Fusion of Inertial and Visual Measurements for RGB-D SLAM on Mobile Devices

Nicholas Brunetto Samuele Salti Nicola Fioraio Tommaso Cavallari
Luigi Di Stefano
Department of Computer Science and Engineering
University of Bologna, Bologna, Italy
nicholas.brunetto@studio.unibo.it {name.surname}@unibo.it

Abstract

Simultaneous Localization and Mapping (SLAM) algorithms have been recently deployed on mobile devices, where they can enable a broad range of novel applications. Nevertheless, pure visual SLAM is inherently weak at operating in environments with a reduced number of visual features. Indeed, even many recent proposals based on RGB-D sensors cannot handle properly such scenarios, as several steps of the algorithms are based on matching visual features. In this work we propose a framework suitable for mobile platforms to fuse pose estimations attained from visual and inertial measurements, with the aim of extending the range of scenarios addressable by mobile visual SLAM. The framework deploys an array of Kalman filters where the careful selection of the state variables and the preprocessing of the inertial sensor measurements result in a simple and effective data fusion process. We present qualitative and quantitative experiments to show the improved SLAM performance delivered by the proposed approach.

1. Introduction

The problem of Simultaneous Localization And Mapping (SLAM) has been traditionally addressed by either expensive 3D sensors, *e.g*. laser scanners, or monocular RGB cameras. The former have enabled impressive results thanks to their high-quality measurements [4], although size and expensiveness limit the breadth of their applications. On the other hand, monocular SLAM has reached a considerable maturity [7, 14] but still mandates massive parallelization by GPGPU to attain dense 3D reconstruction [18]. In the past few years, consumer-grade RGB-D cameras capable of delivering colour and depth information in real-time, *e.g*. the Microsoft Kinect and Asus Xtion Pro Live, have fostered new approaches to tackle the SLAM problem [17, 11, 9].

Nowadays, novel devices aimed at enhancing mobile platforms with RGB-D sensing capabilities are starting to become available, examples including the Structure sensor, Google's Project Tango and the Intel RealSense 3D Camera. However, most RGB-D SLAM systems are unsuited to mobile platforms due to their reliance on massive GPGPU processing to pursue real-time or even interactive operation on desktop PCs [17, 9].

This is not the case, though, of the recently proposed SlamDunk algorithm [10], which features few threads, low memory consumption and, peculiarly, can achieve real-time RGB-D SLAM on a desktop PC without GPGPU acceleration. Due to its favorable computational traits as well as its robustness and speed, SlamDunk was then adopted as reference architecture for the creation of the first RGB-D SLAM framework amenable to mobile platforms [5]. The results reported in [5] show the mobile algorithm to attain an accuracy comparable to the desktop version as well as an approximate speed of about 6-7 frames per second on an off-the-shelf Android tablet.

In spite of the adoption of RGB-D sensing, a major failure case for most visual SLAM algorithms, including SlamDunk, concerns the exploration of environments that are poor in texture, such that image features are scarce. Indeed, all RGB-D SLAM systems but KinectFusion [17] and its variants [6] rely on detection and matching 2D visual features in order to accomplish key tasks such as camera tracking. On the other end, even low-end mobile devices are nowadays equipped with inertial sensors, the most common being accelerometers and gyroscopes. Such sensors allow for estimating the pose of the device with respect to its initial position and may therefore be deployed alongside visual measurements to possibly ameliorate camera tracking. The inertial data suffer of peculiar shortcomings, though, as they are typically noisier than visual measurements and subject to drift over time.

In this paper we propose a framework based on Kalman

filtering to integrate inertial measurements into an RGB-D SLAM system. Our framework is purposely simple to allow for efficient implementation on resource-limited devices. As the sampling rate of inertial sensors is significantly higher than the frame rate of the RGB-D camera, the aim of our Kalman filtering is twofold: on one hand, it smooths out noise as long as only inertial pose measurements are gathered; on the other, it realizes the actual data fusion whenever both inertial and visual SLAM pose estimations are available.

The remainder of this paper is organized as follows. The next section reviews some relevant publications on RGB-D SLAM systems and the fusion of inertial and visual data for SLAM/Visual Odometry. Then, in Sec. 3, we summarize the SlamDunk pipeline, which we used to develop and test our framework to fuse inertial measurements and RGB-D SLAM. The proposed framework is then presented in Sec. 4 and experimental findings reported in Sec. 5. Finally, in Sec. 6, we draw concluding remarks and highlight the major issues to be addressed by future work.

2. Related Work

Most state-of-the-art RGB-D SLAM systems produce high-quality results through acceleration by massive GPGPU processing on high-end desktop platforms [17, 24]. As such, these systems can hardly provide design guidelines towards rendering interactive SLAM feasible on platforms with limited resources such as mobile devices. One of the first RGB-D SLAM proposals was RGB-D Mapping [11], which, indeed, does not rely on GPGPU acceleration. They attain camera tracking by pairwise matching of image features and carry out global pose graph optimization to handle camera drift. To constrain nodes and make the optimization effective, the algorithm looks for possible loop closures by matching image features within a subset of previous keyframes and perform global pose graph relaxation accordingly. Therefore, as the explored space gets wider and the graph size increases, more time is spent in finding loop closures and optimizing poses. A similar approach is deployed by RGB-D SLAM [9], which, moreover, can obtain near real-time processing (less than 15 FPS) by deploying GPGPU processing to compute SIFT visual features. Nonetheless, the speed usually drops after gathering many frames due to the increasing complexity of the global optimization routine. SlamDunk [10], presented in more details in Sec. 3, is a recent proposal whereby some major issues inherent to the discussed works can be dealt with effectively.

As a negative effect, SLAM algorithms deploying image features may exhibit a significant performance deterioration (or even fail) whenever features are scarce and/or less reliable, *e.g.* while exploring low-textured portions of the workspace or processing blurred images due to fast camera movements. However, these issues may be tackled more easily in the mobile realm thanks to the inertial sensors available nowadays inside the majority of the devices. Accordingly, the work by Tanskanen *et al.* [22] uses a standard Extended Kalman Filter to deploy both visual and inertial measurements on a mobile phone. However, the system is designed for object capturing by a monocular camera, with dense 3D reconstruction running at about 0.3-0.5Hz. Differently, our approach leverages on RGB-D sensing to attain dense mapping and can reach much higher frame rates. Another proposal dealing with fusion of visual and inertial measurements is given by the work of Weiss and Siegwart [23], which addresses monocular visual odometry and is mainly aimed at estimating the metric scale. Purposely, the authors propose a complex Extended Kalman Filter formulation which may be fused with any visual odometry engine. Another approach to visual and inertial data fusion is given by the work of Huang *et al.* [12], which makes use of an autonomous micro air vehicle (MAV) equipped with an RGB-D camera and an IMU. The work mainly focuses on the visual algorithm implementation, without showing details on the Extended Kalman Filter used for the fusion. Making a comparison with our approach, their visual algorithm detects and matches FAST features which present less robustness in opposition to the SURF features considered in our work. Moreover, the pose optimization step and other operations are computed separately on a laptop computer, while our approach executes entirely on the mobile platform, reaching slower but still interactive frame rates. Similar to our approach is also the work of Qayyum and Kim [20], which is mainly focused on the usage of a Kinect sensor in outdoor environments. In our work, instead, we decided to focus on indoor scenarios that present a scarcity of visual features.

3. SlamDunk System Overview

Fig. 1 shows the SlamDunk [10] algorithm pipeline, which can be divided into three main modules: Local Mapping, Camera Tracking and Local Optimization. Local Mapping models the camera path as a collection of RGB-D keyframes and stores their poses within a quad-tree data structure. To improve efficiency, the tracking procedure does not consider the full camera path but, instead, a subset of spatially adjacent keyframes is selected from the quad-tree using an *Active Window*. Visual features associated with such keyframes, *e.g.* SURF [2] or SIFT [16], are gathered and stored into a local *Feature Pool*, which is a fast indexing structure enabling efficient feature matching.

The Camera Tracking module is the first to execute, taking the RGB-D frame coming from a generic RGB-D camera as input and returning the estimated camera pose. Visual features are extracted from the RGB image and matched into the *Feature Pool* to find correspondences between the current frame and the keyframes stored in the local map.

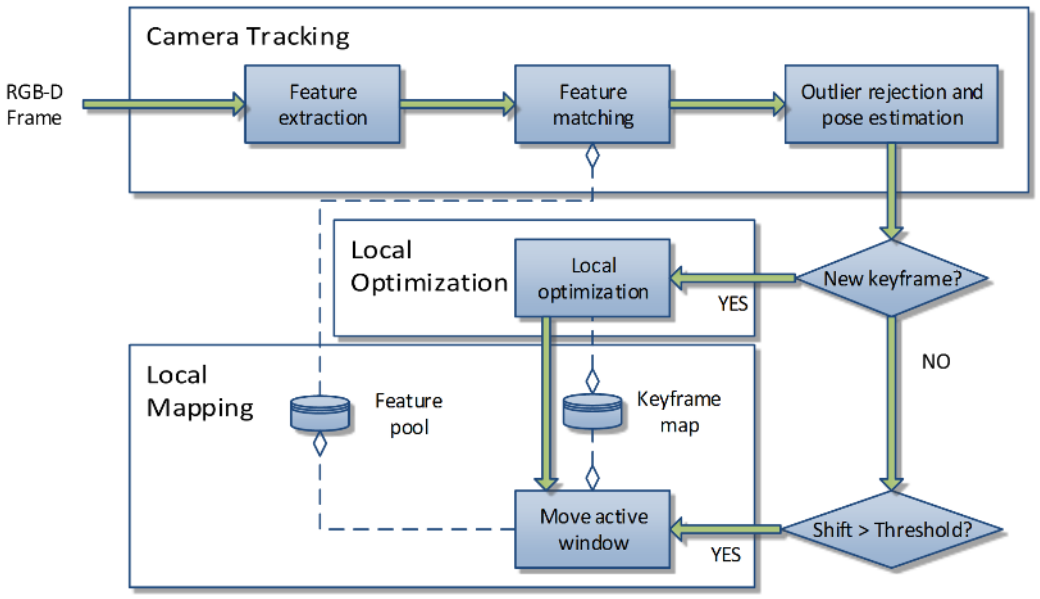

Figure 1. The SlamDunk pipeline comprises three main modules: Camera Tracking, Local Mapping, and Local Optimization.

Using the associated depth measurements, matching pixels are back-projected in the 3D space leading to 3D correspondences. Subsequently, a full 6DOF pose can be robustly estimated by running a standard Absolute Orientation algorithm [1] within a RANSAC framework. Camera pose is represented as a 4×4 matrix in the following format:

$$\mathrm{T} = \begin{pmatrix} \mathrm{R} & t \\ \mathbf{0}_3^\top & 1 \end{pmatrix}, \tag{1}$$

where R represents a 3×3 rotation matrix and t is a translation vector. Points are projected onto the image plane by means of the camera matrix, while the 3D back-projection is computed taking into consideration the associated depth measurement for each pixel, as detailed in [10].

Given the calculated pose, if the currently tracked frame presents a limited overlap with respect to the Local Map, it is promoted as a new keyframe, thereby triggering the Local Optimization module and updating the Local Map itself, which gets centered around the newly spawned keyframe.

The Local Optimization module has the task of optimizing poses across a pose graph associated with the keyframes to minimize the reconstruction error. Further details on the cost function used for this optimization can be found in [10].

A mobile implementation of the algorithm is proposed in [5], in order to cope with the issues related to the reduced computational resources available in mobile devices compared to the typical desktop PC environment. In particular, the feature detection and extraction algorithms used by SlamDunk, *i.e*. SIFT [16] or SURF [2], would have been computationally too expensive for a mobile environment. Hence, to minimize loss of accuracy and maintain an acceptable speed, an optimized Upright-SURF detector and the BRISK descriptor [15] were deployed. Moreover, a new software architecture was developed around the algorithm pipeline, thus allowing for faster communication between the different modules and avoiding bottlenecks in the system.

4. Inertial Data Fusion

The integration of inertial measurements alongside visual measurements sets forth several challenges. First, the sensors available inside common mobile devices present considerable measurement noise. In this work, we use only the most common inertial sensors that can provide a full pose estimation, *i.e*. the gyroscope, to measure angular velocity and integrate it to estimate rotation angles, and the accelerometer, to measure linear acceleration and double integrate it to estimate translations. The magnetometer could also be employed to estimate rotations but it is subject to severe disturbances in the presence of sources of electromagnetic fields and thus we opted not to use it. The visual measurements, too, present some degree of noise that should be considered when fusing pose estimates. A solution that is computationally not expensive and takes into account the errors in estimating the pose from different sensors is offered by the Kalman filter [13].

Fig. 2 shows the new module included into the SlamDunk pipeline, which consists of two Kalman filters. Given measurements coming from the tracking module and the inertial sensors, one filter is used to estimate camera orientation and the other filter to estimate camera position. We add the new module before the keyframe check and the optional optimization step to let the data fusion beneficially influ-

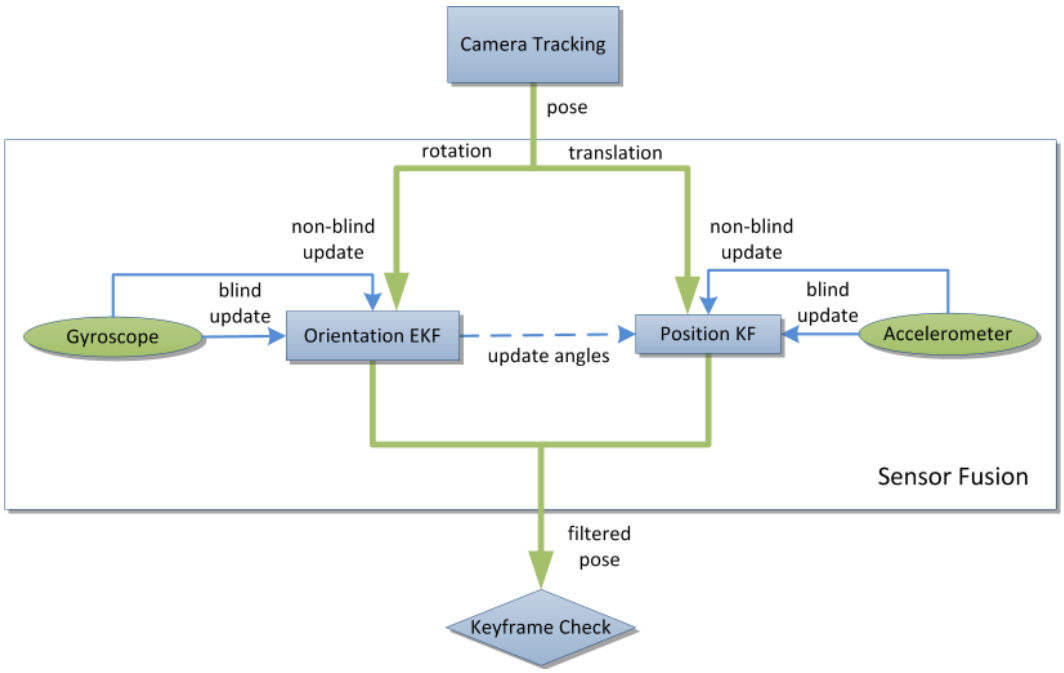

Figure 2. Schematic representation of the sensor fusion module and its connections to the SlamDunk modules.

ence both the camera tracking, by considering its output as the device pose for the current frame, as well as the local optimization, where the filtered pose provides a better initial guess. The subdivision of orientation and position filtering allows us to test two different versions of the module, one with only the orientation estimation and another that also considers position estimation.

A second challenge is represented by the different acquisition rates. In our scenario the accelerometer runs at about 200 Hz, the gyroscope at about 100 Hz, whereas the visual tracker is able to produce a pose estimation every 6 frames on average. We decided to solve this problem by using the same Kalman filters in two different operational modes: a *blind* update mode, where only the measurement coming from the corresponding inertial sensor is used, and a *non-blind* update mode, where both inertial and visual measurements are used. The blind update is executed at the same frequency as the corresponding sensor and is useful to let the internal state of the filter evolve at the correct frame rate and, at the same time, filter the inherent noise. The non-blind update, instead, runs every time a new pose from the camera tracker is available and accomplishes the proper data fusion. After every non-blind update the results are made available to the subsequent modules. This operation is also helpful in reducing the inherent drift caused by integration over time of inertial measurements. Both update modes are preceded by a predict phase, in which the internal states of the filters are propagated to the new time step according to a selected motion model.

While camera tracking runs, the Kalman filter state freezes to the inertial measurement closest to the frame acquisition time and we buffer the following inertial data without running the associated blind updates. As soon as the pose from visual measurements is available, we run one non-blind update and then let the filter advance to the current time step with a series of blind updates, thus consuming the buffered measurements. During the optimization phase, if present, the Kalman filters will continue updating their internal states thanks to the received inertial measurements.

4.1. Estimation of Measurements Uncertainty

Fusion of the poses provided by visual and inertial sensors within a Kalman filtering framework requires estimation of measurement uncertainties. The measurement errors of the inertial sensors depend primarily on the characteristics and quality of the actual physical devices. As such, we considered the uncertainties associated with inertial measurements as constant values and tuned them heurisitically. Conversely, the quality of the pose obtained from visual data varies with the amount of geometric structures and texture in the current frame. Therefore, we estimate the uncertainty of the poses provided by SlamDunk's camera tracking module dynamically. In particular, we estimate the coherence of the feature matches under the estimated pose according to the following formula

$$r_k^{visual} = \frac{\sum\limits_{(u,v)\in m(I_k,I_j)} s(u,v)\,(P_k\, f_u^k - P_j\, f_v^j))^2}{\sum\limits_{(u,v)\in m(I_k,I_j)} s(u,v)} \quad (2)$$
$$\forall I_j \in AW\ ,$$

where the weighted average runs over the set m of all feature matches (u, v) between the current frame I_k and the keyframes I_j in the Active Window (AW). P_k and P_j represent the frame and keyframe poses while f_u^k and f_v^j denote the 3D projections of feature points u and v. Each term in the sum is weighed by the score provided by the feature matching process, $s(u, v)$, so to give more importance to matches with a higher score.

4.2. Orientation Filtering and Fusion

The state variables of the filter dealing with the orientation (Fig. 2, left side) are represented as follows:

$$\mathbf{x}_k^{or} = \begin{pmatrix} \boldsymbol{\theta}_k \\ \boldsymbol{\omega}_k \end{pmatrix}\ , \quad (3)$$

where $\boldsymbol{\theta}_k$ represents the rotation at time k between the Device Reference Frame (DRF) and the World Reference Frame (WRF), *i.e.* the frame coincident with the device's pose at time 0, and $\boldsymbol{\omega}_k$ represents the angular velocity in the DRF. Both $\boldsymbol{\theta}_k$ and $\boldsymbol{\omega}_k$ are expressed in axis-angle notation.

The predict phase is based on a constant angular velocity model. To compute the estimate of the state for the new time step k, we need to integrate the previous angular velocity over the time interval Δt and then combine it with the previous orientation θ_{k-1}. Since rotations cannot be combined directly in the axis-angle representation, they have to be converted forth and back to the rotation matrix format, which is achieved via matrix exponential and logarithm [3]. This results in non-linear predict equations

$$\boldsymbol{\theta}_{k|k-1} = \log(\exp(\boldsymbol{\theta}_{k-1|k-1}) \cdot \exp(\boldsymbol{\omega}_{k-1|k-1}\,\Delta t)) \quad (4)$$
$$\boldsymbol{\omega}_{k|k-1} = \boldsymbol{\omega}_{k-1|k-1} \quad (5)$$

which we handle by an Extended Kalman Filter (EKF) approach. By numerical differentiation of the equations around $\mathbf{x}_{k-1}^{ori}$ we can calculate the state transition matrix $\mathbf{F}_{k-1}$ and consequently update the estimate error covariance matrix $\mathbf{P}_{k|k-1}$ of the filter by the standard EKF equations.

Thanks to the chosen state representation, the blind update is linear and thus relies on the standard update equations of the Kalman filter. The measurement in the blind update is simply the gyroscope angular velocity $\boldsymbol{\omega}_k^{gyro}$ while the corresponding matrix for the residual calculation is given by $\mathbf{H}^{blind} = [\mathbf{0}_{3\times3}\ \mathbf{I}_{3\times3}]$.

In the non-blind update, instead, the measurement is given by $\mathbf{R}_k^{visual}$, *i.e.* the rotation matrix estimated by the camera tracker, and the angular velocity $\boldsymbol{\omega}_k^{gyro}$ estimated by the gyroscope. In this case, the innovation of the filter, *i.e.* the difference between the predicted state and the measurement for the current frame, cannot be computed directly in the angle-axis representation. This leads to non-linear update equations that slightly deviate from the traditional EKF formulas, as done *e.g.* in [8]. In our case, the innovation in the non-blind case for the first three entries of the state vector becomes

$$\mathbf{y}_k^{ori} = \mathbf{R}_k^{visual} \ominus \mathbf{H}\,\boldsymbol{\theta}_{k|k-1}^{ori} \quad (6)$$
$$= \log\left(\mathbf{R}_k^{visual}\exp(\mathbf{H}\,\boldsymbol{\theta}_{k|k-1}^{ori})^T\right) \quad (7)$$

where $\ominus$ denotes motion composition. The Kalman gain $\mathbf{K}_k$ is then estimated as usual, whereas the updated estimate of the state becomes

$$\boldsymbol{\theta}_{k|k} = \boldsymbol{\theta}_{k|k-1} \oplus \mathbf{K}_k\,\mathbf{y}_k^{ori} \quad (8)$$
$$= \log\left(\exp(\mathbf{K}_k\,\mathbf{y}_k^{ori})\ \exp(\boldsymbol{\theta}_{k|k-1})\right). \quad (9)$$

The measurement noise covariance matrix for the blind update is represented as a 3×3 diagonal matrix with the gyroscope noise values r_k^{gyro} on the diagonal, $\mathbf{R}^{blind} = r^{gyro}\,\mathbf{I}_{3\times3}$. For the non-blind update the matrix is expanded to a 6×6 diagonal matrix, taking into consideration both SlamDunk uncertainty r_k^{visual} (Eq. 2) for the first three entries of the measurement vector and the gyroscope noise r^{gyro} for the others.

4.3. Position Filtering and Fusion

As far as the translation component is concerned, we can use a linear Kalman filter (Fig. 2, right side) with the standard kinematic equations for constant acceleration motion. However, the accelerometer measures gravity alongside the desired linear acceleration, and this component must be removed in order to perform double integration of linear acceleration to estimate translation. Moreover, to combine measurements from the accelerometer, which are expressed in the Device Reference Frame, with the state of the filter, which stores the translation with respect to the World Reference Frame, the rotation between the reference frames needs to be compensated.

In our approach the gravity vector is estimated at the start of the application by measuring the accelerometer output in the very first frames and filtering it out by a low-pass filter. Every time the acceleration from the inertial sensor is available, the rotation between the reference frames is compensated by rotating the accelerometer output vector using the transformation provided by the Orientation EKF. This allows for removing the gravity component as detailed in the following equation:

$$\mathbf{a}_k^{WRF} = \exp(\boldsymbol{\theta}_k)\,\mathbf{a}_k^{acc} - \mathbf{g}\,, \quad (10)$$

Figure 3. Mobile setup used in the experiments.

where $\mathbf{a}_k^{acc}$ represents the measurement from the accelerometer, $\boldsymbol{\theta}_k$ represents the current rotation, $\mathbf{g}$ is the estimated gravity vector, and $\mathbf{a}_k^{WRF}$ is the resulting linear acceleration in the WRF, which will be used as measurement for the Kalman filter in the updates. After rotation is compensated, the position estimation along the three axes is computed by a linear Kalman filter.

The state of the filter is given by position, velocity and acceleration of the device:

$$\mathbf{x}_k^{pos} = \begin{pmatrix} \mathbf{p}_k \\ \mathbf{v}_k \\ \mathbf{a}_k \end{pmatrix}. \tag{11}$$

The predict phase is based on the kinematic equations of motion:

$$\mathbf{x}_{k|k-1} = F^{pos}\mathbf{x}_{k-1|k-1}, \quad F^{pos} = \begin{pmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}. \tag{12}$$

The Kalman updates are linear as well. In the blind update, the measurement is given by the linear acceleration only, while the non-blind update fuses the linear acceleration from the sensor with the position estimated by the camera tracker. The measurement noise covariance matrices R^{pos} use the accelerometer noise values r^{acc}, with the addition of the SlamDunk measurement uncertainties r_k^{visual} in case of the non-blind update.

5. Experimental Results

In this section, we present quantitative and qualitative results obtained by running three different versions of mobile SlamDunk. The first version is the original formulation of the algorithm [5], which does not include any fusion of inertial measurements by Kalman filters and thus deploys visual tracking only. The second version is obtained by integrating into SlamDunk only the Orientation Kalman filter (Fig. 2, left side) and using the position estimations provided by the visual tracker. The third version consists in the integration into SlamDunk of both the Orientation and Position Kalman filters (*i.e*. the overall pipeline depicted in Fig. 2). The reference mobile platform used throughout all experiments is a Samsung Galaxy Tab Pro 10.1 tablet running Android 4.4.2 on which we mounted the Structure depth sensor from Occipital [19] as shown in Fig. 3.

To obtain quantitative experiments we have processed three RGB-D sequences, with the RGB channels obtained by the colour camera of the tablet and the depth channel by the Structure sensor. Some RGB frames of these sequences are shown in Fig. 4. Prior to the acquisitions, we calibrated the colour camera and the depth sensor so to be able to record aligned RGB and depth frames. To estimate the ground truth pose, we placed a chessboard in the scene and used the knowledge of the intrinsic parameters of the RGB camera. In some parts of the sequences the device moves away from the chessboard so to allow exploration of a wider environment, later coming back to a position which enables to see the chessboard again. Accordingly, ground-truth poses are available only for those frames featuring the presence of the chessboard. We adopted this simple methodology due to the absence of publicly available datasets that include both RGB-D and inertial data together with ground-truth information. To foster development in this area of research, as an additional contribution of this paper we will make our datasets and the corresponding ground-truth publicly available. The comparison between the three versions of mobile SlamDunk is based on RMSE values calculated using the methodology and tools of the TUM RGB-D benchmark dataset [21].

The quantitative results are reported in Tab. 1. In the case of the sequence *Room Closets*, both formulations of mobile SlamDunk including Kalman filtering turn out significantly more accurate than the version relying on visual tracking only. This is mainly due to the scarcity of distinctive and reliable visual features on the surfaces present in the explored environment. Qualitative results dealing with the *Room Closets* sequence are shown in Fig. 6, which also highlights how a large portion of the room is occupied by a white (*i.e*. nearly feature-less) closet. In the other two sequences, alike, the SlamDunk version integrating the Orientation Kalman filter neatly outperforms the algorithm relying on visual tracking only. Moreover, in all the three sequences we observed a higher error in case the Position Kalman filter is included compared to the results attained using only the Orientation Kalman filter. We ascribe this to propagation of the errors associated with the double integration process required to convert linear accelerations into position measurements.

We also present the mean execution times of the different operations, as shown in Tab. 2. These results clearly show that our lightweight approach does not significantly affect the frame rate of the system, even in the case of a consider-

Measurement	RMSE Kalman (Orientation)	RMSE Kalman (Orientation + Position)	RMSE without Kalman
Room Closets	**0.0532**	0.0710	0.3134
Kitchen	**0.0372**	0.0574	0.0376
Bookcases	**0.0257**	0.1852	0.0365

Table 1. RMSE values (meters) of the different versions with respect to the calculated ground-truth of the sequence.

Operation	Execution Time
Predict + Update blind (Orientation)	0.022 milliseconds
Predict + Update non-blind (Orientation)	0.030 milliseconds
Predict + Update blind (Position)	0.015 milliseconds
Predict + Update non-blind (Position)	0.022 milliseconds

Table 2. Execution times of the Kalman operations.

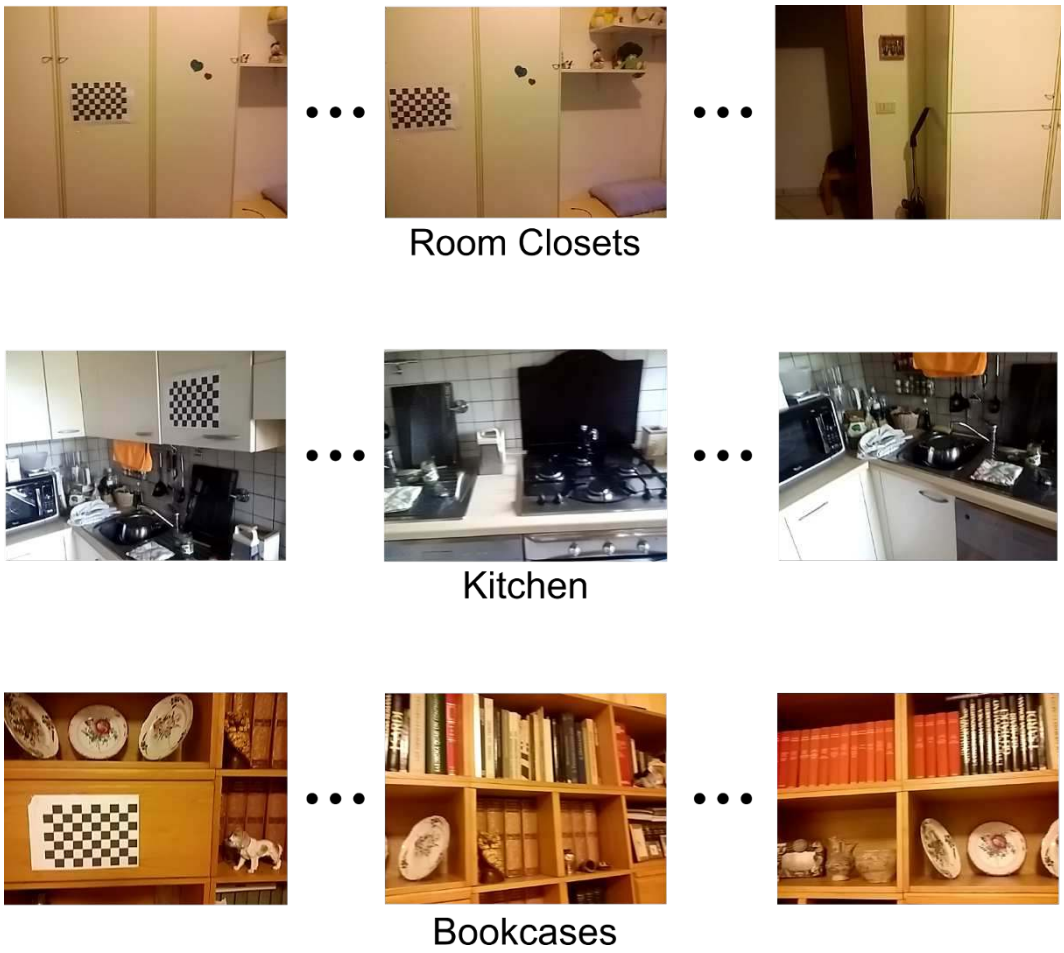

Figure 4. Some RGB frames from the three sequences used to obtain quantitative results.

able number of updates that follow the tracking phase.

Some final reconstructions can also be qualitatively inspected in Fig. 5 and Fig. 6. These two examples show that the improved tracking accuracy provided by the fusion of inertial and visual measurements can produce notable differences in the overall 3D reconstruction quality.

An interactive view of some of the captured reconstructions has been made available as supplementary material.

6. Conclusion and Future Work

Depth sensing on mobile devices will likely become a commodity in the near future so to enable new applications

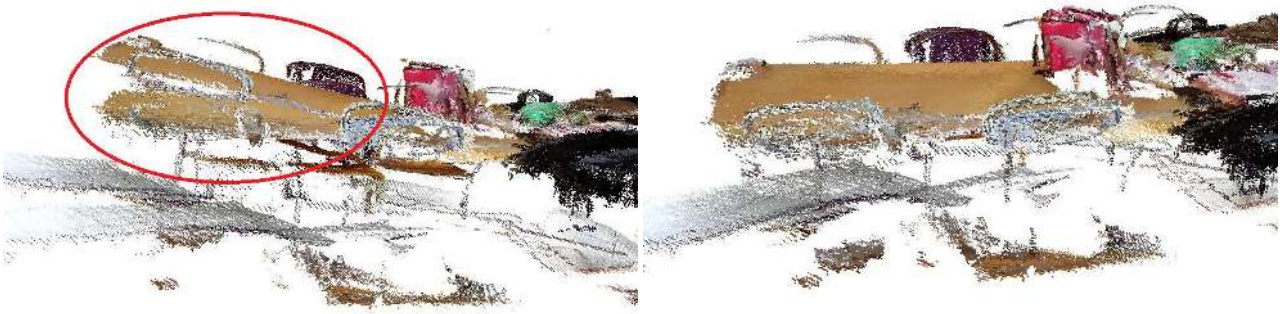

Figure 5. 3D reconstruction of a table using visual measurements only (left) and fusing inertial and visual measurements (right).

Figure 6. Partial 3D reconstruction of a poorly textured room using visual measurements only (left) and fusing inertial and visual measurements (right).

and deliver new experiences to users. In this work, we have shown a simple but effective framework to leverage the inertial sensors widely available in smartphones and tablets to ameliorate RGB-D SLAM on mobile devices. Our findings vouch that fusing the angular velocity measurements provided by the gyroscope alongside visual camera tracking according to an Extended Kalman Filtering formulation can consistently and significantly improve pose estimation with respect to a purely visual SLAM approach. However, with our implementation we found it much less beneficial the deployment of the accelerometer due to issues arising in the double integration process required to attain position measurements. Therefore, we plan to investigate on how to better deploy linear acceleration data for mobile RGB-D SLAM, possible approaches including more accurate numerical integration of acceleration data, velocity reset in near-stationary conditions or more complex filtering schemes, as described *e.g.* in [23].

Eventually, we have created and will make available the first dataset featuring RGB-D streams synchronized with inertial measurements as well as ground-truth camera poses.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, sept 1987.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346 – 359, September, 10 2008.

[3] J.-L. Blanco. A tutorial on se(3) transformation parameterizations and on-manifold optimization. Technical report, University of Malaga, Sept. 2010.

[4] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. Globally consistent 3d mapping with scan matching. *J. of Robotics and Autonomous Systems*, 56:130–142, Feb 2008.

[5] N. Brunetto, N. Fioraio, and L. Di Stefano. Interactive RGB-D SLAM on mobile devices. In *ACCV Workshop on Intelligent Mobile and Egocentric Vision*, Singapore, Nov, 2 2014.

[6] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotics: Science and Systems (RSS)*, Berlin, Germany, June 2013.

[7] A. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067, jun 2007.

[8] E. Eade. *Monocular Simultaneous Localisation and Mapping*. PhD thesis, Cambridge University, 2008.

[9] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3D mapping with an RGB-D camera. *IEEE Trans. Robot.*, 2013.

[10] N. Fioraio and L. Di Stefano. SlamDunk: Affordable real-time RGB-D SLAM. In *ECCV Workshop on Consumer Depth Cameras for Computer Vision*, Zurich, Switzerland, Sept, 6 2014.

[11] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. *The Int. J. of Robotics Research*, 31(5):647–663, feb 2012.

[12] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *International Symposium on Robotics Research (ISRR)*, pages 1–16, 2011.

[13] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[14] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Int. Symposium on Mixed and Augmented Reality (ISMAR)*, pages 225 –234, nov. 2007.

[15] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Int. Conf. on Computer Vision (ICCV)*, Nov, 6 2011.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–119, January, 5 2004.

[17] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *10th IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, Washington, DC, USA, 2011.

[18] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *International Conference on Computer Vision (ICCV)*, pages 2320–2327, nov 2011.

[19] Occipital Inc. The Structure sensor. http://structure.io/, 2014.

[20] U. Qayyum and J. Kim. Inertial-kinect fusion for outdoor 3d navigation. In *Australasian Conference on Robotics and Automation (ACRA)*, 2013.

[21] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Int. Conf. on Intelligent Robot Systems (IROS)*, Oct. 2012.

[22] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3D reconstruction on mobile phones. In *Int. Conf. on Computer Vision (ICCV)*, Sydney, Australia, Dec, 13 2013.

[23] S. Weiss and R. Siegwart. Real-time metrix state estimation for modular vision-inertial systems. In *Int. Conf. on Robotics and Automation (ICRA)*, Shangai, China, May, 9-13 2011.

[24] T. Whelan, M. Kaess, J. Leonard, and J. Mcdonald. Deformation-based loop closure for large scale dense RGB-D SLAM. In *Int. Conf. on Intelligent Robot Systems (IROS)*, Tokyo, Japan, November 2013.