MCSLAM : a Multiple Constrained SLAM

Datta Ramadasan[1]
datta.ramadasan@gmail.com
Marc Chevaldonné[2]
marc.chevaldonne@udamail.fr
Thierry Chateau[1]
thierry.chateau@univ-bpclermont.fr

[1] Pascal Institute
Blaise Pascal University
Clermont-Ferrand, FR

[2] ISIT
Auvergne University
Clermont-Ferrand, FR

Abstract

The real-time localization of a camera in an unknown or partially known environment is a problem addressed by Structure From Motion algorithms and more particularly CSLAM algorithms (Constrained Simultaneous Localization And Mapping). In this paper, we propose a new algorithm, named MCSLAM (Multiple Constrained SLAM), designed to dynamically adapt each optimization to the variable number of parameters families and heterogeneous constraints. An automatic method is used to generate a dedicated optimization algorithm, from an exhaustive list of constraints. To our knowledge, this is the only implementation that combines flexibility and performance. Known objects are used to constrain the 3D structure of the reconstruction and a continuous-time representation of the trajectory is used to deal with motion constraints. A continuous trajectory provides a simple way to add heterogeneous constraints into the optimization framework like other unsynchronised sensors or an evolution model. Several experiments show the effectiveness of our approach in terms of accuracy and execution time compared to the state of the art on several public benchmarks of varying complexity.

1 Introduction

Monocular visual SLAM algorithms compute at the same time the 3D pose of the camera and features from the environment using a sequence of temporally ordered images. It is very useful for applications like mobile robot automatic guidance or augmented reality. Constrained SLAM (CSLAM) have been introduced to inject prior like known 3D object models or motion constraints within the optimisation process (see figure 1). Although many CSLAM contributions have been published in the last decade, there is no real-time implementation that deals dynamically with both a variable number of parameters families (sensor pose, roughly known objects poses and dimensions, delay between sensors, ...) and heterogeneous constraints (reprojection error, distance from points or edges to object surface, acceleration...).

We propose a new algorithm, named MCSLAM (Multiple Constrained SLAM), designed to dynamically adapt each optimization to the variable number of parameters families and heterogeneous constraints. This algorithm is based on three contributions: 1) a new Levenberg-Marquardt optimizer C++ library named LMA 2) an architecture allowing a high level of flexibility and performances 3) a real-time usage of a temporal spline curve as the

Pages 107.1-107.12
DOI: https://dx.doi.org/10.5244/C.29.107

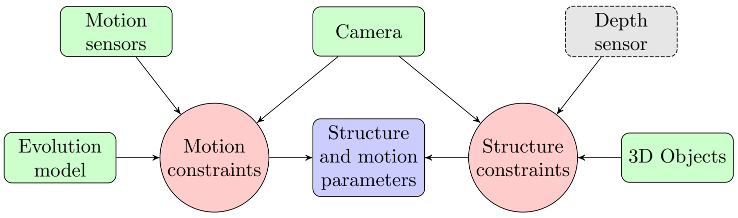


Figure 1: The presented MCSLAM algorithm is designed to be easily customizable. It manages different configurations involving different constraints. For example, a depth sensor could easily be added in the process.

parametric trajectory model that provides an efficient way to add heterogeneous constraints within the optimization. LMA is based on a compile-time algorithm designed to generate a mean least-square solver specialized to a given problem. Using meta-programming technics, LMA offers better run-time performances using a similar API to the state of the art optimization libraries. The design of the MCSLAM allows to easily add or remove some constraints coming from heterogeneous observations on the system motion or the 3D structure of the environment. This design allows us to include a spline curve constraint adapted from the Lovegrove *et .al* work [23] to be usable into a real-time key-frame based SLAM.

Section 2 relates the works on bundle adjustment libraries and CSLAM. Section 3 is a presentation of the proposed LMA library. Then the proposed MCSLAM algorithm is detailed in section 4. Optimization results are shown through a benchmark in 5.1 and MC-SLAM is evaluated in terms of precision on real data in 5.2.

2 Related Work

This section reports recent works published around visual CSLAM algorithm and open-source available libraries used to solve bundle adjustment (BA) problems. SLAM algorithms are generally classified in two main categories: probabilistic methods based on a recursive Bayesian filter [6], and optimization methods based on bundle adjustment [34]. In this paper, we focus on sparse bundle adjustment (BA) methods that offer a good trade-off between accuracy and execution time as shown by Strasdat *et al.* [31].

2.1 Optimization libraries for bundle adjustment

BA is an algorithm based on the minimization of a cost function corresponding, most of the time, to the reprojection error. Lourakis *et al.* [22] present a tutorial about BA applied to camera poses and 3D points. They detailed the internal structure of the hessian matrix (approximated by $J^T J$) and the normal equations resolution using the problem sparsity. They show that it's possible to avoid every zero value and see the hessian like a set of little block matrices in order to save memory space and avoid useless computations. They also detail how to use the Schur complement trick to reduce the complexity of little to medium size problem. However, the Schur complement is the result of a product matrix by matrix which is very time consuming on big size problems. To solve some problems involving thousands of camera and millions of 3D features in the benchmark [2], Agarwal *et al.* exploit the inexact Newton method, whose convergence has been demonstrated by Wright *et al.* [36]. This method solves approximatively the normal equations using only few iterations of the preconditioned conjugated gradient (PCG). And the Schur complement is evaluated implicitly using only matrix by vector product.

Several open-source efficient implementations already exist. **sba**, developed by Lourakis (2004) is one of the first popular library for solving BA problem. **g2o** is a C++ framework develop by Kummerle *et al.* [16] to optimize graph-based non-linear error functions and that uses a sparse structure. **pba** [37] is the most optimized and specialized library for large BA problem (exploit parallel computer architecture using SIMD, multi-thread and GPU). **Ceres** [1] is the most advanced open source library for general non least squares optimization based on highly optimized sparse library, supported by Google.

The optimization algorithm must fit dynamically to a variable constrained configuration. The two state of the art libraries usable to solve this optimization problem are Ceres[1] and g2o [2]. They are able to manage many kind of constraints but they need the data structure of the problem to be rewritten to fit a certain format. Moreover, the genericity of these frameworks is based on the inheritance of the C++ language which negatively impacts the performances. To overcome this limitation, we propose a new optimization library named LMA, opensource, written in C++, and based on the automatic generation of code at compile-time. LMA generates a specialized code to solve a non-linear least squares problem of Levenberg-Marquardt type.

2.2 Constrained-SLAM

Usually, a SLAM algorithm [13, 26] minimizes the reprojection error corresponding to the distance in pixels between the 3D point projected in a camera frame and the observed feature. Some approaches use 3D edges instead of 3D points [8] in slightly textured environments, or in complement with 3D points [14] to improve the robustness of the localization system during rapid camera motion. Recently, Mur-Artal *et al.* [27] propose a robust real-time SLAM framework evaluated on many datasets. They improve the accuracy of classical SLAM using ORB descriptor and by defining additional criteria on the 3D points tracking, key-frames selections, and optimized parameters selections. However, monocular methods will always suffer from the scale factor drift and the error accumulation. To tackle these limitations, constrained SLAM (CSLAM) methods based on monocular vision use some additional informations (known or partially known object, evolution model, additional sensors, ...) to constrain the reconstruction and to increase the accuracy of the localization. Two categories of constraints can be used to improve the accuracy of SLAM reconstructions:1) motion constraints (evolution model, spline, odometer, GPS, IMU, ...) and 2) structure constraints (3D object, 3D points cloud, depth sensors, ...).

Many works propose to include inertial measurement unit (IMU) and GPS data to constrain the bundle adjustment [9, 11, 15, 17, 19, 21, 23, 25]. Whereas Lhuillier [21] proposes a new minimization algorithm to deal with heterogeneous measurements, Kume and Michot [15, 25] minimize a weighted sum of sensors measurements and reprojection errors. In [19], Leutenegger *et al.* use an IMU to constrain the poses of a key-frame based SLAM. This approach linearly interpolates the key-frame poses at the time of the IMU data to apply a motion constraint. A similar problem is tackle by Furgale *et al.* in [9] using a continuous-time

[1]http://ceres-solver.org/

[2]https://github.com/RainerKuemmerle/g2o

representation of the trajectory to avoid data interpolation. Using a b-spline parametrization of the sensors motion, they demonstrate the possibility of estimating the rigid transformation between camera and IMU, and the IMU bias. The same problem is solved by Lovegrove *et al.* in [23] using a cumulative b-spline to define the trajectory of a rolling shutter camera and an IMU. Whereas Furgale *et al.* represent the motion by a spline of position and a spline of rotation (using Cayley-Gibbs-Rodrigues formulation), Lovegrove *et al.* use the Lie Algebra $\mathfrak{se}3$ of the matrix group $\mathbb{SE}3$ to parametrize both position and rotation with one spline.

Concerning structure constraints, Rodriguez *et al.* [29] use a structureless approach (without 3D points) by constraining the camera motion with the epipolar geometry. For augmented reality applications, Tamaazousti *et al.* [33] use observations provided by an accurate model of the observed object. These observations are added into the BA introduced in [26] to constrain the 3D points of the object. Even if the result of the method is very accurate, it doesn't fit with an application involving many coarsely initialized and modeled objects because the pose and the shape of an object aren't optimised during the CSLAM. Similarly to our approach, Galvez *et al.* [10] optimized the parameters of 3D objects detected in real-time in the environment. Their objects detection is based on a exhaustive bags of words dictionary and use the ORB detector. Larnaout *et al.* [18] use some constraints from a terrain elevation model and an approximative city model to constrain the reconstruction in urban environment. The terrain elevation model applies a constraint on the altitude of the trajectory, and the city model applies a constraint on the reconstructed 3D points. Other works involving objects use multi-view stereo system [3] or depth sensors [4, 30, 32, 35] to reconstruct 3D shapes. In [6], Dame *et al.* use a dense SLAM to build a depth-map using 3D shape priors. However, this solution requires a lot of extra computations compared to sparse approaches.

3 LMA: A new implementation of the Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm [20][24] is a very popular non-linear least squares method to solve computer vision problems. The main idea of our implementation of this algorithm is to provide a simple interface with a non-intrusive mechanism of adaptation to a problem while maintaining good performances. Instead of adapting a problem to the specificities of a solver, LMA adapts itself to the problem. LMA works as a meta-program (a program writing another) using C++ template to analyse at compile-time (CT) the problem to optimize from a list of C++ functors (a functor represents a constraint in C++ language). The parameters are deduced from the functors arguments and the degree of freedom (*dof*) of each parameter is defined by the user. Then LMA generates a data structure to store the functors and the parameters in heterogeneous container (also known as tuple) according to the number of parameters families and constraints. The resolution of the normal equations is written efficiently using a sparse representation constructed with a set of small matrices of static sizes (*ie.* known at CT). The size of each small matrix depends of the *dof.* of the corresponding parameters. Storing matrices of different static sizes is possible using tuple. In [28], Polok *et al.* show that implementing the normal equations using only matrices of static sizes offers better runtime performance for least squares problems compared to other sparse approaches used in Ceres. Using tuple, LMA never needs inheritance and polymorphism which allows the compiler to generate an efficient program. Moreover, LMA detects at CT the cross derivatives which are never used by analysing the link between parameters involved in every functor. In a same way, it detects at CT if a part of the hessian is symmetric diagonal in order to use adequate storage and inversion strategy. The LMA library solves the normal equations using dense Cholesky (from Eigen[3]) or a sparse PCG specially designed to manage little matrices of static size. It also implements the classical optimization tricks (like the Schur Complement, or the Implicit Schur method) to be effective on little, medium, and big size problems. At run time, the solver is filled with the constraints using the observations and the corresponding parameters. The minimization policy is managed by the Levenberg-Marquardt algorithm which updates every problem parameters when the global error decreases. Note that LMA also implements common features as automatic differentiation (which outperforms Ceres *Jet* implementation using template expressions) and robust cost functions. LMA is available freely available on git.univ-bpclermont.fr/datta.ramadasan/lma.

4 MCSLAM

This section presents how we designed the MCSLAM algorithm to add motion and structure constraints. Our approach is fully generic on every parameters and constraints. The continuous-time representation of the trajectory is used to deal with constraints on the motion. This allows to mix data from many unsynchronised sensors and evolution model. To apply constraints on the 3D structure of the environment, 3D models of coarsely knowns shapes are used. This approach has been tested with different constraints: reprojection between poses and 3D features (3D points and edges), 3D distance between the 3D objects and the 3D features, constant velocity model and IMU gyrometer and accelerometer on the trajectory. Each parameters family has its own parametrization. Every orientation (for camera, 3D edges, 3D objects and knots) is parametrized using the exponential map [13]. The functioning of the MCSLAM is based on an exhaustive list of constraints and of parameters organized in a graph of dependencies. Each constraint corresponding are created, associated to some parameters, and removed according to the classical scheme of the incremental reconstruction described by Mouragnon [26]. This is described with more details below.

4.1 Motion constraints : Spline

To represent the motion, we use the uniform cumulative b-spline described by Lovegrove *et al.* [23] but we separate position and orientation in two different splines. We use the Rodriguez formula to compute the exponential and the logarithm of $\mathbb{SO}3$ group on the rotation needed to evaluate the spline, its derivative and second derivative. This provides better runtime performances compared to a generalist exponential or logarithm implementation on the 4×4 matrix of the $\mathbb{SE}3$ group. The evaluation of the spline at time t needs four consecutive knots (controls points) $[p_0, p_1, p_2, p_3]$ with respective time $[t_0, t_1, t_2, t_3]$. Those knots are selected such as $t_1 \leq t < t_2$. The time t is normalized between p_1 and p_2 (note that using uniform b-spline, $(t_2 - t_1)$ is constant): $u = (t - t_1)/(t_2 - t_1)$. Kim *et al.* [12] proposed a cumulative form of the b-spline basis function from the Cox-De Boor formula [7]. Using this representation, the coefficient of the spline for a normalized time u are:

$$\{B_1(u), B_2(u), B_3(u)\} = \{(u^3 - 3u^2 + 3u + 5)/6, (-2u^3 + 3u^2 + 3u + 1)/6, u^3/6\} \quad (1)$$

[3]eigen.tuxfamily.org

The position S, velocity $\dot{S}$ and acceleration $\ddot{S}$ of the spline at the normalized time u are computed using the following formulas:

$$S(u) = p_0 + \sum_{i=0}^{2} (p_{i+1} - p_i) B_{i+1}(u) \tag{2}$$

$$\dot{S}(u) = \sum_{i=0}^{2} (p_{i+1} - p_i) \dot{B}_{i+1}(u) \tag{3}$$

$$\ddot{S}(u) = \sum_{i=0}^{2} (p_{i+1} - p_i) \ddot{B}_{i+1}(u) \tag{4}$$

with $\dot{B}$ and $\ddot{B}$ the trivial first and second derivative of B according to u. All the details about the evaluation of the orientation spline are accessible in [23]. Moreover, we adapt the key-frame based SLAM to deal with the spline: a constraint is added in the solver to minimise the difference (in position and orientation) between each optimized key-frame and the spline at the key-frame time. We also use every inter-key-frame poses computed by the localization process to apply a weak constraint on the spline. A temporal sliding window of 3 seconds is used to select, from the SLAM and the IMU, the more recent data used to constrain the spline. And identically to Lovegrove *et al.*, we use a constant velocity model to apply a physical constraint to the spline: this constraint minimizes the variation of velocity (in position and orientation) between every consecutive optimized knots of the spline.

4.2 Structure constraints : 3D Objects

4.2.1 Create and remove 3D objects

When a 3D object is observed, a constraint between the object and its 3D features is created and added to the optimization process. The position, orientation, scale factor and shape of the new 3D object are coarsely initialized (in a semi-automatic manner)[4] and optimized on-line during the MCSLAM . The initialization consists in fitting, approximatively, the 3D model of an object with a subset of 3D features provided by the reconstruction. During the MCSLAM , the user uses the interface to click a convex hull in the image around the object. Then, the 2D features inside the convex hull are used to initialize the object according to the 2D$-$3D associations. Each class of object (from the exhaustive list of parameters) is tested and the class giving the best fit is kept. When none of the 3D features are associated to the 3D object, it is no longer constrained so it is removed from the optimization process.

4.2.2 3D objects and features association

MCSLAM dynamically integrates, in the optimization process, the constraints coming from the objects partially known from the environment (*i.e* whose geometric model is approximatively known). However, the reconstructed 3D features may correspond either to the unknown part of the environment, or to the partially known objects. To ensure the MCSLAM convergence, a step of association between 3D features and the corresponding objects is required at each optimization. In this work, a 3D feature P is associated to an object π if the distance d_P between P and the nearest surface of the object π is lower than a threshold σ_π.

[4]Classical 2D image detection algorithms could be used to achieved a fully automatic initialization.

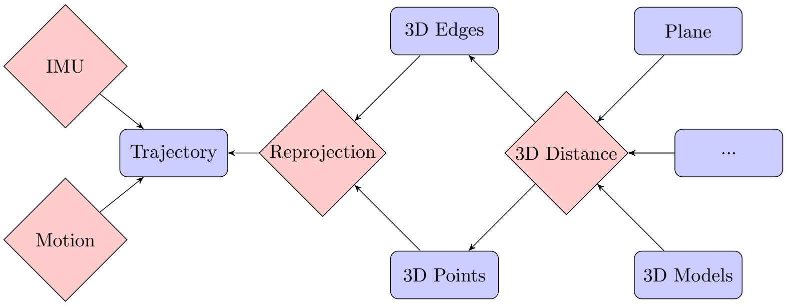


Figure 2: Dependencies graph of MCSLAM .

To choose a value of σ_π invariant to the reconstruction scale factor, the dimension D_π of the object π is used such as $\sigma_\pi = D_\pi \times \lambda$ with λ the intensity of the constraint set to 0.005. If an association is created, the parameters of the 3D features P and the parameters of the corresponding object π are optimized in order to minimize the distance d_P. So, the optimization process combines pixelic errors (reprojection errors) and metric errors (3D distance between features and the objects). The homogenization of these two errors is possible by weighting the 3D errors with the coefficient σ_π.

4.3 MCSLAM implementation

The MCSLAM algorithm is based on the dependencies graph presented by the figure 2. This graph contains a list of constraints, a list of parameters and a list of dependencies. The MCSLAM uses the graph during two important steps. The first step consists of analysing the problem configuration (list of constraints, list of parameters and links between each of them) at compile-time to generate a specified LMA solver, whose cost function to minimize is the sum of the constraints C dynamically added: $\mathcal{E} = \sum_{i=0}^{C} \sum_{k=0}^{K_i} \left\| \frac{\rho_{i,k}}{\sigma_i} \right\|^2$ with K_i the number of observations corresponding to the constraint i, $\rho_{i,k}$ the error associated to the observation k of the constraint i and σ_i the estimation of the measurement error. The σ_i value is specific to each constraint : for 3D objects it's σ_π see section 4.2.2, for the reprojection error it's the feature detector error (2 pixels is used), we use 1 for the gyrometer and 50 for the accelerometer. The second step is the skimming of the graph according to the dependencies to feed the solver with the parameters and the constraints. Each constraint has a list of functors and each functor corresponds to one error term of the cost function to minimize. Moreover, one function has to be written for each constraint: this function has two input, the dependencies graph and the solver. Because the types of those two objects are specific to the graph configuration, this function is written using generic types (C++ template). Thus, each constraint has a full access to the graph and any required data in order to fill the solver. To achieve real-time performances, we extend the incremental reconstruction scheme described by [26] to manage the constraints. It means that the minimization has to deal with constraints involving at the same time optimized (recent parameters) and non-optimized parameters (deprecated parameters). This produces an anchorage of the optimized parameters according to the deprecated parameters. Consequently, the constraints are applied only on parameters corresponding at the end of the reconstruction (spatially and temporally).

5 Experiments

This section presents first an optimization benchmark on a public datasets used in [2]. The aim of this benchmark is to compare LMA, Ceres and g2o in term of performance and accuracy on bundle adjustment problems. Then various configurations of MCSLAM are compared to prove the benefit of using heterogeneous constraints. All the experiments are run on a desktop computer with a processor i7 3.4GHz.

5.1 LMA evaluation

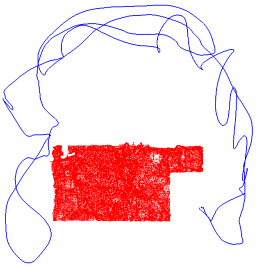
Three comparatives are performed on problems of different sizes using the solvers g2o, Ceres-1.10 and LMA (execution time are summarized in table 3(a)) The first comparative is a basic problem designed to optimize the equation of a circle by minimizing the distance between 2D observations and the circle. The problem is composed of 3 *dof.* and 2000 observations. Each solver iterates 3 times and the computation of the derivatives is numeric and centred. After 3 iterations, the error is the same for the 3 solvers. The computation times are 2.2 ms for g2o, 2.4 ms for Ceres, and 0.51 ms for LMA. In this experiment, LMA is 4.8 times faster than Ceres and 4.3 times faster than g2o. The 3 solvers use the same dense algorithm (from Eigen) to solve the normal equation, so the difference of performance is only made by the absence of inheritance in LMA. The second experiment is performed on an instance of the dataset used in the Ceres benchmark [2]. Each solver is limited to 10 iterations of Levenberg-Marquardt, derivatives are numeric and centred, and the PCG (using the Schur-Jacobi preconditioning [2]) is limited to 20 iterations when it's used. The table 3(a) shows the execution time of each solver on the benchmark instance composed of 16 poses, 22k 3D points and 83k observations. On this instance, LMA is 2 to 3 times faster than Ceres and g2o, with a similar accuracy, using the 4 following algorithms: Sparse (sparse PCG without Schur complement), Dense Schur (dense Cholesky and Schur complement), Sparse Schur (sparse PCG and Schur complement) and Implicit Schur (sparse PCG and implicit Schur complement using inexact Newton method). g2o doesn't implement the sparse resolution without the Schur complement and the implicit method. The third experiment is performed on the 35 instances of the dataset [2] with the Implicit Schur method (using the same configuration as before) and shows that LMA is, on average, 2.5 times faster than Ceres, and more accurate on 21 datasets out of 35. On the biggest dataset(4585 poses and 1.3M of 3D points), LMA takes 131 seconds and is more accurate than Ceres which takes 298 seconds. The convergence rate of LMA is equal to g2o because both use the same damping heuristic to update the lambda parameter of Levenberg-Marquardt. Ceres has a different convergence rate because of the trust region policy which is better for complexes cost functions.

5.2 MCSLAM evaluation

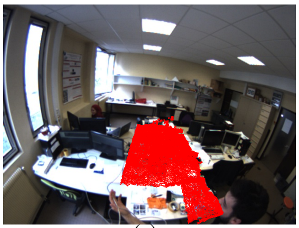
For these experiments, a global shutter video camera running at 60 *fps* is used with a resolution of 640×480. The IMU is a LandMark 40 AHRS running at 100 hertz. The aim of this experiments is to evaluate the impact of the motion constraints on the accuracy of the spline reconstruction. Four different configurations of the MCSLAM are tested : 1) SLAM with a constant velocity model constraint, 2) SLAM with a constant velocity model and IMU constraints, 3) SLAM with a constant velocity model and IMU constraints and IMU's bias optimization, 4) SLAM with spline, constant velocity model, IMU, IMU's bias optimization, and 3D object constraints. The ground truth is an indoor trajectory computed from a

Circle (2 *dof.*)	g2o	Ceres	LMA
Dense	2.26	2.49	**0.51**
BAL$_1$(66k *dof.*)	g2o	Ceres	LMA
Sparse	x	2.63	**1.06**
Dense Schur	3.60	2.36	**1.01**
Sparse Schur	3.49	2.42	**1.10**
Implicit Schur	x	2.97	**1.12**
BAL$_2$ (4M *dof.*)	g2o	Ceres	LMA
Implicit Schur	x	298	**131**

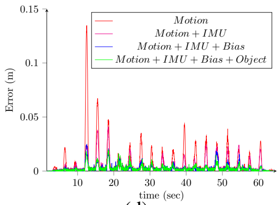
(a)



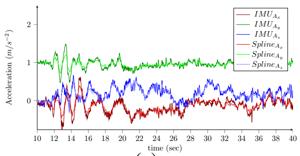
(b)



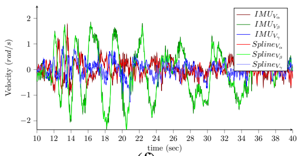
(c)



(d)



(e)



(f)

Figure 3: (a) LMA compared to Ceres and g2o (results are in seconds). (b) Ground truth of the second experiment. (c) Image from the sequence of the second experiment. (d) Error in position of the SLAM using different configuration of constraints. (e,f) IMU's accelerometer and gyrometer compared to the spline acceleration and orientation velocity.

structure from motion algorithm. An approximative edge model (which is generated using the Acute3D software) of a real object of the environment is used to apply the structure constraints. The orientation and position parameters of the 3D objects are included in the optimization with others parameters. The real trajectory begins and ends at the exact same point. This sequence is composed by 3.6k images and contains slow and fast motions. The ground truth is visible on the figure 3(b) in blue, and the edge model in red and an image from the camera is shown on the figure 3(c). Each MCSLAM configuration is run in real-time on the sequence and compared to the ground truth. Moreover, to highlight the robustness of the system, we artificially stop the camera poses constraints on the the spline during 1 second every 2 seconds. During this period, the spline is only constrained by the evolution model and the IMU. The errors in position are shown in the figure 3(d). We observe that the reconstruction is getting better when the process uses more constraints. For those experiments, the reconstruction process takes approximatively 0.1 second and is executed in parallel of the localization process. The localization process is based on vision and takes 1 ms. Figures 3(e) and 3(f) shows the acceleration and the orientation velocity of the SLAM trajectory and the IMU data using all constraints (motion model, IMU with bias estimation and 3D object). This experiment show the importance of using many heterogeneous constraints with the MCSLAM to increase the accuracy and the robustness of the reconstruction.

6 Conclusion

The proposed MCSLAM approach allows an easy implementation of SLAM problem using a variable number of parameters families and constraints. The real-time implementation of MCSLAM is based on the LMA optimization library whose performances outperform the state of the art alternative especially on BA problem. Thanks to the generic management of the constraints, the method can fit many kind of scenario, involving different classes of 3D objects and constraints on a continuous-time trajectory. The approach allows the simultaneous usage of motion constraints, geometric constraints and reprojection constraints on 3D points and edges. Using an important number of constraints of various kind brings a good accuracy and stability to the reconstruction process. The application of the MCSLAM to augmented reality using many classes of objects and sensors fusion has illustrated the accuracy and the performances of the method.

Acknowledgements

This work is partly funded by the European Union. Europe is engaged in Auvergne with the European regional development fund.

References

[1] Sameer Agarwal and Keir Mierle. Ceres solver: Tutorial & reference. *Google Inc*, 2, 2012.

[2] Sameer Agarwal, Noah Snavely, Steven M Seitz, and Richard Szeliski. Bundle adjustment in the large. In *Computer Vision–ECCV 2010*, pages 29–42. Springer, 2010.

[3] Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. Dense object reconstruction with semantic priors. In *CVPR*, pages 1264–1271. IEEE, 2013.

[4] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[5] Amaury Dame, Victor A Prisacariu, Carl Y Ren, and Ian Reid. Dense reconstruction using 3d object shape priors. In *CVPR*, pages 1288–1295. IEEE, 2013.

[6] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[7] Carl De Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6(1): 50–62, 1972.

[8] Ethan Eade and Tom Drummond. Edge landmarks in monocular slam. In *BMVC*, pages 7–16, 2006.

[9] Paul Furgale, Timothy D Barfoot, and Gabe Sibley. Continuous-time batch estimation using temporal basis functions. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2088–2095. IEEE, 2012.

[10] Dorian Gálvez-López, Marta Salas, Juan D Tardós, and JMM Montiel. Real-time monocular object slam. *arXiv preprint arXiv:1504.02398*, 2015.

[11] Vadim Indelman, Stephen Williams, Michael Kaess, and Frank Dellaert. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61(8):721–738, 2013.

[12] Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 369–376. ACM, 1995.

[13] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[14] Georg Klein and David Murray. Improving the agility of keyframe-based slam. In *Computer Vision–ECCV 2008*, pages 802–815. Springer, 2008.

[15] Hideyuki Kume, Takafumi Taketomi, Tomokazu Sato, and Naokazu Yokoya. Extrinsic camera parameter estimation using video images and gps considering gps positioning accuracy. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3923–3926. IEEE, 2010.

[16] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[17] Dorra Larnaout, Vincent Gay-Bellile, Steve Bourgeois, and Michel Dhome. Vehicle 6-dof localization based on slam constrained by gps and digital elevation model information. In *ICIP*, pages 2504–2508, 2013.

[18] Dorra Larnaout, Vincent Gay-Bellile, Steve Bourgeois, Benjamin Labbé, and Michel Dhome. Fast and automatic city-scale environment modeling for an accurate 6dof vehicle localization. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 265–266. IEEE, 2013.

[19] Stefan Leutenegger, Paul Timothy Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart. Keyframe-based visual-inertial slam using nonlinear optimization. In *Robotics: Science and Systems*, 2013.

[20] Kenneth Levenberg. A method for the solution of certain problems in least squares. *Quarterly of applied mathematics*, 2:164–168, 1944.

[21] Maxime Lhuillier. Incremental fusion of structure-from-motion and gps using constrained bundle adjustments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(12):2489–2495, 2012.

[22] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009. doi: http://doi.acm.org/10.1145/1486525.1486527.

[23] Steven Lovegrove, Alonso Patron-Perez, and Gabe Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *Proceedings of the British machine vision conference*, pages 93–1, 2013.

[24] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[25] Julien Michot, Adrien Bartoli, and Francois Gaspard. Bi-objective bundle adjustment with application to multi-sensor slam. *3DPVT10*, 3025, 2010.

[26] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 363–370. IEEE, 2006.

[27] Raul Mur-Artal, JMM Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *arXiv preprint arXiv:1502.00956*, 2015.

[28] Lukáš Polok, S Viorela Ila, and Pavel Smrž. Cache efficient implementation for block matrix operations. In *Proceedings of the 21st High Performance Computing Symposia*. Association for Computing Machinery, 2013.

[29] Antonio L Rodriguez, Pedro E Lopez-de Teruel, and Alberto Ruiz. Gea optimization for live structureless motion estimation. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 715–718. IEEE, 2011.

[30] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: simultaneous localisation and mapping at the level of objects. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1352–1359. IEEE, 2013.

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

[32] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5182–5189. IEEE, 2013.

[33] Mohamed Tamaazousti, Vincent Gay-Bellile, SN Collette, Steve Bourgeois, and Michel Dhome. Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3073–3080. IEEE, 2011.

[34] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment, a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[35] Tommi Tykkala, Andrew I Comport, and Joni-Kristian Kamarainen. Photorealistic 3d mapping of indoors by rgb-d scanning process. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1050–1055. IEEE, 2013.

[36] SJ Wright and John Norman Holt. An inexact levenberg-marquardt method for large sparse nonlinear least squres. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 26(04):387–403, 1985.

[37] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064. IEEE, 2011.