Online frame-to-model pipeline to 3D reconstruction with depth cameras using RGB-D information

Thiago Dornelles and Claudio Rosito Jung
Institute of Informatics – Federal University of Rio Grande do Sul
Emails: thiago.azevedo87@gmail.com, crjung@inf.ufrgs.br

***Abstract*—This work presents an online pipeline for incremental 3D reconstruction and 6-DoF camera pose estimation based on colored point clouds captured by consumer RGB-D cameras. The proposed approach combines geometric matching provided by the point cloud with photometric matching provided by the color sensor through an adaptive weighting scheme that avoids eventual misalignment errors between RGB and depth data. Our experimental results indicate that the 3D reconstructions achieved by the proposed scheme are visually better or similar than a competitive approach.**

I. Introduction

Three Dimensional reconstruction is the process of estimating the 3D structure (shape and appearance) of an object, which can be achieved using various methods. We can use a set of color captures of the subject from different views (which is te core of Structure-from-Motion – SfM – in Computer Vision) or a set of point clouds acquired from different views (using a laser scan or time of flight camera, for instance). Now we have a large availability of low-cost RGB-D sensors, making 3D scanning software a popular application for the maker culture. These RGB-D sensors capture simultaneously color (RBG) and depth (D) information, generating a colored 3D point cloud as output. Some of these sensors focus on near-range applications, being useful to build natural user interfaces, 3D object scanning and face recognition, for example. Some of them are able to capture longer-range depth values, being designed to build applications that make a wider perception of the environment like autonomous cars, robot navigation and scene reconstruction, to name a few applications.

Although the mathematical formulation for shorter- or longer-range applications is the same, the nature of captured scenes vary. For example, algorithms for 3D alignment that use mid-range sensors (2 to 4 meters) focus on the reconstruction of larger scenes (e.g., full indoor environments), such as [1]. In these cases, the sensor is able to capture several 3D objects at different locations and orientations (e.g., walls, chairs, tables), so that many point and depth correspondences across different frames can be obtained. Also, the full Field-of-View (FoV) of the sensor might contain scene objects, and it was recently shown in [2] that using features that are spatially spread tends to produce more accurate results in the context of epipolar matrix estimation. On the other hand, scanning smaller objects require near-range sensors (0.5 meters to 1 meter) to obtain enough geometric and textural details. In this scenario, the captured image contains mostly the object (typically in the central part of the camera FoV) and some background, leading to limited geometry/color/texture variability. Hence, finding correspondences across different views is a more challenging task, which compromises both camera pose estimation and 3D reconstruction.

In this paper, we present a 3D reconstruction algorithm for temporally continuous RGB-D data suited for near-range sensors. The core of our approach is to take advantage of geometric alignment (depth data) when there is low color information, and explore color matching to compensate for low geometric information (e.g. in locally planar regions). In this context, errors in the alignment of depth and RGB images may lead to the estimation of erroneous correspondences, especially in the regions of discontinuity or with a wide inclination concerning the camera axis. To lessen these problems and the inherent drift of the alignment process, we use a weighted correspondence scheme that considers normal map information to choose which point correspondences are reliable to the alignment process. Additionally, an optional loop closure process can be executed at the end of the process in order to achieve global consistency in the final trajectory if it is applicable.

II. Related work

Various approaches have been proposed to address the 3D reconstruction problem; most of them depend on reliable point matching between adjacent frames to estimate a rigid transformation that describes camera movement between these frames. This review will focus on methods that explore point-cloud data (with or without associated color information), and direct the reader to [3] for a survey on surface reconstruction from point clouds.

The classical technique called Iterative Closest Point (ICP) [4] consists of establishing initial correspondences between point clouds, and then iteratively finding a rigid transformation that best matches each of the selected pairs of points belonging to the input point clouds. Several variants of this algorithm have been proposed, such as the point-to-point formulation [4], in which a simple closed formulation is used to iteratively minimize the distance between points. To improve convergence, the minimization of a point-to-plane formulation [5] is used with non-linear solving methods. More recently, Serafin and Grisetti [6] use a series of criteria to prune bad correspondences, using normal and curvature information while uses normal and tangent information to generate an extended measurement of each point correspondence that improves the convergence of the minimization function.

Other works use 2D [7] or 3D [8] feature descriptors to find a global transformation to align point clouds. These algorithms require some pre-processing to obtain discriminative points and generate its descriptors before finding matches between these features. Errors on this matching process can lead to gross errors on final alignment. To overcome this problem, techniques like RANSAC try to cope with spurious matches (outliers) and can be used to minimize its influence, but unfortunately, it can increase algorithm complexity.

Direct methods using dense RGB-D information such as [9] and [10] aim to find the optimal alignment between the current frame and a warped reference frame minimizing a photometric error function. In comparison to ICP, these approaches present more convergence issues when the displacements between frames get larger, and rolling shutter cameras are used. Nonetheless, coarse-to-fine alignment strategies can be implemented to alleviate this limitation. As an advantage, this class of algorithms can be easily parallelized being suited to real-time applications while keeping good accuracy.

Many other direct dense alignment algorithms have been proposed, such as [11], [12]. These algorithms use color and geometrical information like depth and normal maps to formulate the matching problem as an energy minimization function. In [11], an objective function is built combining photometric and geometric error, while Corte et al. [12] present a general framework that uses information from range maps, depth maps, normal maps and color images.

The approaches presented in [13] and [14] rely only on depth maps, and the alignment occurs in a frame-to-model fashion where incoming frames are incrementally integrated into a reference model. The well-known Kinect Fusion [13] uses a volumetric data structure called Truncated Signed Distance Field (TSDF), which builds in real-time a mesh of the reference model, filtering data that fall inside voxel regions giving a better appearance to the final reconstruction. Similarly, Keller and his colleagues [14] build an incremental fused point cloud as the reference model and projects this with splatting rendering to generate a depth map to align with new depth frames. It is observed in [15] that a frame-to-model pipeline drastically reduces drift.

The use of colored point clouds is attractive because the two modalities can be used in a complementary manner: if the object lacks geometry, color can help; conversely, geometric information can be useful when the object lacks textural information. However, colored point clouds in off-the-shelf cameras are acquired by two different sensors, and small errors in the camera calibration parameters might generate a wrong mapping of color and geometric information. This problem is particularly challenging when using near-field depth cameras, since objects closer to the camera are more prone to such misalignment. This work presents a 3D reconstruction approach that explores both color and depth information, similarly to [11]. As contributions, we explicitly handle the misalignment problem by introducing a weighted scheme based on normal map information. Moreover, we implement a frame-to-model TSDF based scheme instead of frame-to-frame used on [11] in order to reduce camera drift and filter points of the model while are removed the noisy point insertions during the online model construction. The proposed approach is presented next.

III. The proposed approach

We propose a pipeline that can build 3D models on-the-fly, fusing the acquired point cloud into a refined consolidated model, considering that at first, the consolidated model is just the first acquired point cloud. We use the current pose to project the model, creating a depth map at time t that will be used to find the next camera pose at time $t+1$ using a weighted combination of photometric and geometric features. An overview of the proposed approach is illustrated in Fig. 1, and each step is detailed next.

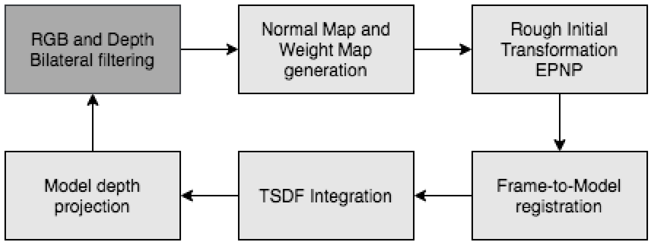
RGB and Depth Bilateral filtering
Normal Map and Weight Map generation
Rough Initial Transformation EPNP
Frame-to-Model registration
TSDF Integration
Model depth projection

Fig. 1. Proposed iterative pipeline

A. Color and Depth Bilateral filtering

Cheap RGB-D Cameras have very limited image quality, particularly when it works under low illumination conditions, for example, as the case of indoor photographing many artifacts like salt and pepper noise and the Gaussian noise appear when we take photos. In the case of depth capturing, often pixels are obtained with bad depth estimates, which are undesirable both to the registration process and model integration process. To improve quality with algorithm complexity of the final model, normal map estimation and get better results at photometric/depth alignment, we perform a fast bilateral filtering [16] on input RGB and depth images. This filter smoothes noise of depth measurements and RGB pixels, while preserving the edges of the original image based on two parameters: σ_s to control spatial smoothing and σ_r controls range smoothing.

B. Normal Map and Weight Map Estimation

Although the pipeline presented in this work is generic for any point cloud capture device, we note that devices that explore Time of Flight (ToF) or structured light present limitations when the scanned surface is locally orthogonal (or close to orthogonal) to the emitting rays. In these cases, the amount of reflected light back to the sensor (in the case of ToF cameras) is small, so that the depth estimate (and consequently the corresponding 3D camera coordinates) might be very wrong, generating "flying pixels" that are disconnected from the object.

To avoid generating point cloud models with flying pixels or similar artifacts, we first prune the acquired point cloud using a pre-processing step based on the surface normals. More precisely, we compute the angular distance $\theta(n_c, n_p)$ between the local normal vector n_p at each point cloud location and the camera axis vector n_c, given by

$$\theta(n_c, n_p) = \arccos < n_p, n_c >, \tag{1}$$

where $< \cdot, \cdot >$ denotes the inner product. If the angle is sufficiently large (based on a threshold T_θ), the point is removed from the point cloud.

C. Rough Initial Registration based on PnP

Although we assume a temporally continuous capture of the object, rapid camera (or object) motion might generate adjacent point clouds that are significantly apart from each other. Since our fine alignment scheme (that will be described next) might not converge in such cases, we first perform a rough alignment of the point clouds.

Although we can use other geometric descriptors such as SHOT [8], in this work we opted to use image-based features instead. More precisely, we selected ORB image features [17] because they showed to provide more stable matchings when analyzing objects at a near distance – which is the case of this work – than geometric features. To estimate the 3D position of extracted keypoints from the reference view, we unproject the keypoints to 3D space using the reference depth map frame. Assuming that the depth camera is calibrated (which is the case, since this information is used to fuse RGB and depth values), we end up with the classical Perspective-n-Point (PnP) problem. From the several existing solutions, we chose the EPnP algorithm [18] due to its robustness and speed, also including a RANSAC-based outlier removal scheme. We check the validity of each transformation using a criterion that verifies if the transformation matches a minimum of 99% of the inlier points. We define a point as inlier when the 2D image distance between a point and its matching point is below 100 pixels, and the descriptor distance between them is lower than 80% of the distance of the second-best matching point. Otherwise, we choose to use the identity matrix as the initial transformation.

D. Refined Registration

After the initial pose estimation provided by EPnP, we fine-tune the estimate by solving a convex optimization problem that aims to minimize intensity and depth differences between pixels of each pair of the RGB-D images in two consecutive video frames. The goal of the optimization problem is to obtain optimal parameters of a rigid transformation matrix T that describes the 6-DoF synthetic camera pose that renders the best alignment between two colored point clouds: the first one is the current model, rendered by a synthetic RGB-D camera at the location of the last camera pose, and the second one is the RGB-D frame currently being captured by the sensor. We find the optimal rigid transformation between these two frames using a fine-tuned Gauss-Newton algorithm over a joint photometric and geometric error function.

Following [11], we have the photometric and geometric errors combined in the same objective function. As in their work, our algorithm receives as input a pair of registered RGB-D images (I_i, D_i), (I_j, D_j) and a transformation T that roughly aligns the images, obtained from ePnP.

1) Photometric Error: The Photometric Error E_P is formulated as the sum of squared intensity differences, as in [9]–[11]. As argued by these authors, the choice of combining RGB channels in one gray scale value reduces the computational cost without significant loss of accuracy. In this work, we consider findings of [19]–[21] and introduce a sensor independent weighting function $w_P(q)$ for each pixel at the location q that aims to reduce the importance of residuals which the depth estimate might be degraded. As this depth accuracy degradation might produce erroneous RGB to Depth matching, we prefer to give progressively less weight to the color pixels that are aligned with poor depth estimates.

The photometric error E_P is given by

$$E_P(T) = \sum_{p,q} w_P(q) \left(I_i(p) - I_j(q)\right)^2 , \tag{2}$$

where $p = (u, v)^T$ is a pixel in the registered (I_i, D_i) image pair and $q = (u', v')^T$ is its correspondent pixel in the (I_j, D_j) image pair. This correspondence is found by unprojecting q back to the 3D space (using pixel location, depth estimate and camera intrinsic parameters), applying the transformation T and re-projecting it into pixel coordinates of the (I_j, D_j) pair:

$$p = \pi_{uv}(s(\pi^{-1}(D_j(q)), T)), \tag{3}$$

Where s is the function that applies rigid transformation to the homogeneous points according to:

$$s(x, T) = Tx. \tag{4}$$

Also, the inverse point projection π^{-1} from depth map to homogeneous 3D point is defined as:

$$\pi^{-1}(u, v, d) = \left(\frac{d(u - c_x)}{f_x}, \frac{d(v - c_y)}{f_y}, d, 1\right)^\top , \tag{5}$$

where f_x and f_y are focal lengths and (c_x, c_y) is the principal point coordinate of the camera. The function π maps a 3D point $w = (w_x, w_y, y_z, 1)^\top$ into image coordinates u, v plus an additional coordinate d to store depth information based on how much a point is away from the sensor.

$$\pi(w_x, w_y, w_z) = \left(\frac{w_x f_x}{w_z} + c_x, \frac{w_y f_y}{w_z} + c_y, w_z\right)^\top , \tag{6}$$

To extract information from images we defined two convenient the functions π_{uv} and π_d. π_{uv} returns the first two coordinates of π which are the pixel coordinate on the image plane (I_i, D_i) and π_d returns only the last coordinate which stores the depth of the point.

2) Geometric Error: The Geometric Error E_G is obtained in a similar way to the ICP point-to-plane formulation of [5], except that we chose using only the third component from normals $N_i(p)$ and 3D points of depth maps D_i and D_j.

We are first intending to align confident points from planes that are parallel to the camera's image plane while avoiding alignment of noisy points that lies on planes that are almost orthogonal to the image plane. As $n_z = \cos(\theta)$, we are giving greater weighting to the fronto-parallel tangent planes of the incoming point cloud.

Differently from the proposed photometric error modelling, here we can easily measure how much the values of the correspondent points change as the camera moves and the difference must be calculated using values from π_d and $D_i(p)$, where π_d is the warped depth value of $W_j(q)$. As in the photometric error we also use a weighting scheme, and define

$$W_j(q) = s(\pi^{-1}(D_j(q)), T)) \tag{7}$$

$$E_G(T) = \sum_{p,q} w_G(p) N_i(p) \left(D_i(p) - \pi_d(W_j(q))\right)^2, \tag{8}$$

where $w_G(p)$ is a weighting function for the geometric error. In this work we used $w_G = w_P$, and this common weighting function will be detailed in section III-D3.

3) Point weighting: As discussed in III-A, time of flight and structured light depth cameras are more accurate when scanning fronto-parallel regions, since the amount of reflected/captured light is larger. The previous statement is confirmed in the case of Kinect v1 and v2 sensors as presented in the work of [19], [22]. Besides using this information for removing potentially bad points, we also explore it for weighting the global photometric and geometric distances. More precisely, the largest weight should be assigned to points that present tangent planes fronto-parallel to the camera sensor, and progressively less weight as the tangent planes become closer to fronto-orthogonal. This behavior is captured by the angle $\theta(n_c, n_p)$ between the camera viewing direction n_p and the normal vector n_p to the tangent plane of the point p under consideration. Our weighting function w_P is given by

$$w_P(p) = \max\{0, \cos\left(\sigma\theta(n_c, n_p)\right)\}, \tag{9}$$

where σ is a parameter that controls how fast the weight decreases as the angle θ increases, so that θ values above $\pi/(2\sigma)$ lead to null weights (i.e., the corresponding points are discarded from the analysis). We selected $\sigma = 1.3$ based on experiments and by the strong indication of works [19], [22] that most of Z-axial error (depth measurement) occurs when tangent plane angle varies from 70° to 90° with respect to the camera pointing vector.

It is important to note that the use of $w_P(p)$ also helps with another common issue when dealing with RGB-D cameras: color bleeding. Note that depth and color information is captured by different calibrated sensors, and later fused together based on the (intrinsic and extrinsic) parameters of each sensor. Since these parameters typically present some error, the registration of the two sensors is not perfect. In particular, the misalignment is more noticeable along depth discontinuities (which typically arise on the boundaries between two objects). At these locations, the colors of one object might be wrongly projected to the other object, which could lead to inconsistencies between E_P and E_G. Since depth discontinuities tend to generate a front-orthogonal local tangent plane, its effect is implicitly alleviated by our weighting function.

4) Optimization of Joint Error: We previously presented individual photometric and geometric errors, but we seek a solution where both objective functions are minimized as much as possible. For that purpose, we build a joint objective function that combines these errors through a weighted sum:

$$E(T) = E_P(T) + \lambda E_G(T), \tag{10}$$

where $\lambda \geq 0$ controls the relative weight of the geometric error. Here if we set $\lambda > 1$ we are favoring geometric alignment over the photometric alignment.

To represent the 6-DoF with a minimal representation we use the twist vector $\xi = [\omega_x, \omega_y, \omega_z, t_x, t_y, t_z]^\top$ that combines rotational and translational motion parameters to describe a rigid motion. This twist vector has its Lie algebra correspondence in matrix form $\hat{\xi} \in \text{se}(3)$ as follows:

$$\hat{\xi} = \begin{bmatrix} 0 & -\omega_z & \omega_y & t_x \\ \omega_z & 0 & -\omega_x & t_y \\ -\omega_y & \omega_x & 0 & t_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

From this matrix we can obtain the 6-DoF transformation $T \in \text{SE}(3)$ matrix using the matrix exponential $T = \exp(\hat{\xi})$.

To find the best transformation to this non-linear objective function, we use the Gauss-Newton method since it provides fast convergence. To calculate the Jacobian for residuals of the projected RGB and Depth pixels r_P, r_G we need to compute Jr_P and Jr_G as shown below:

$$Jr_P = \frac{\partial r_P}{\partial \xi} = \frac{\partial r_P}{\partial x} \frac{\partial x}{\partial X} \frac{\partial X}{\partial \xi}, \tag{12}$$

$$Jr_G = \frac{\partial r_G}{\partial \xi} = \lambda \left[N_r \frac{\partial X}{\partial \xi} \right], \tag{13}$$

where $\frac{\partial r_P}{\partial x}$ is the image gradients from the gray intensity image obtained from the original RGB image, N_r is the pixel associated normal vector obtained from the depth map using the technique described on paper of Badino et al. [23], $\frac{\partial x}{\partial X}$ is the perspective projection derivatives, $\frac{\partial X}{\partial \xi}$ is the 6-DoF motion derivative of 3D points. We then define

$$J_\xi = (Jr_P, Jr_G)^\top, \quad r = (r_P, r_G)^\top, \tag{14}$$

and we use these variables to solve the following linear system in order to obtain motion increments $\Delta\xi$:

$$J_\xi^\top J_\xi \Delta\xi = -J_\xi r. \tag{15}$$

To check if we have found a new motion increment that minimizes the objective function, we update the matrix $\hat{\xi}$ to calculate the rigid motion matrix $T_t = \exp(\hat{\xi})$ that aligns the model with the current frame. This process is repeated until the objective function no longer decreases its value or has reached to the maximum of iterations allowed. The pipeline keeps an updated matrix $\tilde{T}_{t-1}$ containing the last pose of camera until this last estimate. By concatenating the last pose matrix and the inverse of the last found transformation T_t^{-1} we get the new actual camera pose matrix:

$$\tilde{T}_t = \tilde{T}_{t-1} T_t^{-1} \tag{16}$$

E. TSDF Integration and Model generation

To incrementally build the mesh and use it as a smooth reference model, we use a type of Signed Distance Function [24] and integrate point cloud data into a Truncated Signed Distance Function (TSDF) as in KinectFusion [15]. The core idea of this step is to accumulate the measured distances between points that fall into the same voxel space with a weighted running average given by:

$$D_{i+1}(p) = \frac{W_i(p) D_i(p) + w_{i+1}(p) d_{i+1}(p)}{W_i(p) + w_{i+1}(p)}, \tag{17}$$

$$W_{i+1}(p) = W_i(p) + w_{i+1}(p), \tag{18}$$

where $D_i(p)$ is the accumulated average of signed distance of the point p to the centroid c_i of the voxel that contains the point, $W_i(p)$ is a function that returns the previous voxel accumulated weight, $w_{i+1}(p)$ is defines the incremental weight (set to 1). The term $d_{i+1}(p)$,is the new point distance to surface scaled by the truncation value μ that will be accumulated into the same voxel, given by

$$d_{i+1}(\mathbf{p}) = \begin{cases} \operatorname{sgn}(\phi_i(\mathbf{p})) & \text{if } |\phi_i(\mathbf{p})| > \mu \\ \phi_i(\mathbf{p})/\mu & \text{otherwise} \end{cases} \tag{19}$$

where

$$\phi_i(p) = \pi(c_i) - \pi(p) \tag{20}$$

presents a signed distance between the point and the voxel centroid.

At the very first integration of a point cloud to the TSDF structure we initialize all the with voxels with $W_i(p) = 0$ and $W_{i+1}(p) = 1$ and the update voxels where the incoming points falls with $d_{i+1}(p)$ function. Finally to extract the surface from the TSDF voxels at every iteration, the classical Marching Cubes algorithm [25] is used.

F. Model projection

The core of the proposed approach described in section III-D requires a pair of color and a pair of depth images (at adjacent frames). Although all these four images are provided by the RGB-D camera, we adopted a frame-to-model alignment.

More precisely, we synthetically generate the depth image D_i by projecting the model from the last pose tracked of the camera, rendering only the closest visible points of the model concerning the current camera position. This synthetic generated depth image is smoother and contains more reliable points than we usually have with depth images acquired from depth sensors, which improves the alignment process avoiding measurement errors. On the other hand, RGB images obtained by synthesizing a view from the current 3D model are not adequate because they tend to accumulate errors due to color bleeding when RGB and depth present some misalignment. Hence, we use a hybrid approach by selecting the color images directly from the sensor and extracting the depth image from the consolidated model.

IV. Experimental Results

We implemented our algorithm in C++11, with the help of the libraries `OpenCV` for image manipulation and `Open3D` for 3D visualization and TSDF data structures. All tests were run on a PC running Linux Mint 18.3 in an Intel i5-7400 processor with 8 GB of RAM. All the pipeline runs only on CPU, except for visualization tasks that explore GPU processing.

To evaluate the accuracy of our reconstruction approach, we performed tests using the 3D-Printed RGB-D Object Dataset introduced in [26]. This dataset consists of five different 3D-printed objects that were scanned using RGB-D sensors of different quality and with two types of scanning camera motions: stopped camera pointing to the object on a turntable and a handheld camera motion (more erratic) around the object. Here we use only the Kinect recorded scenes of the dataset because only these can emulate an application using off-the-shelf cameras, noting that the phase shift and synthetic data are not suitable to the scope of this work.

A. Parameter setting

We set $\lambda = 5$ in Equation (10), favoring a tight depth alignment while using the color information to disambiguate point matchings on planar regions of depth data. Larger values of λ tend to make the error function behave like the common ICP distance error. Conversely, lower λ values tend to preserve photometric errors at lower levels but not necessarily with a proper depth alignment. This value for λ was found based on empirical tests covering all the scenes of the used dataset. For the edge-aware smoothing using the bilateral filter, we selected $\sigma_s = 3$ and $\sigma_r = 5$, also based on experiments.

To remove possible flying points we use a conservative pruning angle value $T_\theta = 70°$ was chosen for Equation (1), which might also remove some good points. However, we noted that removing a few good points at some frames is better than keeping bad estimates, since missed good points tend to appear again (at a better angle) during other frames of the capture.

B. Quantitative evaluation

1) Camera trajectory (pose): The Relative Pose Error (RPE) measures local accuracy of the camera trajectory estimate, whereas the Absolute Trajectory Error (ATE) captures the global accuracy of the camera pose estimates comparing the ground truth camera pose and the estimated camera pose [27]. We computed the well-known root-mean-square error (RMSE) using these two metrics (ATE and RPE) for the results produced by our method and [11], which is implemented in the Open3D library (from this point we will refer to Open3D as the Colored Point Cloud registration module of the library). For a fair comparison, Open3D is used as a substitute for the "Frame-to-Model Registration" step of our algorithm presented in Fig. 1. As shown in Table I, our algorithm consistently yields better RPE results than Open3D, probably because of our weighting point scheme that gives lower confidence to points that are tilted away from the camera image plane. This effect is especially noticeable in scenes that contain more sudden movements, i.e., those made by handheld motion. We have also included our results without the proposed weighting scheme, named "W/o weight" in Table I. As can be observed, the weighting scheme reduces the average RPE RSME error in about 10%. The ATE differences are smaller for some datasets, such as for "Teddy Turntable", but in others, the results produced by our approach presents a considerably smaller error. For instance, our error for the "Bunny Turntable" dataset was one-third of the error obtained by Open3D. We can also observe that using the weighting scheme increases the ATE RMSE error for some datasets, but on average, it is better than the version without weighting.

TABLE I
RPE ($\times 10^3$)/ATE ($\times 10^2$) RMSE RESULTS

Dataset	Open3D	Proposed	W/o weight
Teddy Turntable	1.66 / 1.39	**0.60** / 1.29	0.61 / **1.13**
Bunny Turntable	9.58 / 2.80	**0.57** / 0.81	0.58 / **0.79**
Kenny Turntable	2.88 / 1.39	**0.60 / 1.15**	0.62 / 1.16
Tank Turntable	3.61 / 1.22	**0.56 / 1.12**	0.57 / 1.13
Leopard Turntable	2.98 / 1.24	**0.62 / 1.10**	0.65 / 1.11
Teddy Handheld	12.05 / 1.83	**7.34 / 1.16**	7.44 / 1.30
Bunny Handheld	10.34 / 1.52	**4.79 / 0.99**	7.50 / 1.30
Kenny Handheld	6.75 / 0.90	**1.60 / 0.75**	2.20 / 1.02
Tank Handheld	10.82 / 2.21	6.83 / 1.49	**5.38 / 1.41**
Leopard Handheld	10.57 / 1.87	**5.76 / 0.92**	7.40 / 1.21
Average	7.12 / 1.64	**2.93 / 1.08**	3.29 / 1.15

2) 3D Reconstruction: We compare the quality of the generated models by comparing the 3D point distances of the obtained models, and the corresponding "ground-truth" model generated using ground truth pose data. By using the Cloud to Mesh (C2M) tool provided by CloudCompare [28], we can obtain a histogram of distance differences, which is calculated with a point cloud and a mesh: one is the ground truth mesh, and the other is a point cloud sampled from the mesh created with our pipeline. Each of the points of the cloud has its distance calculated with respect to the closest triangle from the ground truth mesh, and these distances are accumulated in a histogram. We can obtain a measure of the mesh fidelity by computing the mean value and standard deviation of the error histograms, for instance.

We summarize the results in table II, using the average of absolute distance error (note that distances might be positive or negative, depending if a given point is inside or outside the mesh) and standard deviation of the signed distance error. The first one is used to verify if the generated mesh presents its points close to the ground truth mesh, which might indicate if the mesh has as global shift with respect to the geometry of the ground truth mesh. The standard deviation with the signed distance error measures the deviation from the error average: if it is large, we can expect more deformations in the final generated mesh. All of these measurements assume that two meshes are sufficiently well-registered to get the minimal distance between the correspondence points.

TABLE II
C2M MEAN $\pm$ STD. DEV. SCALED BY A FACTOR OF 10^4

Dataset	Open3D	Proposed	W/o weight
Teddy T.	13.30 ± 21.53	$\mathbf{11.54} \pm \mathbf{18.71}$	12.50 ± 19.85
Bunny T.	90.39 ± 119.07	$\mathbf{11.66} \pm \mathbf{17.56}$	11.97 ± 18.40
Kenny T.	$\mathbf{2.64} \pm 14.47$	$8.64 \pm \mathbf{14.15}$	10.14 ± 17.66
Tank T.	$\mathbf{5.65} \pm \mathbf{11.01}$	10.20 ± 17.87	10.58 ± 19.37
Leopard T.	$\mathbf{7.85} \pm \mathbf{14.27}$	11.49 ± 18.70	12.07 ± 20.04
Teddy H.	33.68 ± 65.37	$\mathbf{9.48} \pm \mathbf{18.05}$	9.75 ± 19.11
Bunny H.	12.58 ± 30.53	$\mathbf{6.20} \pm \mathbf{9.71}$	7.22 ± 11.55
Kenny H.	$\mathbf{7.17} \pm 17.98$	12.30 ± 20.61	$7.72 \pm \mathbf{14.21}$
Tank H.	16.03 ± 30.90	$\mathbf{4.93} \pm \mathbf{8.60}$	7.90 ± 12.90
Leopard H.	16.94 ± 30.69	$\mathbf{8.35} \pm \mathbf{16.88}$	18.78 ± 44.66
Average	20.62 ± 35.58	$\mathbf{9.48} \pm \mathbf{16.08}$	10.86 ± 19.77

The table II indicates that the use of the weighting scheme provides consistently better results than not using it (both in terms of the mean and standard deviation). Also, our full approach (with weighting) presented smaller both average error and standard deviation than the baseline considering all datasets (see the last row of the table). In particular, the improvement is more noticeable in the handheld sequences, which indicates that our method can better handle erratic camera motion.

C. *Qualitative evaluation*

Although C2M presents a quantitative way of comparing 3D models, visual inspection is paramount for identifying the introduction of possible artifacts. In particular, the preservation of geometric details and thin structures of the generated models is essential for evaluating if the reconstruction succeeded. Fig. 2 shows the GT model and the results produced by our approach and Open3D for three datasets, where the red rectangles highlight regions for which Open3D generated artifacts. Some artifacts are errors in geometry reconstruction, while others are observed as wrong color blending. It is also interesting to note that our reconstruction results were very similar to the GT model.

As additional visual results, we have also created datasets related to 3D body scans and compared our results with Open3D. Fig. 3 shows the results of a partial (top, 220 frames) and a full (bottom, 438 frames) body scan produced by the two methods. The results of both methods look coherent in the partial scan, but a closer inspection indicates that the shirt texture on the shoulder of the person was somewhat deformed by Open3D. In the full scan, Open3D was not able to provide good alignments in the back of the person, and we aborted the execution. Our approach, on the other hand, did not show such drifts. A handheld scanned scene with smaller objects is shown in Fig. 4. Although both results are visually good, Open3d

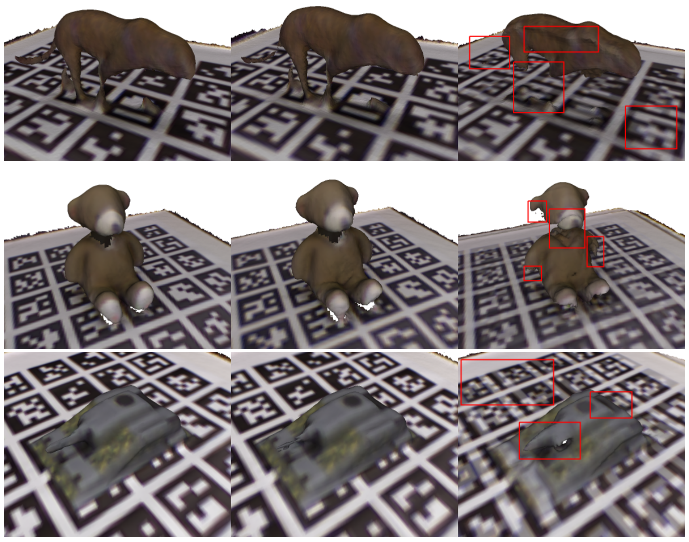
Fig. 2. Top to Bottom: Leopard Handheld, Teddy Handheld and Tank Handheld datasets. From left to right: GT model, Proposed and Open3D.

generates small artifacts in some geometrical structures, as highlighted by the red rectangles.

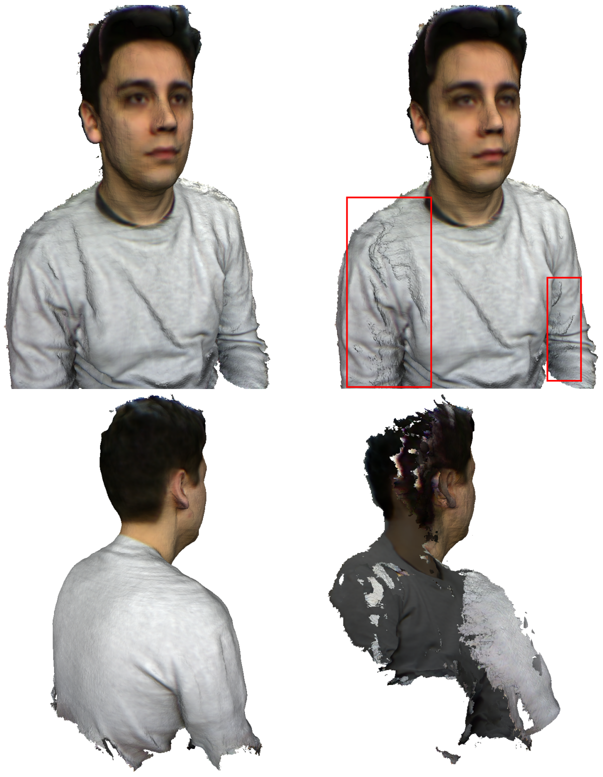
Fig. 3. Results of partial (top) and full (bottom) body scanning. Left: Proposed, Right: Open3D

D. Execution speed

Although the tested implementation is not optimized for parallel processing and did not explore GPU acceleration, we check the execution times for each pipeline step in order to check for possible processing bottlenecks. We confirm by measurements that time performance was affected mainly by the two components that had significant complexity: Bilateral Filtering, and RGB-D Alignment as shown in Table III. These two components are fully parallelizable, but for now, our implementation did not have this concern in mind. Open3D alignment step presents an average time of 856 milliseconds per iteration.

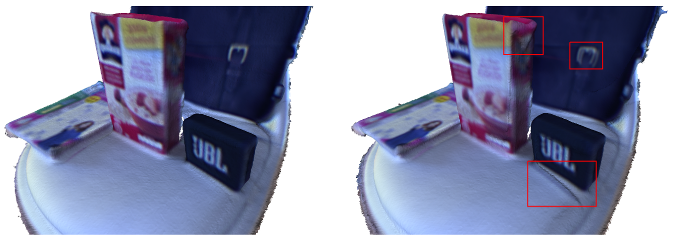
Fig. 4. Handheld motion scanning: Proposed vs Open3d

TABLE III
AVERAGE IN MILLISECONDS PER FRAME

Bilateral Filter	**Normal Map**	**Rough Init.**	**RGB-D Alignment**	**TSDF Integration**
1004	168	68	1045	251

E. Pose Graph Optimization Results

As a final experiment, we evaluated the effect of adding a post-processing scheme based on loop closure, which is adequate when performing full scans. More precisely, we coupled the pose graph optimization algorithm [29] implemented in the `Open3D` library [30] to the obtained camera trajectory. Fig. 5 illustrates a visual example, indicating that the overall geometry of the meshes is improved with the post-processing step. Interestingly, the texture on the planar surface was blurred after the optimization step, which corroborates the hypothesis of the misalignment of color and depth sensors.

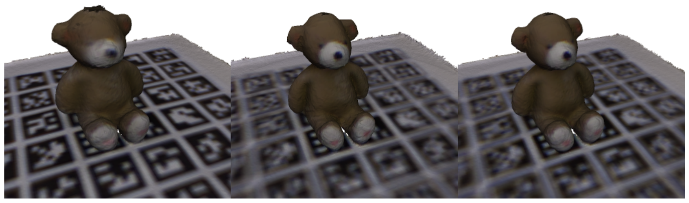
Fig. 5. Teddy Turntable: Proposed, Proposed + Pose Graph and GT

V. Conclusions and Future Work

We have presented an online 3D reconstruction pipeline based on RGB-D video sequences focused on near-range captures. The core of the proposed approach is an iterative pose alignment procedure that considers a weighted combination of color and depth images aiming to reduce the inherent noise of depth sensors that also accounts for possible misalignment between the color and depth sensors. As additional contributions, we perform a rough alignment based on PnP that is able to handle larger motion between adjacent frames of the sequence, and the use of a frame-to-model registration scheme that further reduces the influence of noise in the depth image,

Our experimental results indicate that both the pose and the obtained 3D models with our method are comparable to or better than a state-of-the-art method. A qualitative analysis (visual inspection) indicates that the proposed model is capable of keeping geometric texture and thin structures, and at the same time, avoids color bleeding artifacts that arise due to the misalignment of depth and color sensors.

As future work, the most immediate issue to address is code optimization to accelerate Bilateral filter and RGB-D alignment performance. In addition, the idea of weighting the correspondences could be improved through the use of machine learning techniques, by recognizing image areas that we could set larger weights in order to ensure the best alignment of point clouds. As an alternative to the projective correspondence implemented in this work, the nearest neighbor correspondence could be used as in the Open3D technique, possibly resulting in smoother alignments.

References

[1] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 76a, 2017.

[2] T. L. T. d. Silveira and C. R. Jung, "Perturbation analysis of the 8-point algorithm: A case study for wide fov cameras," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," in *Computer Graphics Forum*, vol. 36, no. 1. Wiley Online Library, 2017, pp. 301–329.

[4] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[5] K. lim Low, "Linear least-squares optimization for point-to-plane icp surface registration," Tech. Rep., 2004.

[6] J. Serafin and G. Grisetti, "Nicp: Dense normal based point cloud registration," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 742–749.

[7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[8] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 356–369. [Online]. Available: http://dl.acm.org/citation.cfm?id=1927006.1927035

[9] F. Steinbrucker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense rgb-d images," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)(ICCVW)*, vol. 00, Nov. 2012, pp. 719–722. [Online]. Available: doi.ieeecomputersociety.org/10.1109/ICCVW.2011.6130321

[10] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras." in *ICRA*. IEEE, 2013, pp. 3748–3754.

[11] J. Park, Q. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 143–152.

[12] B. D. Corte, I. Bogoslavskyi, C. Stachniss, and G. Grisetti, "A general framework for flexible multi-cue photometric point cloud registration," *CoRR*, vol. abs/1709.05945, 2017. [Online]. Available: http://arxiv.org/abs/1709.05945

[13] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '11. New York, NY, USA: ACM, 2011, pp. 559–568. [Online]. Available: http://doi.acm.org/10.1145/2047196.2047270

[14] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," in *2013 International Conference on 3D Vision - 3DV 2013*, June 2013, pp. 1–8.

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Oct 2011, pp. 127–136.

[16] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 24–52, Jan 2009. [Online]. Available: https://doi.org/10.1007/s11263-007-0110-8

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[18] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, 02 2009.

[19] C. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," 10 2012.

[20] K. Khoshelham and E. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," in *Sensors 2012, 12, 1437–1454. 2013*, p. 8238.

[21] T. Mallick, P. P. Das, and A. K. Majumdar, "Characterizations of noise in kinect depth images: A review," *IEEE Sensors Journal*, vol. 14, no. 6, pp. 1731–1740, June 2014.

[22] N. Grossmann, "Extracting sensor specific noise models," Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, aug 2017, 1. [Online]. Available: https://www.cg.tuwien.ac.at/research/publications/2017/grossmann-2016-baa/

[23] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3084–3091.

[24] B. Curless and M. Levoy, "A volumetric method for building complex models from range images." in *SIGGRAPH*, J. Fujii, Ed. ACM, 1996, pp. 303–312. [Online]. Available: http://dblp.uni-trier.de/db/conf/siggraph/siggraph1996.html#CurlessL96

[25] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm." in *SIGGRAPH*, M. C. Stone, Ed. ACM, 1987, pp. 163–169. [Online]. Available: http://dblp.uni-trier.de/db/conf/siggraph/siggraph1987.html#LorensenC87

[26] M. Slavcheva, W. Kehl, N. Navab, and S. Ilic, "SDF-2-SDF: Highly Accurate 3D Object Reconstruction," in *European Conference on Computer Vision (ECCV)*, 2016.

[27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 573–580.

[28] "Cloudcompare: 3d point cloud and mesh processing software," http://www.danielgm.net/cc/, accessed: December 10, 2019.

[29] Sungjoon Choi, Q. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 5556–5565.

[30] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.