ms}$, which results in a sliding window over the last $25\,\mathrm{s}$. The test drive was carried out on a route of about $16\,\mathrm{km}$ in rural and urban areas in Germany.

In practice, these sources do not exhibit white Gaussian noises with zero mean. To show this, we evaluate the autocorrelations of their noises and find significant components other than the zero component, meaning that the noises are not independently distributed. Furthermore, the means are nonzero. Moreover, pose sources 0 and 1 are third-party black box modules without proper uncertainty models. We therefore assume constant covariance matrices for their estimates. These are the sample covariance matrices obtained by evaluating the performance of the pose sources on other, similar datasets. Therefore, they are expected to be in the correct order of magnitude, however, they always provide the same static uncertainty estimate regardless of the actual estimate, which is suboptimal.

Despite the input data not being ideal for the presented fusion approach, we demonstrate the applicability of the fusion and show a decent performance. A detailed analysis with the help of simulated data in Section 4.4 evaluates the performance for controlled noise.

4.1.1 Pose estimation quality

We evaluate the estimation quality of the PoseGraphFusion by comparing its output to the batch solution. Fig. 8 shows the development of the position error over time. Pose sources 0 and 1 provide only coarse localisation estimates with RMS errors of $1.06\,\mathrm{m}$ and $1.23\,\mathrm{m}$, respectively, whereas pose source 2 performs better with an RMS error of $0.28\,\mathrm{m}$. The PoseGraphFusion stays close to the performance of the best pose source with an RMS position error of $0.38\,\mathrm{m}$ and heading error of $1.16°$. Theoretically, the output of the fusion is supposed to improve compared to the input sources. However, the noises are not independently distributed with zero mean and the uncertainty models are rough as mentioned above. Additionally, the input pose sources provide estimates in the past and are not always available as we will see in the next evaluation.

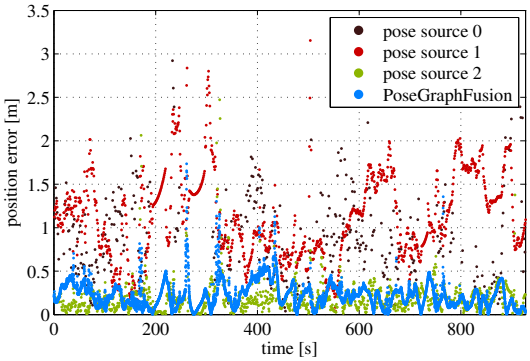


Fig. 8: Position error over time for the three input sources and the fused estimate. The performance of the PoseGraphFusion is linked to the performance of the input sources. Its RMS error is $0.38\,\mathrm{m}$.

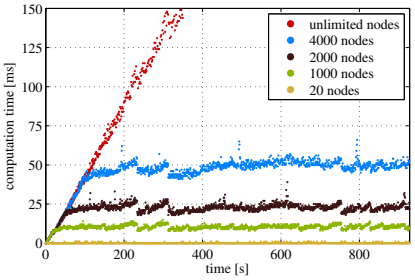
4.1.2 Latency and availability

As for the current real-world experiment, we demonstrate the capability of our approach to combine the different time behaviours of the input sources and generate a well-defined output time behaviour. For this purpose, we examine the latencies of their estimates: the pose sources 0, 1, and 2 all exhibit a latency of up to $0.3\,\mathrm{s}$. This implies that their pose estimates are too old to directly feed them into our planner. In 95% of the time the PoseGraphFusion exhibits a pose latency of $10\,\mathrm{ms}$ or less. Moreover, the pose sources are not always able to calculate and provide a pose estimate due to, for example, sensor or model failures. Consequently, the pose sources have a limited availability, ranging from 66.98% (pose source 0) over 97.76% (pose source 2) to 100% (pose source 1 and odometry) of the time of this experiment. The PoseGraphFusion achieves an availability of 100% of the time. Note that this influences the position accuracy as the pose sources can simply refuse to send a pose estimate in difficult situations, whereas the PoseGraphFusion is bound to deliver.

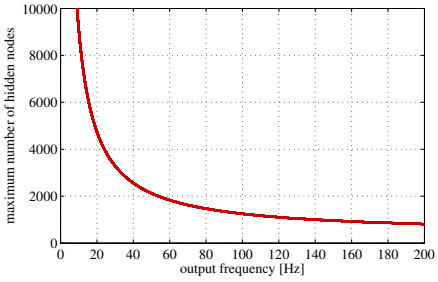
Having shown that our system is capable of integrating different time behaviours into a single well-defined output time behaviour, we study the question whether the proposed algorithm runs fast enough for online usage on a car.

4.1.3 Runtime performance

Next we concern ourselves with the runtime performance of the PoseGraphFusion. The software was repeatedly run on a single core of a laptop with an Intel i7-4800QM processor. Fig. 9a shows the computation time at each time step for different numbers of hidden nodes in the sliding window pose graph. The red curve illustrates the need for limiting the size of the graph as otherwise the computation time grows unboundedly and gets quickly too demanding for a decent output frequency. A choice of $M = 4000$ hidden nodes allows us to set the output frequency to $f \approx \frac{1}{50\,\mathrm{ms}} = 20\,\mathrm{Hz}$. This choice includes some spare time per cycle in case that the computation time is temporarily higher due to a different CPU load (see the fluctuations in Fig. 9a).



(a) Computation time for four input sources. It is roughly constant when the number of nodes is limited.



(b) Parameter space (assuming four input sources). The red curve marks the maximum number of hidden nodes $M_{\max}$ that can still be processed fast enough to attain the corresponding output frequency f.

Fig. 9: Runtime performance.

The near-constant computation time once the graph attains its full size is expected as the optimisation of a chain pose graph of fixed size is constant as detailed in previous sections. Also, the linear increase in computation time before the number of nodes equals M is in line with our theoretical expectations as the solution of a problem in the form of a chain pose graph has a runtime complexity of $\mathcal{O}(n)$. An empirical analysis of the data depicted as a red curve in Fig. 9a (unlimited

number of hidden nodes) reveals the reciprocal relationship between the maximum number of hidden nodes $M_{\max}$ and the attainable output frequency f, as shown in Fig. 9b. We obtained this graph by computing the maximum output frequency for each data point and subsequently fitting a curve $y = \frac{1}{f}$ to it.

Different parametrisations allow us to balance the need for a high output frequency versus the desire for more hidden nodes. This parameter space serves as a basis for choosing f and $M_{\max}$, but we investigate in the next experiment how to react at runtime to situations in which less CPU time than usual is available, i.e., situations in which the relationship depicted in Fig. 9b does not hold temporarily.

4.2 *Resource-adaptive online state estimation*

This experiment is designed to show that the PoseGraphFusion effectively adapts online to the available computational resources. To this end, we perform an experiment on the data which was recorded during the experiment detailed in Section 4.1 and inspect the dynamic adaptation of the number of hidden nodes M_t with the goal of limiting the computation time c_t such that it is on average less or equal to $\frac{1}{f}$, as detailed in Section 3.3. To clearly see the impact of the PID controller, we maximise the CPU load multiple times during the experiment. As a consequence, the computation time increases in these phases because less CPU time is available for the PoseGraphFusion. The PID controller counteracts the exceeding computation time by regulating M_t as can be seen in Fig. 10.

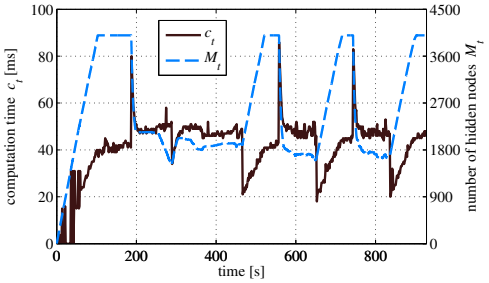


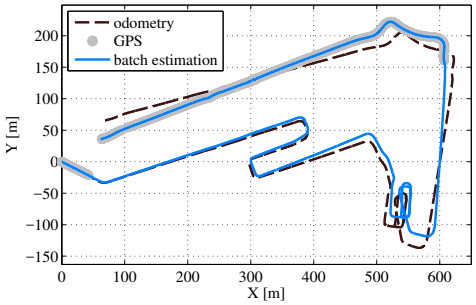
Fig. 10: A PID controller controls the number of hidden nodes to attain a predefined setpoint for the computation time (here: $50\,\mathrm{ms}$). We maximise the CPU load in three sequences and observe how M_t is controlled such that c_t stays as much as possible beneath the computation time limit.

We repeat the experiment with the same CPU load pattern but without the PID controller. The computation time exceeds the limit of $50\,\mathrm{ms}$ by roughly 60% in phases when less CPU time is available. The overall mean computation time amounts to $71\,\mathrm{ms}$ whereas it is $41\,\mathrm{ms}$ with the PID controller enabled. We conclude that the resource-adaptive state estimation in the form of dynamically controlling M_t is essential for meeting a certain computation time limit on average. This in turn is directly related to the availability of the pose estimates.

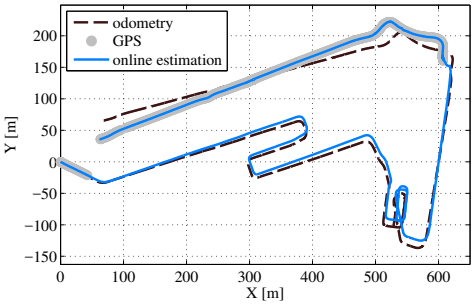
4.3 *Batch estimation*

We also perform batch estimation (offline) using our approach by setting M to infinity so that the sliding window encompasses all time steps. We recorded data on our prototype vehicle and process it offline to support our claims that our approach scales up to an offline batch estimation. The experiment additionally shows that our approach can handle substantial GPS signal dropouts. This is also true for the online state estimation, which we show by applying it to the same data. To obtain the dataset, we recorded roughly $15\,\mathrm{min}$ of odometry and GPS data directly before

entering an underground car park, while driving through it, and at the exit. We filter the valid GPS measurements by requiring at least five satellites to be seen, which results in missing GPS data for the entire transit through the car park. Additionally, it took a while after exiting it before reliable GPS measurements could be acquired. Fig. 11 shows the availability of the GPS signal and the accumulated drift in the raw odometry readings, which amounts to approximately $30\,\mathrm{m}$ and $3.2°$. The figure also shows the optimised global trajectory, once computed offline (Fig. 11a) and once in an online fashion (Fig. 11b).



(a) Our offline batch estimation provides us with the fully optimised trajectory. It takes advantage of the local correctness of odometry and the global correctness of GPS data.



(b) The overall shape of the trajectory of the online estimation is qualitatively comparable to the offline estimation. The most noticeable difference is the transition towards the GPS measurements once they become available again.

Fig. 11: We recorded GPS and odometry data while driving through an underground car park. The GPS signal is only available before entering and shortly after exiting. Both the online and offline estimation are able to handle substantial GPS signal dropouts.

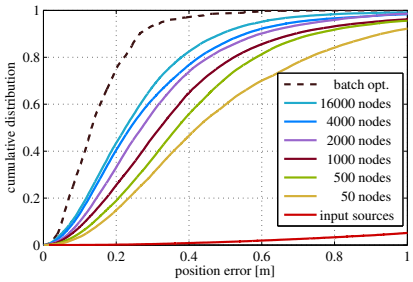
In total, the experiments with data gathered on a real prototype show that the PoseGraphFusion generates a smooth trajectory, continuously provides accurate pose estimates with a low latency at a configurable output frequency under real-world conditions, and is able to scale up the full batch estimation. Two last questions remain unanswered so far: how many hidden nodes are needed at least to fulfil a given accuracy requirement, and do more hidden nodes actually reduce the estimation error? To answer these questions, we perform two experiments with simulated input data.

4.4 *Simulated input data*

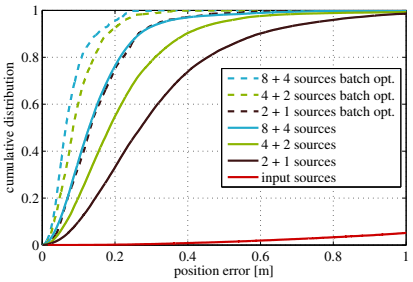
The next experiment is conducted on simulated input data and serves to investigate the estimation quality as a function of the number of hidden nodes M. This is related to the question of the required number of hidden nodes, illustrates the need of delayed marginalisation, and most importantly demonstrates that the sliding window estimate converges to the online batch solution for an increasing number of hidden nodes. All inputs are sampled around the true values from a Gaussian distribution with a standard deviation of $3.0\,\mathrm{m}$, $3.0\,\mathrm{m}$ and $4°$, respectively, in lateral and longitudinal direction and heading orientation of the vehicle. Fig. 12a shows that choices for M with a decent accuracy lie well within the space of possible parametrisations. Automated driving functions commonly require sub-lane localization accuracy, such that the lateral localization error is below $1.25\,\mathrm{m}$. Furthermore, the cumulative distribution of the position error improves for more hidden nodes and approaches the batch optimisation.

The second experiment with simulated data is designed to show that the position error decreases with an increase in the number of sources. The noise terms are identical for all sources

and equal to the values mentioned above. Fig. 12b shows how the fusion of inaccurate pose estimates (red curve) leads to a much more accurate estimate and improves further for an increasing number of input sources. The online combination of eight global pose and four odometry sources is roughly as accurate as the offline batch optimisation of two global pose sources and one odometry source. This result reinforces the motivation for a pose fusion algorithm. In summary, a higher number of hidden nodes and more input sources are both advantageous with respect to the position accuracy given certain noise assumptions.



(a) The cumulative distribution of the position error for different numbers of hidden nodes M. The configurations demonstrate that the accuracy increases for higher values of M, thus showing the need for delayed marginalisation.



(b) The cumulative distribution of the position error for different numbers of input sources. The legend entries *"a + b sources"* denote a global pose sources (e.g., GPS) and b odometry sources.

Fig. 12: Effects of using more hidden nodes (a) or having more input sources (b). The red curve shows an example for the quality of a noisy input source. The batch optimisations (dashed lines) depict the theoretically attainable upper bound of accuracy if future information were available.

5 Conclusion

The pose fusion concept presented in this paper is motivated by the need for fast, recent, accurate, and highly available pose estimates. We approached these requirements by proposing a sliding window graph-based optimisation scheme, which scales from a fast incremental solution to an online batch solution by increasing the sliding window size. Furthermore, we described the design of chain pose graphs for efficient optimisation. These graphs do not suffer from additional fill-in after marginalisation and allow us to collapse all marginalisation information into a prior node. We have shown with experiments on a real vehicle that the PoseGraphFusion is fast, provides timely estimates, is accurate, yields a high availability, and adapts to the available computational resources. The proposed system works with generic input sources and is general enough to be applied to other autonomous systems.

Future lines of research include the investigation how richer error models can be incorporated. This includes studying the influence of nonzero mean noise and how to reduce it. Also, we currently use the sample covariance matrix for pose sources which do not provide uncertainty estimates. In the future, we will explore methods to dynamically estimate these covariance matrices at runtime. Furthermore, we will combine the pose fusion and a Simultaneous Localization and Mapping (SLAM) system. This will allow us to study the potential benefits of the tightly coupled integration of SLAM features with the loosely coupled integration of pose sources.

References

AGARWAL, P. & OLSON, E., 2012: Variable reordering strategies for SLAM. – Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 3844–3850.

AGARWAL, P., BURGARD, W. & STACHNISS, C., 2014a: Helmert's and Bowie's geodetic mapping methods and their relation to graph-based SLAM. – Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). IEEE, 3619–3625.

AGARWAL, P., BURGARD, W. & STACHNISS, C., 2014b: A survey of geodetic approaches to mapping and the relationship to graph-based SLAM. – IEEE Robotics & Automation Magazine **21** (3): 63–80.

BAR-SHALOM, Y., 2002: Update with out-of-sequence measurements in tracking: Exact solution. – IEEE Transactions on Aerospace and Electronic Systems **38** (3): 769–778.

BARFOOT, T. D., 2016: State Estimation For Robotics: A Matrix-Lie-Group Approach. – Draft in preparation for publication by Cambridge University Press.

BELL, B. & CATHEY, F., 1993: The Iterated Kalman filter update as a Gauss-Newton method. – IEEE Transactions on Automatic Control **38** (2): 294–297.

CHIU, H. P., WILLIAMS, S., DELLAERT, F., SAMARASEKERA, S. & KUMAR, R., 2013: Robust vision-aided Navigation using Sliding-Window Factor Graphs. – Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 46–53.

CUCCI, D. A. & MATTEUCCI, M., 2013: A flexible framework for mobile robot pose estimation and multi-sensor self-calibration. – Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO), 361–368.

GOLFARELLI, M., MAIO, D. & RIZZI, S., 1998: Elastic correction of dead-reckoning errors in map building. – Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 905–911.

HARTLEY, R. & ZISSERMAN, A., 2004: Multiple View Geometry in Computer Vision. second edn. – Cambridge University Press.

INDELMAN, V., WILLIAMS, S., KAESS, M. & DELLAERT, F., 2012: Factor graph based incremental smoothing in inertial navigation systems. – Proceedings of the International Conference on Information Fusion (FUSION), 2154–2161.

JULIER, S. J. & UHLMANN, J. K., 1997: A non-divergent estimation algorithm in the presence of unknown correlations. – Proceedings of the American Control Conference (ACC), 2369–2373.

JULIER, S. J. & UHLMANN, J. K., 2007: Using covariance intersection for SLAM. – Robotics and Autonomous Systems **55** (1): 3–20.

KAESS, M., JOHANNSSON, H., ROBERTS, R., ILA, V., LEONARD, J. & DELLAERT, F., 2012: iSAM2: Incremental smoothing and mapping using the Bayes tree. – International Journal of Robotics Research **31** (2): 216–235.

KUBELKA, V., OSWALD, L., POMERLEAU, F., COLAS, F., SVOBODA, T. & REINSTEIN, M., 2015: Robust data fusion of multimodal sensory information for mobile robots. – Journal of Field Robotics **32** (4): 447–473.

KÜMMERLE, R., 2013: State Estimation and Optimization for Mobile Robot Navigation, Ph.D. dissertation. University of Freiburg.

KÜMMERLE, R., GRISETTI, G., STRASDAT, H., KONOLIGE, K. & BURGARD, W., 2011: g2o: A general framework for graph optimization. – Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 3607–3613.

LYNEN, S., ACHTELIK, M. W., WEISS, S., CHLI, M. & SIEGWART, R., 2013: A robust and modular multi-sensor fusion approach applied to MAV navigation. – Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 3923–3929.

MERFELS, C. & STACHNISS, C., 2016: Pose fusion with chain pose graphs for automated driving. – Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 3116–3123.

MERFELS, C., RIEMENSCHNEIDER, T. & STACHNISS, C., 2016: Pose fusion with biased and dependent data for automated driving. – Proceedings of the Positioning and Navigation for Intelligent Transportation Systems Conference (POSNAV): ISSN: 2191-8287.

OLSON, E., LEONARD, J. & TELLER, S. J., 2006: Fast iterative alignment of pose graphs with poor initial estimates. – Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2262–2269.

REINHARDT, M., NOACK, B. & HANEBECK, U. D., 2012: Closed-form optimization of covariance intersection for low-dimensional matrices. – Proceedings of the International Conference on Information Fusion (FUSION). IEEE, 1891–1896.

SCHLEGEL, S., KORN, N. & SCHEUERMANN, G., 2012: On the interpolation of data with normally distributed uncertainty for visualization. – IEEE Transactions on Visualization and Computer Graphics

18 (12): 2305–2314.

SIBLEY, G., MATTHIES, L. & SUKHATME, G., 2010: Sliding window filter with application to planetary landing. – Journal of Field Robotics **27** (5): 587–608.

STRASDAT, H., MONTIEL, J. & DAVISON, A. J., 2012: Visual SLAM: Why filter?. – Image and Vision Computing **30** (2): 65–77.

WEISS, S., ACHTELIK, M. W., CHLI, M. & SIEGWART, R., 2012: Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV. – Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 31–38.

Addresses of the Authors:

CHRISTIAN MERFELS, University of Bonn, Institute of Geodesy and Geoinformation, D-53115 Bonn, e-mail: cmerfels@cmerfels.de

Prof. Dr. CYRILL STACHNISS, University of Bonn, Institute of Geodesy and Geoinformation, D-53155 Bonn, e-mail: cyrill.stachniss@igg.uni-bonn.de

Manuskript eingereicht: January 2017
Angenommen: –