

Dense Tracking with Range Cameras Using Key Frames

Andrés Díaz^{1(✉)}, Lina Paz², Eduardo Caicedo¹, and Pedro Piniés²

¹ School of Electrical and Electronic Engineering, Universidad del Valle, Cali,
Colombia

{andres.a.diaz,eduardo.caicedo}@correounivalle.edu.co

² Oxford Mobile Robotics Group, University of Oxford, Oxford, UK
{linapaz,ppinies}@robots.ox.ac.uk

Abstract. We present a low cost localization system that exploits dense image information to continuously track the position of a range camera in 6DOF. This work has two main contributions: First, the localization of the camera is performed with respect to a set of keyframes selected according to a spatial criteria producing a less populated and more uniform distribution of keyframes in space. This allows us to avoid the computational overload caused by having to estimate a depthmap at the frame rate of the camera as it is common in other dense sequential methods. Second, we propose a two-stage approach to compute the current location of the camera with respect to its closest keyframe. During the first stage, our system calculates an initial relative pose estimate from a sparse set of 3D to 2D point correspondences. This estimate is then refined during the second stage using a dense image alignment. The refinement step is stated as a Non Linear Least Squares (NLQs) optimisation embedded in a coarse to fine approach that minimizes the photo-consistency error between the current image and a warped version of the image associated to the closest keyframe and its depth map.

To validate the accuracy of our system, we conducted experiments using datasets with perfectly known trajectory and with both, perfect ray-traced images and images with noise and blur. We also evaluate the accuracy of the system using datasets with RGBD images taken at different frame-rates, and the improvements in convergence due to our coarse-to-find approach. Our assessment shows that our system is able to achieve millimeter accuracy. Most of the expensive calculations are carried out by exploiting parallel computation on a GPU.

Keywords: Localization using dense techniques · Range camera · 6 DOF · Keyframes · Non linear squares optimization · Parallel computing · Noise and blur · Different frame rates · Coarse-to-fine scheme

1 Introduction

The aim of (*Simultaneous Localization and Mapping*, SLAM) is to estimate the position of a mobile platform moving in an unknown environment and, at the

same time, build a map of this environment in an incremental way. Many solutions have been proposed using different sensor modalities; sonar rings, laser scanners, perspective cameras, omnidirectional cameras, stereo cameras, inertial sensors, and combination of them. Due to its versatility and low cost, vision-based SLAM systems have shown great potential in the fields of mobile robotics, augmented reality, wearable devices, user interfaces, surgical procedures, among others. Popular visual SLAM systems use sparse features such as corners [6], edges [12], or planes [18], and have as principal goal to localize the camera. More recently, dense techniques for localization and mapping have gained attention [14]. They do not use sparse features but all pixels in the image, showing better accuracy for the localization and providing high quality maps. For achieving real time performance these dense methods are based on optimization algorithms implemented on GPUs. Moreover, the mapping and localization algorithms run in parallel, on two or more CPU cores or GPUs, without the need of propagating a joint probability of the camera-features estimates. With these techniques more accurate estimates of trajectories (even with fast motions) and more robust augmented reality applications are obtained.

We can find three main frameworks in the literature about tracking the position of a camera using dense information: frame-to-frame [16] (when the pose of the camera is computed with respect to a locally generated depth map), frame-to-model-vertices [15], frame-to-model-TSDF [2]:

In frame-to-frame approaches using the current image I_c^{RGB} and a reference image I_r^{RGB} with its corresponding depth map I_r^{depth} we can compute the relative transformation between both as follows: using an initial estimate of the relative position between the current and the reference image we can use the I_r^{depth} and I_c^{RGB} to create a warp image I_{warp}^{RGB} that should resemble the reference image I_r^{RGB} . We can then calculate the differences and correct accordingly the transformation in order to reduce them.

In a frame-to-model-vertices approach the idea is to estimate the pose of the current camera with respect a continuously built 3D dense model. Given the current depth map at time k a set of vertices and normals can be calculated from it (V_k, N_k) respectively. From the model and the current estimate of the pose of the camera with respect to the model we can make a prediction of these elements ($\hat{V}_k, \hat{N}_k$). Again, comparing the observations with the predictions we can correct the current pose estimate.

Finally in frame-to-model-TSDF approaches the camera is also localized with respect to the model but instead of predicting vertices and normals what is compared are the values of the truncated signed distance function (TSDF) used to represent the model.

The advantage of the model based approaches is that drift in the camera pose does not accumulate if the model is continuously observed. In the case of the frame-to-frame approach, the advantage is that a model is not required to compute the trajectory of the camera. In that sense, model based approaches are more similar to Bundle Adjustment techniques whereas frame-to-frame ones to Visual Odometry algorithms.

In this paper, we focus on the problem of continuously localizing an RGBD camera that moves freely in 6DOF following a frame-to-frame approach. Our contribution is in the use of a set of depth maps available only for a set of selected frames or “keyframes”. In this way, we can overcome the computational payload that commonly whips other visual localization systems. These techniques will be used in a posterior work together with a dense mapping module for making localization and mapping using only a monocular camera.

In Sect. 2 we analyse the most important sparse and dense localization systems. In Sect. 3 we focus on the proposed technique for dense localization of a range camera with keyframes. We provide details about the criteria for keyframe selection, the 3D-2D visual odometry technique used for getting an initial pose estimate and the refinement of the camera pose with NLQs. Our experiments and the results of the performance of the algorithms implemented on graphic processor are explained in Sect. 4. Finally, we draw our conclusions and future work in Sect. 5.

2 Related Work

Cameras are inexpensive sensors that provide rich information from the observed scene, therefore becoming an ideal sensor for many robotics applications with embedded visual navigation systems. In addition, increase in computing power and new developments in computer vision techniques (e.g. robust feature detectors, outlier rejection and matching algorithms) allow their use in real time applications. In robotics, the first visual SLAM systems proposed were based on the Extended Kalman Filter (EKF) estimator [5, 6]. Other systems have excelled for introducing novel techniques such as patch association and planes-based map building [18], inverse depth parametrization [4] and large scale SLAM [17]. Unfortunately, these approaches provide a sparse map representation, only hundreds of points can be estimated at a given instant, which limits the accuracy of the camera localization.

The system presented in [11] radically changed the filtering paradigm used until that moment to non-linear optimisation techniques using keyframes. This system is pioneer in balancing the computations between two CPU cores, one for localization and the other for mapping respectively. It uses a Five-Point algorithm [20] to initialise the system. As we mentioned before, keyframes are selected to reduce the computational complexity of the optimization. Camera tracking is still carried out on the intermediate camera frames by re-projecting the estimated 3D points and minimizing a photoconsistency error. Despite this approach is able to create very accurate maps in real time, it represents the environment sparsely. The map can only contain up to 6000 points and 120 keyframes. Nonetheless, this work represents a transition between sparse and dense systems. In our approach, we also use keyframes to track the position of the camera.

Recently, [21] proposed a mapping approach able to compute dense depth maps from monocular images. In this context, dense means that every pixel in

the image contains an estimate of the depth to the corresponding object in the scene. Depth maps are computed by minimizing an energy function composed of a regularisation term and a photometric term (data term). A generalized threshold scheme is used for minimizing the photometric error of the energy function. The localization is based on bundle adjusting a set of keyframes [11] on a CPU. The system can reach 24fps for a depth map resolution of 480×360 when a GPU is used to calculate the depthmaps. Similarly, the work in [16] builds high quality dense maps of small environments using a GPU and a monocular camera in real time. Again, depth maps are obtained by minimising an energy function. The main difference with respect to the previous approach is that a decoupling scheme is implemented during the optimization. This way, the problem is decomposed into two tied minimization problems. In the first problem, a convex energy function composed of a regularisation and a quadratic terms is minimized using a primal-dual algorithm [3]. For the second problem, an exhaustive search is carried out over a finite range of discrete inverse depth values. The camera is localized by aligning the current observed frame to a synthetic one created using a nearby image and its depthmap. The camera motion corresponds then to the relative transformation that produces the best alignment between the actual and the synthetic images. A pyramid of images is built to accelerate convergence allowing faster motions of the camera. The initialization of the system is similar to [11].

The large scale direct monocular SLAM system LSD-SLAM [7] uses direct image alignment and filtering-based estimation of semi-dense depth maps. In addition to the photometric residual, the authors incorporate a depth residual which penalizes deviations in inverse depth between keyframes, allowing to directly estimate the scaled transformation between them, operating on Sim(3). Moreover, the effect of noise depth values is included probabilistically into tracking. The global map is represented as a pose graph consisting of keyframes as vertices with 3D similarity transforms as edges. The system detects loop closures and scale-drift, and runs on real-time on a CPU.

In [19] an energy-based approach to estimate the rigid body motion of a handheld RGB-D camera for a static scene is presented. The reprojection error is defined such that the warped second image matches the first image, using consecutive images of the trajectory. The authors minimize this non-convex reprojection error by a sequence of convex optimization problems obtained by linearizing the data term and solving the resulting normal equations in a coarse to fine manner. This system is better than the iterative closest point ICP algorithm for continuous camera tracking and is able to perform at 12.5 Hz on CPU.

3 Dense Localization Algorithm

In this section we are going to explain the main processes developed for estimating the camera pose using a dense technique. Without loss of generality, consider a sequence of images I_{c_i} , for $i = 1 : m$, gathered by the camera. Our aim is to estimate the relative transformation $T_{r_j c_i}$, for $j = 1 : n$, at each frame, where

r_j is the closet keyframe. Algorithm 1 presents the order in which the processes are carried out.

Since a coarse-to-fine scheme is used, the T_{cr} obtained after convergence in a level is used in the following lower level as initial estimate, improving convergence, specially in lower levels where the computational cost is higher. The Levenberg-Marquardt iterates until the stopping criteria is fulfilled: the norm of variations in the motion vector is small (less than a threshold ε) or the number of iteration reaches the maximum value $iter_{max}$.

Algorithm 1. Relative transformation using dense information

Data: Current Intensity Image I_c , Reference Intensity Image I_r and Reference Depth Image D_r .

Result: Relative Transformation Matrix T_{cr} .

Compute the initial T_{cr}

Build a four level pyramid for I_c , I_r and D_r

Define the maximum number of iterations $iter_{max}$ for each level

for level $l = 4 : 1$ **do**

 Compute the warped Image I_w , the residual r , the cost C and the Jacobian J for level l

 iter=1

 found=false

while ($found == false$) and ($iter < iter_{max}$) **do**

 Compute the variation in the motion vector $d\xi = (J'J + \alpha I)^{-1} J'r$ where ξ is the minimal parametrization of the motion

 Update the new Relative Transformation $T_{cr_{new}}$ using $d\xi$:

$T_{cr_{new}} = exp_{SE3}([d\xi]_x) * T_{cr}$

 Compute the new warped Image $I_{w_{new}}$ and the new cost C_{new}

if $|d\xi| < \varepsilon$ **then**

 found=true

else

if $C_{new} < C$ **then**

 Update the cost C , the relative transformation T_{cr} and the warped Image I_w with the new values

 Compute the Jacobian J

 Update α , $\alpha \leftarrow \frac{\alpha}{10}$

else

 Update α , $\alpha \leftarrow \alpha * 10$

 iter++

 Use the estimated T_{cr} as initial value in the next lower level

The camera trajectory is obtained concatenating relative transformations, e.g. $T_{wr1}T_{r1r2}T_{r2r3}...T_{r_{j-1}r_j}T_{r_jc_i}$, where the world coordinate system corresponds to the coordinate system of the first keyframe, therefore $T_{wr1} = [I | 0]$. A dense model of the environment can be generated using the poses and depth maps from keyframes T_{wrj}, D_{rj} .

In the remaining of this section we will describe the criteria for keyframe selection, the calculation of an initial pose estimate, how to refine this estimate using a photometric error obtained by aligning the warped and observed images, and finally, its minimization using Levenberg-Marquardt.

3.1 Keyframe Selection

The mapping and global optimization is performed only for certain frames of reference in order to achieve higher performance. These frames are known as keyframes and are selected according to certain temporal and spatial criteria [11].

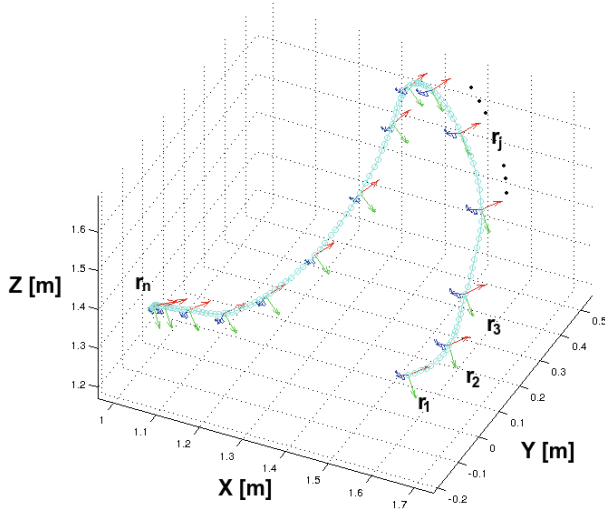


Fig. 1. The camera trajectory is drawn in cyan. Keyframes r_j are drawn with their local coordinate systems. They are selected based on a time criteria, e.g. for every nine images a new keyframe is initialized.

For the sake of simplicity, Fig. 1 shows an example of keyframes selected based just on a time criteria. Each keyframe r_j has a transformation matrix T_{wr} , an RGB image I_r and a depth map D_r . The system saves all keyframes with its corresponding information, which allows it to initialize new keyframes, estimate the relative pose of the current image I_c with respect to the nearest keyframe and perform optimization for getting a consistent map and trajectory.

We set thresholds to the displacement and to the change in orientation (no time criteria): our system initializes a new keyframe when the displacement with respect to any other keyframe is greater than 0.13 m or when the sum of changes in orientation with respect to the last keyframe (around X, Y and Z axis) is greater than 30° . Thus, if the displacement and change in orientation of the

camera is smaller than these thresholds, no new keyframes will be initialized. In this way, the map is compact in the sense that it does not grow arbitrarily with time due to little or no movement, even with repeated movements in restricted areas as we will see in the results section.

The frame of the camera when the system is initialized is taken as the first keyframe, with transformation matrix $T_{wr_1} = [I \mid 0]$. For the following images I_{c_i} , the position of the previous image is used to define the nearest keyframe. The estimate of relative motion is performed considering the correspondence between the RGB image associated to the nearest keyframe and the current image. The pose of the new image is defined as $T_{wr_{near}} T_{r_{near} c_i}$.

3.2 Initial Pose Estimate

In the motion estimate process the relative motion between the current image and the nearest keyframe is computed. There are three methods for computing the relative motion T_{cr} between images I_r and I_c using two sets of corresponding features f_r and f_c :

- 2D to 2D. Both f_r and f_c are specified in 2D image coordinates.
- 3D to 3D. Both f_r and f_c are specified in 3D.
- 3D to 2D. f_r are specified in 3D and f_c are their corresponding reprojections on the image I_c .

We are interested in 3D to 2D methods where the transformation T_{cr} is computed from correspondences between 3D points χ_r and 2D points p_c . χ_r can be obtained from the depth map D_r associated to the reference image I_r , it means, the image taken from the keyframe. We use the depth map loaded from the dataset of A. Handa [9]. In Fig. 2 we can see the RGB image I_r and its corresponding depth map D_r .

We apply a SURF feature detector and a SURF descriptor extractor to I_r and to I_c which belongs to the set of images associated to I_r , that is I_{c_i} for $i = 1, 2, 3, \dots, m$. Then a matcher based on the FLANN algorithm computes the matches between SURF descriptors of both images. The matches are validated if they fulfil three conditions. The first one is that the distance between the two matched descriptors is smaller than a threshold. The second one is that the correlation index between a patch around the descriptor in I_r and the descriptor in I_c is greater than a threshold. The final one is that the matches are inliers when the fundamental matrix is computed using RANSAC. This means that the descriptor in I_c must lie on the corresponding epipolar line.

We set the threshold in distance to five times the minimum distance between all matches and the correlation index threshold to 0.7. Finally, as the goal is to find T_{cr} that minimizes the image reprojection error, we apply a solver based on PNP algorithm that uses RANSAC for outlier rejection

$$\arg \min_{T_{cr}} \sum_k \|p_{c_i}^k - \hat{p}_{r_j}^k\|^2, \quad (1)$$

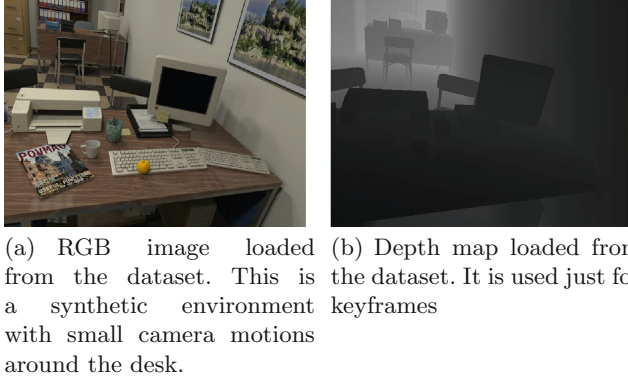


Fig. 2. Perfect ray-traced RGB Image (a) and Depth map (b) of the dataset of A. Handa [9]. They are used for computing the initial pose estimate and for the posterior refinement.

where $\hat{p}_{r_j}^k$ is the reprojection of the k^{th} 3D point $\chi_{r_j}^k$ into image I_{c_i} according to the transformation T_{cr} . The transformation of a feature $p_{r_j}^k$ with 2D coordinates μ to a spacial coordinate χ_r^k and the transformation of a spacial point $\chi_c^k = T_{cr}\chi_r^k$ to a point in the image plane c_i , $\hat{p}_{r_j}^k$, are described in Subsect. 3.3.

3.3 Pose Refinement

The goal of the localization algorithm is to estimate the relative transformation matrix between two camera poses T_{cr} (Fig. 3) using all the depth data D_r of the associated keyframe and all the intensity data of I_r and I_c . The camera pose of the current Image C_c is given by $C_c = T_{cr}C_r$, where C_r is the camera pose of the reference image. The current image is warped with the transformation T_{cr} in order to compare it with the reference image. In case the transformation T_{cr} is the true 3D transformation between the two cameras, the images I_c and I_r should be identical and the residual should be zero [1].

The warping consists in projecting the reference depthmap D_r into 3D space, transforming by T_{cr} and projecting back into the current image I_c . Algorithm 2 summarizes the steps of this process. K is the intrinsic calibration matrix, π represents the function that computes normalized homogeneous coordinates and $\hat{\mu}$ is the homogeneous version of the coordinate in the image plane. A residual function is defined as the difference between $I_r(\mu)$ and $I_w(\mu)$,

$$r(\mu, T_{cr}) = I_r(\mu) - I_c(K\pi(T_{cr}D_r(\mu)K^{-1}\hat{\mu})). \quad (2)$$

Since the depth $D_r(\mu)$ is known, T_{cr} is defined as the argument that minimizes the residual function. T_{cr} is a homogeneous transformation matrix defined by a rotation matrix R_{cr} and a translation vector t ,

$$T_{cr} = \begin{bmatrix} R_{cr} & t \\ 0 & 1 \end{bmatrix}. \quad (3)$$

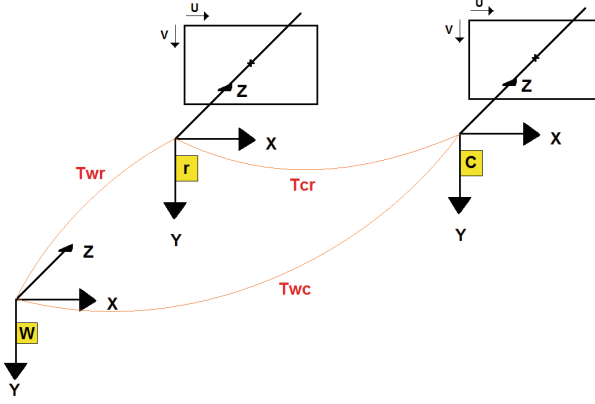


Fig. 3. Current image I_c and reference image I_r with their corresponding local and global coordinate systems. The transformations that relate coordinate systems are drawn in red. (Color figure online)

Algorithm 2. Warping an Image

Data: Current Intensity Image I_c , Reference Depth Image D_r , Transformation T_{cr} and Intrinsic Matrix K .

Result: Warped Image I_w .

Transform coordinates in the image plane I_r to spacial coordinates:

$$\chi_r(\mu) = D_r(\mu)K^{-1}\hat{\mu}$$

Change of coordinate system: $\chi_c(\mu) = T_{cr}\chi_r(\mu)$

Transform spatial coordinates to coordinates in the image plane: $\hat{\mu} = K\pi(\chi_c)$,

$$\text{where } \pi(\chi_c) = (X_n, Y_n, 1)^T = \left(\frac{X_c}{Z_c}, \frac{Y_c}{Z_c}, 1 \right)^T$$

Compute the warped Image: $I_w(\mu) = I_c(K\pi(T_{cr}D_r(\mu)K^{-1}\hat{\mu})) = I_c(\hat{\mu})$

The vector $\xi = (\mathbf{w}, \mathbf{v})$ represents the relative transformation matrix T_{cr} with the smallest possible number of parameters. The orientation of the vector $\mathbf{w} = (w_1, w_2, w_3)$ defines the axis of rotation and the magnitude of the vector defines the amount of rotation. The vector $\mathbf{v} = (v_1, v_2, v_3)$ corresponds to the translation t rotated over $\mathbf{w}$.

An exponential mapping between the 6-element vector ξ in the Lie algebra $\mathfrak{se}(3)$ [23] to members of the group $SE(3)$ is made, as in [11] with a motion model and feature association (sparse techniques) or in [16] using a dense estimate of rotation [13] and image alignment. Computing the exponential mapping involves the operator $[\cdot]_x$, which maps a vector to its *skew* matrix,

$$\exp_{SE3}([\xi]_x) = T_{cr}, \quad (4)$$

where $[\xi]_x$ is

$$[\xi]_x = \begin{bmatrix} 0 & -w_z & w_y & v_x \\ w_z & 0 & -w_x & v_y \\ -w_y & w_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (5)$$

The updated transformation T_{cr} is expressed as the product of an initial transformation T_0 and an update in the motion vector $d\xi = \xi - \xi_0$, that is,

$$T_{cr} = dT * T = \exp_{SE3}([d\xi]_x) * T_0. \quad (6)$$

This way, the singularities are avoided since the incremental update is calculated in the tangent space around the identity (i.e. it is always very small) and mapped back onto the manifold $SE(3)$. With this transformation, the residual defined in (2) is

$$r(\mu, \xi) = I_r(\mu) - I_c(K\pi(\exp_{SE3}([d\xi]_x)T_0D_r(\mu)K^{-1}\hat{\mu})). \quad (7)$$

Finally, the localization problem consist in finding ξ that minimizes the following cost function:

$$\min_{\xi} \frac{1}{2} \int [r(\mu, \xi)]^2 d\mu. \quad (8)$$

This is a non-linear minimization problem that we solve with Levenberg Marquardt [10]. Since r is non-linear and non-convex in ξ we linealize it using a first order Taylor expansion:

$$r(\xi) \approx r(\xi_0) + \left. \frac{dr(\xi)}{d\xi} \right|_{\xi_0} (\xi - \xi_0). \quad (9)$$

The quantity $\frac{dr(\xi)}{d\xi}$ is the Jacobian of r and is given by

$$J = \begin{pmatrix} \nabla r_1(\xi)^T \\ \nabla r_2(\xi)^T \\ \vdots \\ \nabla r_i(\xi)^T \end{pmatrix}, \quad (10)$$

where $\nabla r_i(\xi)$ is the derivative of the i^{th} component of the residual vector.

Computation of the Jacobian. Lets define a function W for the warping process

$$W(\mu, \xi) = \pi(\overbrace{K\exp_{SE(3)}([d\xi]_x)T_0D_r(\mu)K^{-1}\hat{\mu}}^{p_{1n}}), \quad (11)$$

p_1

where $\pi(X) = (\frac{x}{z}, \frac{y}{z}, 1)^T$, p_1 denotes the transformed point and p_{1n} is the transformed back-projected (normalized) point. The residual can be written as

$$r(\mu, \xi) = I_c(W(\mu, \xi)) - I_r(\mu). \quad (12)$$

The derivative of the residual can be written using the chain rule,

$$\frac{dr(\xi)}{d\xi} = \frac{dI_c(W(\xi, \mu))}{dW(\xi, \mu)} \Big|_{W(\xi_0, \mu)} \frac{dW(\xi, x)}{d\pi} \Big|_{p_{1n}} \frac{d\pi(\xi, x)}{d\xi} \Big|_{\xi_0}. \quad (13)$$

The first term $\frac{dI_c(W(\xi, \mu))}{dW(\xi, \mu)} \Big|_{W(\xi_0, \mu)}$ is just the image gradient of the warped image $\nabla I_w(\mu)$. The second term involves the projection $\pi(\bullet)$ and therefore

$$\frac{dW(\xi, x)}{d\pi} \Big|_{p_{1n}} = \begin{bmatrix} \frac{1}{z} & 0 & \frac{-x}{z^2} \\ 0 & \frac{1}{z} & \frac{-y}{z^2} \end{bmatrix}. \quad (14)$$

The third term $\frac{d\pi(\xi, x)}{d\xi} \Big|_{\xi_0} = \left[\frac{d\pi}{dw_x} \frac{d\pi}{dw_y} \frac{d\pi}{dw_z} \frac{d\pi}{dv_x} \frac{d\pi}{dv_y} \frac{d\pi}{dv_z} \right]$. Putting together all these derivatives, it becomes

$$\frac{d\pi(\xi, x)}{d\xi} \Big|_{\xi_0} = K \begin{bmatrix} 0 & p_z & -p_y & 1 & 0 & 0 \\ -p_z & 0 & p_x & 0 & 1 & 0 \\ p_y & -p_x & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (15)$$

The first 3×3 submatrix is the negative skew matrix of the vector p_1 and the second 3×3 submatrix is the identity matrix. Equation (15) simplifies to

$$\frac{d\pi(\xi, x)}{d\xi} \Big|_{\xi_0} = K [-p_1]_x \mid I_3]. \quad (16)$$

Now that the Jacobian was defined, the variation in the motion vector $d\xi$ can be obtained by solving (17).

$$J^T J d\xi = -J^T r. \quad (17)$$

For the Levenberg-Marquardt method the Hessian can be approximated by $J^T J + \alpha I$. For a large α the method approaches the Steepest Descent method, while for a small α the method approaches the Gauss-Newton method. Generally, the Steepest Descent works better further away from the local minima and the Gauss-Newton approximation works better close the local minima where the quadratic approximation is good [1]. We start with a small initial value of α , $\alpha = 1e-3$. After each iteration, the new cost is compared with the previous cost (see Algorithm 1). If the cost has decreased, α is reduced by a factor of 10 and the parameters are updated. If the cost has increased, the parameters are not updated and α is increased by a factor of 10. Solving (17) is efficient because we just need invert a 6×6 matrix and use $d\xi$ in (6).

4 Results

The dense localization algorithm runs on a laptop Hewlett Packard with a processor Intel Core i7 - 2.2 GHz, 11.7 GB of RAM memory, a graphic processor NVIDIA GEFORCE 755M and Ubuntu 14.04 as operating system. OpenCV

is employed for processing images, OpenGL for drawing the camera trajectory and the keyframes, and Cuda 6.0 for getting a better performance using graphic processor. The following analysis is made using the synthetic RGB-D dataset of A. Handa [9] and is based on three criteria; time, convergence and accuracy.

4.1 Time

We have implemented the most important algorithms of the dense localization module on GPU. The Jacobian is computed in 11.1 ms in the lower level (the full image, 640×480 pixels). The warped image and the residual is computed in 3.9 ms for the same level. The update in the camera position and orientation is made in 5.3 ms. The other algorithms are implemented on CPU. The building of the pyramid is made in 1.6 ms. The load of images I_r and I_{c_i} is made in 3.5 ms. The initial estimate is computed in 31.3 ms.

The system processes 10 fps at the moment, for two iterations of the Levenberg-Mardquart algorithm in the lower level of a four level pyramid. We believe that this performance can be improved by implementing all the algorithms of the localization module on GPU. This is left as future work.

4.2 Convergence

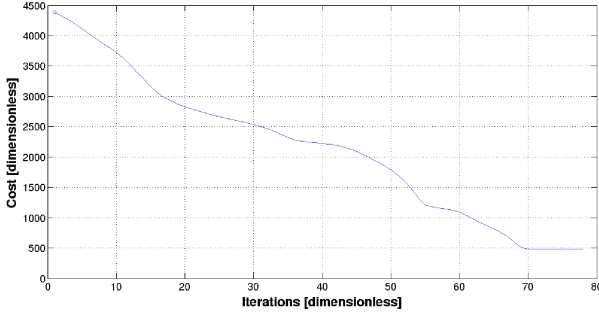
For analysing the performance of the algorithm when it uses a coarse-to-fine scheme, we present the number of iterations required by the algorithm for converging before and after implementing this scheme. In both cases the initial relative transformation is $T_{cr} = \begin{bmatrix} R_{cr} & 0.6t \\ 0 & 1 \end{bmatrix}$, where R_{cr} and t are respectively the ground truth in rotation and translation that relates frame r and c . The images 1 and 10 from the 20 fps dataset was used as I_r and I_c , respectively.

For the case in which a coarse-to-fine scheme is not used, the initial cost is 4386.8. In the iteration 76, the change in the motion vector $d\xi$ is less than $1e - 6$ and the algorithm stops, converging to a cost of 481.9.

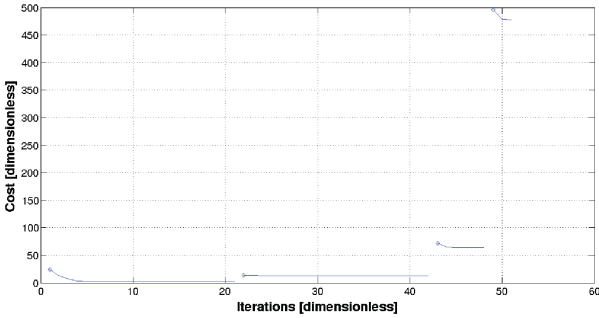
When a four level pyramid with a scale of 2 is built, the maximum number of iterations in each level is set to: 20, 20, 5 and 2, from higher to lower levels. The initial and final cost for each level is presented in Table 1.

Table 1. Information about convergence using a coarse-to-fine scheme

Level	Size [pix.]	Initial cost	Final cost	Max. iterations
4	60×80	24.27	2.47	20
3	120×160	13.89	12.67	20
2	240×320	72.00	64.20	5
1	480×640	495.44	477.72	2



(a) Convergence without using a coarse-to-fine scheme.



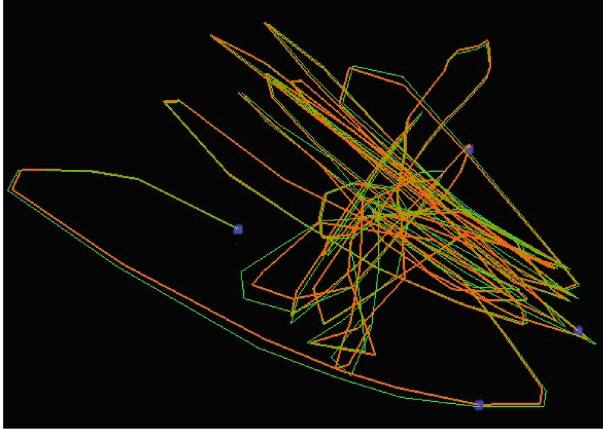
(b) Convergence using a four level pyramid.

Fig. 4. Comparison between using only one image and a four level pyramid. (a) requires 76 iterations with an image of 480×640 pixels for converging. (b) requires 2 iterations in the lower level (image of 480×640 pixels) for converging.

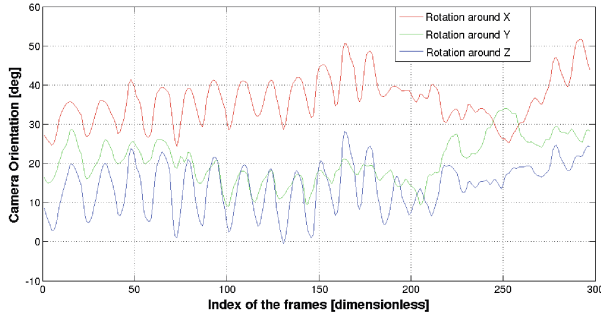
Note that the initial cost in the lowest level (bigger image) is very close to the final cost, unlike in the first case. For this reason only two iterations are required in the lowest level when a coarse-to-fine scheme is implemented (Fig. 4).

4.3 Accuracy

The accuracy in the camera position is quantified using three datasets; two of them consist of perfect ray-traced images taken at 20 fps and 60 fps and the last one consist of images with motion blur and noise, taken at 60 fps (60 fps-blur). We define the error as the Euclidean distance between the ground truth in trajectory and the estimated one. Figure 5(a) shows the estimated trajectory (in orange) and the ground truth (in green) for the 60 fps dataset, obtained in OpenGL and drawn incrementally as the relative transformations are computed and concatenated. The positions of keyframes are drawn as blue points. The analysis in orientation will be made for each axis, comparing the ground truth in orientation with the estimated one along the time. The ground truth in camera rotation around the axis X, Y and Z are shown in Fig. 5(b).



(a) Trajectory estimated (in orange) and ground truth (in green). The position of keyframes are drawn as blue points.



(b) Ground truth in rotations around X, Y, and Z

Fig. 5. Trajectory and orientation of the camera for the 60 fps dataset. The positions of keyframes are drawn as blue points in (a) (Color figure online)

In Fig. 6(a) we draw the error in camera position for the 60 fps dataset (no blur) as a solid red line and the error in camera position for the 60 fps-blur dataset as a dashed blue line, and highlight the errors of keyframes by drawing them as points. Figure 6(b) shows the error in rotations around X, Y, and Z for the 60 fps dataset and Fig. 6(c) shows the corresponding errors for the 60 fps-blur dataset. Note that the errors in both position and orientation for the images with motion blur and noise are greater than the errors for perfect ray-traced images. Errors obtained with the 20 fps dataset are very similar to the solid red line in Fig. 6(a) for camera position and to the errors in Fig. 6(b) for camera orientation so they are not shown.

Table 2. Summary of accuracy in camera position

	20 fps	60 fps	60 fps-blur
RMSE [cm]	<i>0.74</i>	<i>0.52</i>	<i>1.12</i>
Max. error [cm]	4.2	4.82	4.54
Mean error [cm]	0.44	0.33	0.92
Median error [cm]	0.25	0.22	0.87
Standard dev. [cm]	0.59	0.41	0.65

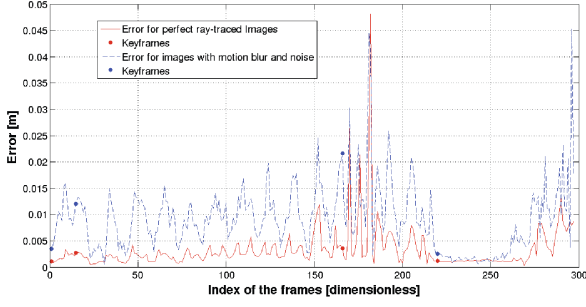
Table 3. Summary of accuracy in camera orientation

	20 fps			60			60-blur		
	X	Y	Z	X	Y	Z	X	Y	Z
RMSE [deg]	0.24	0.21	0.12	0.14	0.16	0.10	0.31	0.34	0.43
Max. error [deg]	0.74	1.52	0.75	0.35	1.63	0.93	1.06	1.34	0.70
Min. error [deg]	-0.64	-0.44	-0.24	-0.57	-0.72	0.40	-1.66	-1.15	-1.90
Mean error [deg]	-0.11	-0.02	-0.04	-4e-3	-0.06	-0.04	-0.02	-0.23	-0.34
Median Error [deg]	-0.13	-0.04	-0.06	-0.02	-0.05	-0.03	-2.9e-3	-0.22	-0.30
Standard dev. [deg]	0.21	0.21	0.11	0.14	0.15	0.10	0.31	0.25	0.27

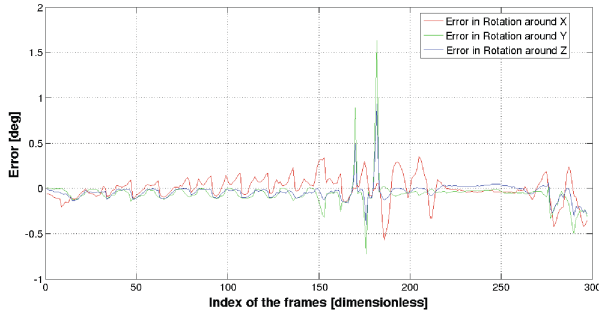
Table 2 shows the root mean square error RMSE, the maximum error, the mean error, the median error and the standard deviation for the camera position for the three datasets. With the 60 fps dataset (no blur) the RMSE and Standard deviation decrease although the maximum error increases, compared to the ones obtained with the 20 fps dataset. When the 60 fps-blur dataset is used, the RMSE and the standard deviation increases due to difficulties for getting a good initial estimate and for aligning images with blur and noise. Despite these difficulties, the algorithm converges to a good estimate. We can say that the system has high accuracy in position for both frame rates, and for both perfect ray-traced images and images with blur and noise; in the worst case, the error is 0.16 % of the distance navigated by the camera that corresponds to 6.93 m.

Table 3 shows the root mean square error, the maximum error, the minimum error, the mean error, the median error and the standard deviation, for the estimated rotations for the three datasets. The RMSE and the standard deviation decrease with the 60 fps dataset although the maximum error increases for rotations around Y and Z. When the 60 fps-blur dataset is used, the RMSE and the standard deviation increase due to the reasons given for the errors in camera position when the images have blur and noise. Note that the maximum error is 1.63° and the minimum error is -1.90° so we finally can claim that the system has high accuracy in both camera position and camera orientation for different frame rates and even with motion blur and noise.

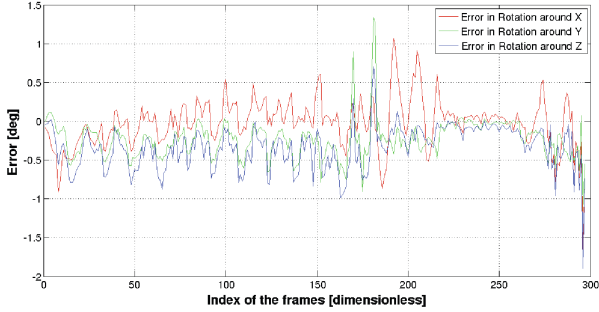
For comparison purposes, we use the information given in [7] about the RMSE in camera position for several localization systems. Four datasets were



(a) Error in camera position for 60 fps and 60 fps-blur datasets



(b) Error in rotations around X, Y, and Z for perfect ray-traced images (60 fps dataset)



(c) Error in rotations around X, Y, and Z for images with motion blur and noise (60 fps-blur dataset)

Fig. 6. Error in camera pose (position and orientation). The keyframe positions are drawn as blue points in (a) (Color figure online)

used; two real sequences from the TUM RGB-D benchmark [22] and two simulated sequences from [9]. These two synthetic datasets, called sim/desk and sim/slowmo, and the ones that we use in this work, 20 fps, 60 fps and 60 fps-

blur, have the same author, A. Handa, so we compare the accuracy using the information of sim/slowmo. Our system, with a RMSE of 0.41 cm for the 60 fps dataset, outperforms the semi-dense mono VO [8] that has a RMSE 2.51 cm. However, our system has inferior accuracy with respect to LSD-SLAM [7] and direct RGB-D SLAM that have 0.35 cm and 0.13 cm, respectively. It may be due to the use of a probabilistic approach that incorporates information about noise on the depth maps into tracking and the use of a framework for graph optimization.

The dense map was built offline with point cloud library 1.7.0 and using depth maps of keyframes. It is shown in Fig. 7. Note that all the local clouds, after being referenced to a global coordinate system, fit tightly.



Fig. 7. Dense map built using depth maps of keyframes. The trajectory is drawn in orange and keyframes are blue spheres. (Color figure online)

5 Conclusion

We have developed a system that estimates the pose of a range camera in 6DOF in a small environment. Our goal is to use this system in augmented reality applications. The proposed algorithm uses a 3D-2D visual odometry technique for computing an initial estimate that is later refined using image alignment. In order to reduce the computational cost, the current camera pose is computed with respect to the closest keyframe stored in the system. Since the localization and mapping algorithms run on different threads, they can be designed and implemented independently and later, they can be joined again to form a more general system. With this goal in mind, in this paper, we have focused on improving the performance of the dense localization algorithm in terms of time and accuracy. The results have been more than satisfactory:

First, using a coarse-to-fine scheme we have improved the convergence of the system, requiring just two iterations of the Levenberg-Marquardt algorithm at the lowest level of our four level pyramid; As a result, we have boosted the performance of the system achieving to process 10fps implementing most of the steps on a GPU.

Second, using the proposed two step process to localize the camera with respect to a keyframe –3D to 2D followed by the image alignment- results in high accuracy estimates even when using blurred input images. In the worst case -with blur and noise-, the RMSE is 1.12 cm in translation and 0.31° , 0.34° , and 0.43° in the rotations around X, Y and Z axis respectively.

For future work, we will concentrate on the depth map creation, depth map fusion, and the implementation of the whole system on a GPU.

Acknowledgments. The authors would like to thank Fundación CEIBA for the financial support that has made the development of this work possible.

References

1. Baker, S., Matthews, I.: Lucas-Kanade 20 years on: a unifying framework: Part 1. *Int. J. Comput. Vis. (IJCV)* **3**(56), 221–255 (2004)
2. Bylow, E., Sturm, J., Kerl, C., Kahl, F., Cremers, D.: Real-time camera tracking and 3D reconstruction using signed distance functions. In: *Robotics: Science and Systems Conference (RSS)*, June 2013
3. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vis.* **1**(40), 120–145 (2011)
4. Civera, J., Davison, A.J., Montiel, J.M.M.: Inverse depth parametrization for monocular SLAM. *IEEE Trans. Robot.* **24**(5), 932–945 (2008)
5. Davison, A.J.: Real-time simultaneous localisation and mapping with a single camera. In: *Proceedings of the International Conference on Computer Vision, Nice*, October 2003
6. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: MonoSLAM: real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(6), 1052–1067 (2007)
7. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) *ECCV 2014*. LNCS, vol. 8690, pp. 834–849. Springer, Heidelberg (2014). doi:[10.1007/978-3-319-10605-2_54](https://doi.org/10.1007/978-3-319-10605-2_54)
8. Engel, J., Sturm, J., Cremers, D.: Semi-dense visual odometry for a monocular camera. In: *2013 IEEE International Conference on Computer Vision (ICCV)*, pp. 1449–1456, December 2013
9. Handa, A., Newcombe, R.A., Angeli, A., Davison, A.J.: Real-time camera tracking: when is high frame-rate best? In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) *ECCV 2012*. LNCS, vol. 7578, pp. 222–235. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-33786-4_17](https://doi.org/10.1007/978-3-642-33786-4_17)
10. Hartley, R.I., Zisserman, A.: *Multiple View Geometry in Computer Vision*, 2nd edn. Cambridge University Press, Cambridge (2004). ISBN 0521540518
11. Klein, G., Murray, D.W.: Parallel tracking and mapping for small AR workspaces. In: *Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*, November 2007

12. Klein, G., Murray, D.: Improving the agility of keyframe-based SLAM. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008. LNCS, vol. 5303, pp. 802–815. Springer, Heidelberg (2008). doi:[10.1007/978-3-540-88688-4_59](https://doi.org/10.1007/978-3-540-88688-4_59)
13. Lovegrove, S., Davison, A.J.: Real-time spherical mosaicing using whole image alignment. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6313, pp. 73–86. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-15558-1_6](https://doi.org/10.1007/978-3-642-15558-1_6)
14. Newcombe, R.A., Davison, A.J.: Live dense reconstruction with a single moving camera. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1498–1505, June 2010. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5539794>
15. Newcombe, R.A., Davison, A.J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., Fitzgibbon, A.: KinectFusion: real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136, October 2011. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6162880>
16. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: dense tracking and mapping in real-time. Department of Computing, Imperial College London, UK (2012)
17. Piniés, P., Paz, L.M., Gálvez-López, D., Tardós, J.D.: CI-graph simultaneous localization and mapping for three-dimensional reconstruction of large and complex environments using a multicamera system. *J. Field Robot.* **27**(5), 561–586 (2010)
18. Silveira, G., Malis, E., Rives, P.: An efficient direct method for improving visual SLAM, 10–14 April 2007
19. Steinbrucker, F., Sturm, J., Cremers, D.: Real-time visual odometry from dense RGB-D images. In: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pp. 719–722 (2011). doi:[10.1109/ICCVW.2011.6130321](https://doi.org/10.1109/ICCVW.2011.6130321)
20. Stewénius, H., Engels, C., Nistér, D.: Recent developments on direct relative orientation. *ISPRS J. Photogramm. Remote Sens.* **60**, 284–294 (2006)
21. Stühmer, J., Gumhold, S., Cremers, D.: Real-time dense geometry from a handheld camera. In: Gesele, M., Roth, S., Kuijper, A., Schiele, B., Schindler, K. (eds.) DAGM 2010. LNCS, vol. 6376, pp. 11–20. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-15986-2_2](https://doi.org/10.1007/978-3-642-15986-2_2)
22. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the International Conference on Intelligent Robot Systems (IROS), October 2012
23. Varadarajan, V.S.: Lie Groups, Lie Algebras, and Their Representations. Graduate Text in Mathematics, vol. 102. Prentice-Hall, Englewood Cliffs (1974)

Robotics

12th Latin American Robotics Symposium and Third
Brazilian Symposium on Robotics, LARS 2015/SBR 2015,
Uberlândia, Brazil, October 28 - November 1, 2015,
Revised Selected Papers

Osorio, F.; Sales Goncalves, R. (Eds.)

2016, XII, 299 p. 160 illus., Softcover

ISBN: 978-3-319-47246-1