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

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 (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...). The proposed algorithm is based on three contributions: 1) a new Levenberg-Marquardt C++ library named LMA and freely available [1] 2) an architecture allowing a high level of flexibility and performances 3) a real-time usage of a temporal spline curve as the parametric trajectory model that provides an efficient way to add heterogeneous constraints within the optimization.

The main idea of LMA is to provide a simple interface with a non-intrusive mechanism of adaptation to a problem while maintaining good performances. LMA works as a meta-program using C++ template to analyse at compile-time (CT) the problem to optimize from a list of C++ functors. 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 *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. The LMA library solves the normal equations using dense Cholesky or a sparse PCG specially designed to manage little matrices of static size. It also implements the classical optimization tricks to be effective on little, medium, and big size problems. LMA also implements common features as automatic differentiation and robust cost functions.

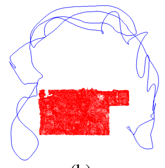
The continuous-time representation of the trajectory is used to deal with constraints on the motion. This allows to mix data from many unsynchronised sensors and evolution model. To apply constraints on the 3D structure of the environment, 3D models of coarsely knowns shapes are used. To represent the motion, we use the uniform cumulative b-spline described by Lovegrove *et al.* [2] but we separate position and orientation in two different splines and we use the Rodriguez formula to compute the exponential and the logarithm of $\mathbb{SO}3$ group. Moreover, we adapt the key-frame based SLAM to deal with the spline: key-frames are constrained to be on the spline. 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.

The MCSLAM algorithm is based on a graph composed by constraints, parameters and dependencies. First, the problem configuration is analysed 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. Then the graph is skimmed 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. 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 [3] to manage the constraints. 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).

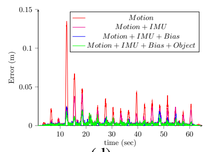
[1] git.univ-bpclermont.fr/datta.ramadasan/lma

Circle (2 *dof.*)	g2o	Ceres	LMA
Dense	2.26	2.49	**0.51**
$\mathbf{BAL_1}$(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**
$\mathbf{BAL_2}$ (4M *dof.*)	g2o	Ceres	LMA
Implicit Schur	x	298	**131**

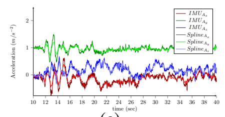
(a)


(b)

(c)



(d)


(e)

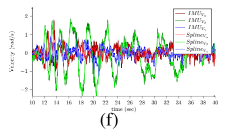

(f)

Figure 1: (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.

Three comparatives summarized in table 1(a) are performed on problems of different sizes using the solvers g2o, Ceres-1.10 and LMA. The first comparative is a basic problem designed to optimize the equation of a circle. In this experiment, LMA is 4.8 times faster than Ceres and 4.3 times faster than g2o. The second experiment is performed on an instance of the dataset used in the Ceres benchmark [1]. Each solver is limited to 10 iterations of Levenberg-Marquardt, derivatives are numeric and centred, and the PCG is limited to 20 iterations when it's used. On the first instance, LMA is 2 to 3 times faster than Ceres and g2o, with a similar accuracy, using the 4 different linear solvers. The third experiment is performed on the 35 instances of the dataset [1] with the Implicit Schur method and shows that LMA is, on average, 2.5 times faster than Ceres, and more accurate on 21 datasets out of 35.

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 structure from motion algorithm. The ground truth is visible on the figure 1(b) in blue, and the edge model in red and an image from the camera is shown on the figure 1(c). 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. The errors in position are shown in the figure 1(d). Figures 1(e) and 1(f) shows the acceleration and the orientation velocity of the trajectory and the IMU data using all constraints.

[1] 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.

[2] 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.

[3] 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.