sensors

MDPI

Article

Enhancement of RGB-D Image Alignment Using Fiducial Markers

Tiago Madeira [1,*], Miguel Oliveira [1,2] and Paulo Dias [1,3]

1 Institute of Electronics and Informatics Engineering of Aveiro, University of Aveiro, 3810-193 Aveiro, Portugal; mriem@ua.pt (M.O.); paulo.dias@ua.pt (P.D.)
2 Department of Mechanical Engineering, University of Aveiro, 3810-193 Aveiro, Portugal
3 Department of Electronics, Telecommunications and Informatics, University of Aveiro, 3810-193 Aveiro, Portugal
* Correspondence: tiagomadeira@ua.pt

Received: 6 February 2020; Accepted: 6 March 2020; Published: 9 March 2020

Abstract: Three-dimensional (3D) reconstruction methods generate a 3D textured model from the combination of data from several captures. As such, the geometrical transformations between these captures are required. The process of computing or refining these transformations is referred to as alignment. It is often a difficult problem to handle, in particular due to a lack of accuracy in the matching of features. We propose an optimization framework that takes advantage of fiducial markers placed in the scene. Since these markers are robustly detected, the problem of incorrect matching of features is overcome. The proposed procedure is capable of enhancing the 3D models created using consumer level RGB-D hand-held cameras, reducing visual artefacts caused by misalignments. One problem inherent to this solution is that the scene is polluted by the markers. Therefore, a tool was developed to allow their removal from the texture of the scene. Results show that our optimization framework is able to significantly reduce alignment errors between captures, which results in visually appealing reconstructions. Furthermore, the markers used to enhance the alignment are seamlessly removed from the final model texture.

Keywords: computer vision; geometric optimization; camera calibration; 3D reconstruction; texture; inpainting; fiducial markers; point clouds; projection of 3D points

1. Introduction

Three-dimensional (3D) reconstruction is the creation of 3D models from the captured shape and appearance of real objects. It is a field that has its roots in several areas within computer vision and graphics, and has gained high importance in others, such as architecture [1], robotics [2], autonomous driving [3], medicine [4], agriculture [5], and archaeology [6]. Most of the current model acquisition technologies are based on LiDAR (Light Detection And Ranging) [7,8], RGB-D cameras [9,10], and image-based approaches, such as photogrammetry [11,12]. Despite the improvements that have been achieved, methods that rely on professional instruments and operation result in high costs, both capital and logistical [13].

The introduction of low-cost RGB-D cameras created an opportunity for 3D reconstruction of scenes to be performed at consumer-level. The increasing popularity of these sensors promoted the research of their use to reconstruct 3D indoor scenes [14–16]. Their manoeuvrability, permitting hand-held operation and allowing the user to get closer to parts of the scene and capturing them from different angles, proves to be an advantage in indoor environments with high probability of object occlusion, when compared with fixed high resolution scanners, which usually require more space and fixed poses to operate. As such, even though they are generally less accurate than fixed scanners such as LiDAR, mobile or hand-held scanners are known to be more suitable to perform indoor scanning [17]. These devices are also compelling for applications such as indoor navigation, since the way they are operated inherently conveys information about the empty space, which often corresponds to the navigable space, as the device is carried through it, by a person or a robot for instance.

Indoor 3D models have great potential in object tracking and interaction [18], scene understanding [19], virtual environment rendering [20], indoor localisation [21] and route planning [22], amongst others. Given the rapid development of LBS (Location-Based Services) and indoor applications, fast acquisition and high-fidelity reconstruction of complete indoor 3D scenes has become an important task [23]. Currently there is extensive research conducted into live 3D reconstruction techniques, where the final model is built and visualized during the data acquisition process. However, post-processing techniques show significant potential and need for future research, particularly in 3D reconstruction using RGB-D cameras [24]. As such, it would be interesting to complement the existing live 3D reconstruction techniques that use low-cost RGB-D hand-held cameras, considering their advantages and placement at consumer-level in the market, with some automatic post processing, to see whether their results can be improved, creating significant additional value.

The main objective of this work is to develop an optimization procedure capable of enhancing the 3D reconstructions created using a hand-held RGB-D camera, through the utilisation of fiducial markers placed in the environment.

2. Related Work

RGB-D cameras can obtain the depth of an object and the corresponding texture information, through the combination of a depth image and a standard RGB image. The depth images are commonly obtained using a ToF (Time-of-Flight) depth sensor measuring system [25] or Structured Light triangulation [26,27]. ToF works by measuring the round trip time of an artificial light signal, in this case IR (Infrared Radiation), to resolve the distance between the camera and the target object. Structured light works by projecting known patterns on to the scene and then measuring the way they are deformed, allowing for the calculation of depth and surface information. Modern RGB-D approaches are mostly based on the fundamental research by Curless and Levoy [28] who introduced the work of volumetric fusion, providing the foundation for the first real-time RGB-D reconstruction methods [29].

Registration is the process of alignment, in the same coordinate frames, of multiple captures, with different viewpoints, of the same scene or object. The result may be an extended version of a two-dimentional (2D) image, such as a panoramic photograph, or a 3D representation of the scene. Within the context of 3D reconstruction, registration translates to the problem of aligning the multiple point clouds that make up the 3D model of the scene, see Figure 1. When using RGB-D cameras, the dual nature of these sensors allows for the setup of the problem using two different approaches: registration can be computed by aligning several depth images or, conversely, the RGB images. Although either approach is feasible, in this work the RGB information was used for computing the alignment, since depth data had several disadvantages: lower resolution, relatively small precision, and sensitivity to lighting conditions.

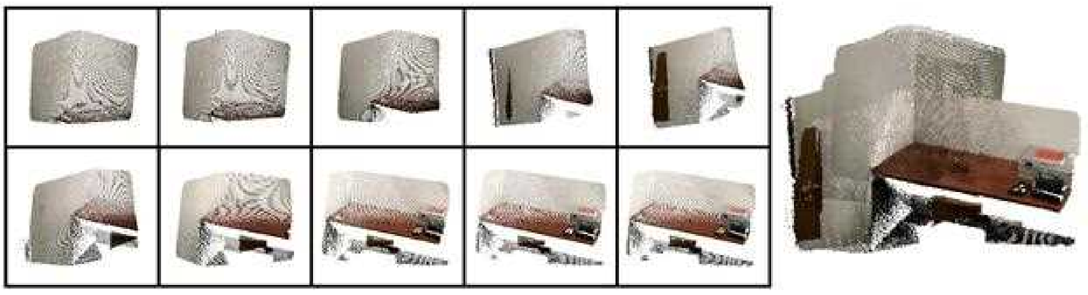
Figure 1. Example of registration of several point 3D point clouds to create a complete 3D model.

Considering the example of an indoor reconstruction performed with a handheld device. This device would capture information as it travels through multiple positions in the environment, allowing for the collection of data from various places of the scene in different angles, places that may be occluded, or too far to be captured from the initial viewpoint of the acquisition. In order to obtain the completed 3D model, these multiple parts of the scene must be registered together. The alignment of point clouds is a problem directly tackled by ICP (Iterative Closest Point) [30]. However, this algorithm works by minimising the difference in a pair of point clouds. In order to align multiple ones, the algorithm must be applied many times. If this is done in a chain, for example, it creates a very real possibility of a drift, a cumulative error that grows as each pair is fit together.

In order to perform a registration, distinctive features of the captures, such as blobs, edges, contours, line intersections, or corners may be used to find correspondences between them. These features are typically represented in a feature vector. The selected features should be invariant to scale and rotation [31]. Examples are SIFT (Scale-Invariant Feature Transform) [32], SURF (Speeded Up Robust Features) [33], and HOG (Histogram of Oriented Gradients) [34]. The most common algorithms used in feature detection are Canny [35] and Sobel [36] for edge detection, Harris [37] and SUSAN [38] for edge and corner detection, Shi-Tomasi [39] and level curve curvature [40] for corner detection, FAST [41], Laplacian of Gaussian [42], and Difference of Gaussians [32] for corner and blob detection, MSER [43], PCBR [44], and Gray-level blobs [45] for blob detection.

Following the feature detection in a pair of captures, a set of key-points per capture is generated. Then, the matching of these key-points is performed, that is, identifying corresponding key-points between captures. There is a vast range of different approaches that tackle the matching problem, from brute-force matchers [46] and FLANN (Fast Library for Approximate Nearest Neighbours) [47] to pattern recognition [48], all of them very dependent on the type of scene and the available processing time. This step is still a challenge and there is always the possibility that a significant number of detections will be matched incorrectly.

As discussed previously, the feature matching process seldom works perfectly in real case scenarios. Matching features incorrectly means it is impossible to find a transformation that works for all matches found. One way to solve this problem is by recursively estimating a model that works for a subset of feature matches, until the size of that subset is considered large enough, RANSAC [49], while discarding the remaining matches, or even using the Iterative Closest Point (ICP) algorithm [30], that does not require individual feature matching in some implementations [50]. While there is still a matching procedure, it is merely based on distance. Each feature is paired with its closest neighbour of the reference capture and the transformation model is recursively estimated through a hill climbing algorithm, so the distance between neighbours approaches zero.

Some specialised software packages can automatically identify matching features in multiple images, such as Autodesk ReCap [51] or AliceVision Meshroom [52]. The distinctive features used are often corners or line segments. In the following step, instead of trying to align captures, the matching of the features is used to produce estimates of the camera positions and orientations (pose) and the 3D coordinates of these said features, producing a point cloud. It is important to refer that the better these estimates are, the better alignment of the captures we should expect to see, and as such the registration becomes an optimization problem.

In the context of this work, Visual Simultaneous Localisation and Mapping (Visual SLAM) [53] may be used. It utilises 3D vision to perform location and mapping, by solving an optimisation problem where the goal is to compute the configuration of camera poses and point positions that minimises the average reprojection error. The method of choice to solve this problem is called bundle adjustment, a nonlinear least squares algorithm which, given a suitable starting configuration, iteratively approaches the minimum error for the whole system.

Given a set of measured image feature locations and correspondences between them, the goal of bundle adjustment is to find 3D point positions and camera parameters that minimise the reprojection error. This optimization problem is usually formulated as a non-linear least squares problem, where the error is the squared ℓ^2 norm, or Euclidean norm, of the difference between the observed feature key-points in the image and the projection of the corresponding 3D points to the image plane of the camera (reprojection error). The LM (Levenberg-Marquardt) algorithm [54] is the most popular algorithm for bundle adjustment.

In 2010, Agarwal et. al. [55] presented the design and implementation of a new inexact Newton type Bundle Adjustment algorithm. Considering the existence of a series of 3D points in the real world, these points are captured in images by different cameras, each camera being defined by its orientation and translation relative to a reference frame, its focal length and distortion parameters. After the desired acquisitions have been completed, the 3D points are projected into the images and the 2D coordinates are then compared to the ones obtained by feature detection in the images. The goal being to adjust the initial estimation of the camera parameters and the position of the points in order to minimise the reprojection errors, that is,

$$\min_{\widehat{\mathrm{P}}^i, \widehat{\mathbf{X}}_j} \sum_{ij} \ell^2 \left(\widehat{\mathrm{P}}^i \widehat{\mathbf{X}}_j, \mathbf{x}_j^i \right)^2, \tag{1}$$

where ℓ^2 is the Euclidean norm, $\mathbf{x}_j^i$ are the coordinates of the j-th point as seen by the i-th camera, $\widehat{\mathrm{P}}^i$ is the projection matrix of the i-th camera, and $\widehat{\mathbf{X}}_j$ are the 3D points [56].

Another technique of which a key component is Bundle adjustment is Structure from Motion (SfM). SfM solves a problem analogous to visual SLAM, the main difference being that SLAM is usually meant to work in real-time on an ordered sequence of images, while SfM approaches often work on an unordered set of images as post-processing, many times done in the cloud. It can, using several images captured by one or multiple cameras, produce point cloud based 3D models, similar to those obtained by RGB-D cameras or LiDAR. This technique can be used to create models of objects with consumer-grade digital cameras and has been made possible by advances in computers, digital cameras, and UAV (Unmanned Aerial Vehicles). Together, these advances have made it feasible for a wide range of users to be able to generate 3D models, without extensive expertise or expensive equipment. SfM uses triangulation to calculate the relative 3D positions (X, Y, Z) of objects from pairs of images, often captured from a single moving camera, or with different cameras, in different locations and/or angles.

One of the main problems of registration, as discussed previously, is the unreliability of the detected features between captures. In the case of Bundle Adjustment, the minimisation of the reprojection may be computed using a wrongly matched feature in a pair of images, causing errors in the alignment of captures. Our approach uses fiducial markers to improve the feature detection and matching step. Markers are placed on the environment as visual features to be detected in post-processing. Aruco markers [57,58] were used (binary square fiducial markers) since existing software allows for robust and fast detection. Each marker has an identifier (id) (see Figure 2), making it very hard for false matching to occur: although under poor lighting conditions or in blurred images some aruco markers may go undetected, it is highly improbable that they will be identified with the wrong id.

The biggest downside to this approach is the pollution of the acquired scene's texture with the fiducial markers. Therefore, along with the optimization procedure, an additional tool was developed, that allows the automatic removal of aruco markers from the texture of the scene, taking advantage of the obtained registration between acquisitions.

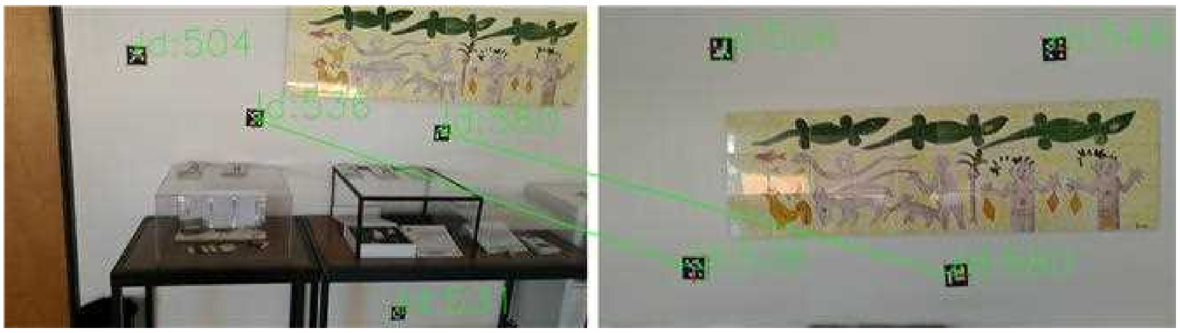

Figure 2. Detection and matching of aruco markers in a pair of captures.

3. Optimization of Camera Pose Estimation Using Fiducial Markers

This section focuses on an optimization module developed to improve the registration of 3D reconstructions using fiducial markers. The problem was approached by creating a generic optimization Application Programming Interface (API) and applying it to the specific case of refining the pose of each camera with respect to a common reference frame in a 3D reconstruction procedure.

A camera is defined by the sensor's internal properties and its positioning in space, relative to a reference frame. The former is represented by the intrinsic camera matrix, which was always consistent since the same device was used for all captures. It was retrieved using a chessboard based camera calibration procedure and can be represented as

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where c_x, c_y are the principal point coordinates, and f_x, f_y are the focal lengths expressed in pixel units. The latter is represented by an extrinsic camera matrix, such as

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}, \tag{3}$$

which may be decomposed into two components, a translation and a rotation. The translation can be represented as the vector $T = (t_x, t_y, t_z)$, and the rotation can be represented as a 3×3 rotation matrix R.

The optimization is performed as a bundle adjustment, meaning the objective of the optimization is to refine the camera poses and the 3D point positions. As such, the set of parameters to be optimized $\mathbf{\Phi}$, is defined as

$$\mathbf{\Phi} = \Big[\overbrace{x_{i=1}, y_{i=1}, z_{i=1}, r1_{i=1}, r2_{i=1}, r3_{i=1}, \ldots, x_{i=I}, y_{i=I}, z_{i=I}, r1_{i=I}, r2_{i=I}, r3_{i=I},}^{Camera\ poses}$$
$$\overbrace{x_{j=1}, y_{j=1}, z_{j=1}, \ldots, x_{j=J}, y_{j=J}, z_{j=J}}^{Marker\ translations} \Big], \tag{4}$$

where i refers to the i-th camera, of the set of I cameras, and j refers to the j-th aruco marker, of the set of J aruco markers. Notice that, in this vector, the rotation for one camera (r_1, r_2, r_3) is represented through the axis/angle parameterization, as opposed to the 3×3 rotation matrix format of the camera's extrinsic matrix. Because a rotation matrix has $3 \times 3 = 9$ elements, but only 3 degrees of freedom, a different parameterization is necessary in order to intrinsically incorporate constraints on the rotations during the optimization.

Popular parameterization for rotations are Euler angles, quaternions, and axis/angle representation. However, not all representations are suitable for an optimization. Parameterization should not introduce more numerical sensitivity than the one inherent to the problem itself, as this decreases the chances of convergence in optimizations. When the parameterization formats follow this rule, they can be referred to as fair parameterization [59]. For example, Euler angles, which are probably the most used angle parameterization, are not suitable for optimizations [60], because they do not yield smooth movements, each rotation is non-unique and, most notably, they introduce singularities, known as Gimbal lock, where one degree of freedom is lost [60]. Because quaternions have 4 components which are norm-1 constrained, and this introduces some complexity in the algorithms, they are usually not used for optimizations [60], even though they are a fair parameterization. The axis/angle parameterization is the most widely used to represent a rotation in an optimization. It is a fair parameterization and has only three components, two values to define a unit vector indicating the direction of an axis of rotation, and one value to define an angle. In this way, any rotation can be represented as a rotation around this axis, by an angle θ. To convert between the axis/angle format and the 3×3 rotation matrix format (used in the data model to ease implementation) the Rodrigues' rotation formula was used.

The datasets were obtained using the Google Tango platform, meaning they already contained information about the poses of the cameras, which was used as a first guess for the optimization procedure. Thus, in this sense, the proposed approach is a refinement of an initial proposal of camera pose and corresponding reconstructed digital 3D model. However, if this data is not available, our system is able to produce one by utilizing the detected markers. In this case, one of the cameras is considered as the World reference frame and the transformations from each camera to the World are calculated using chains of transformations between markers and cameras, taking advantage of common marker detections between cameras. We also needed a first guess for the positions of the 3D points, which correspond to the aruco marker centers. Their poses must be determined in the world reference frame. Having an initial guess of the camera poses (transformations from each camera to the world), for each aruco marker we need only a transformation from its reference frame to one of the cameras in which it can be detected. Thus, the initial positions for the 3D points are obtained by calculating the aggregate transformation from each aruco marker to the world reference frame as

$$ {}^{A_j}\mathbf{T}_W = {}^{C_i}\mathbf{T}_W \cdot {}^{A_j}\mathbf{T}_{C_i} , \tag{5}$$

where A_j refers to the j-th aruco marker detected, C_i refers to the i-th camera (the first in which the j-th aruco marker can be detected), and W refers to the world reference frame.

The cost function is based on the reprojection error, that is, the geometric error corresponding to the distance (in pixels) between a 3D projected point (with the evaluated pose) and a measured one (computed from the aruco detection), as in Bundle Adjustment. The relationship between a 3D point in the world and the pixels that correspond to its projection on an image plane can be expressed as

$$ s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\,T \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} , \tag{6}$$

where X, Y, Z are the coordinates of a 3D point in the world; u, v are the coordinates of the projection point in pixels; K is the intrinsic camera matrix and T is the extrinsic camera matrix. T translates coordinates of a 3D point (X, Y, Z) to a coordinate system fixed with respect to the camera, meaning it corresponds to the transformation from the world to the camera reference frame.

The optimization is performed using a nonlinear least-squares regression, as indicated for Bundle Adjustment problems. The function *least squares* from the SciPy library [61] is used. It requires a cost function, a vector of parameters, bounds (infinite by default), and a sparse matrix (initializes to identity if not available) to carry out the optimization. At the end of the optimization, the function returns a vector with the optimized parameters for the pose of the cameras and the position of the 3D points, which minimises the reprojection errors. Because of the wrapper implemented, these values are automatically converted and copied from this vector to their place in the data models.

The visualisation function for this optimization consists of showing all the images, each with the aruco markers centers detected, the initial projections connected to the target by a blue line, and the projections at the current step of the optimization, as seen in Figure 3 on the left. In this case, it is the end of the optimization, and so the final position is overlapping with the detected one. This is complemented by a 3D representation of the pose of the cameras and position of the 3D points in the scene, showcased in Figrue 3 on the right.

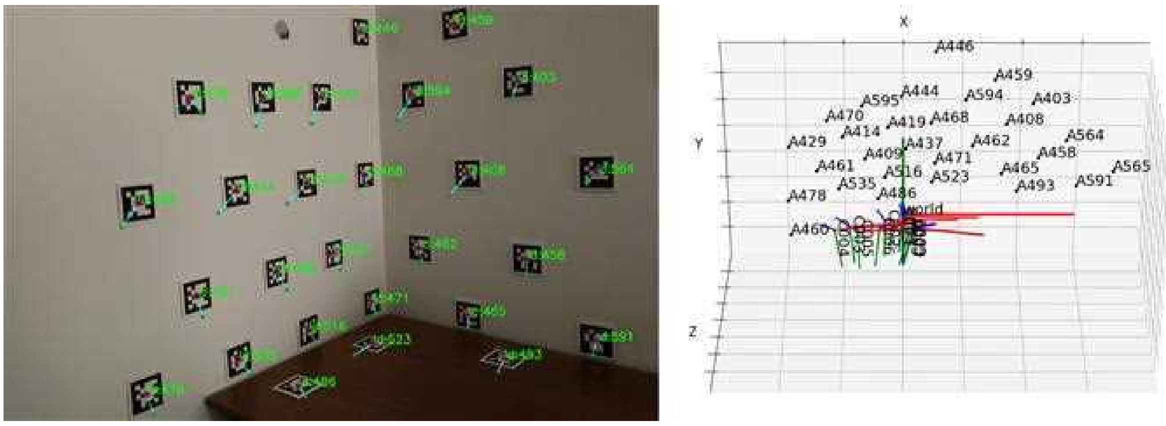

Figure 3. Visualisation function: (left) Aruco marker detected position in red (target of optimization), initial projection in green, current projection in dark blue, drawn over one image; (right) 3D representation of the position of the cameras and aruco markers in the scene.

The estimated transforms are applied to the point clouds to ensure that the optimized datasets are in the calibrated reference frame. The applied transformation is

$$^{\text{oldW}}\mathbf{T}_{\text{newW}} = {}^{C_i}\mathbf{T}_{\text{newW}} \cdot {}^{\text{oldW}}\mathbf{T}_{C_i}\,, \tag{7}$$

where `oldW` refers to the world reference frame before optimization, `newW` refers to the world reference frame after optimization, and C_i refers to the i-th camera.

In Figure 4, we can see a merged point cloud coloured with the original images. The enhancement of the registration caused by the optimization improves the texture significantly, as showcased by Figure 5 in further detail. Figure 6 presents the results of the process in several views, showing clearly a better alignment of the different point clouds after the optimisation.

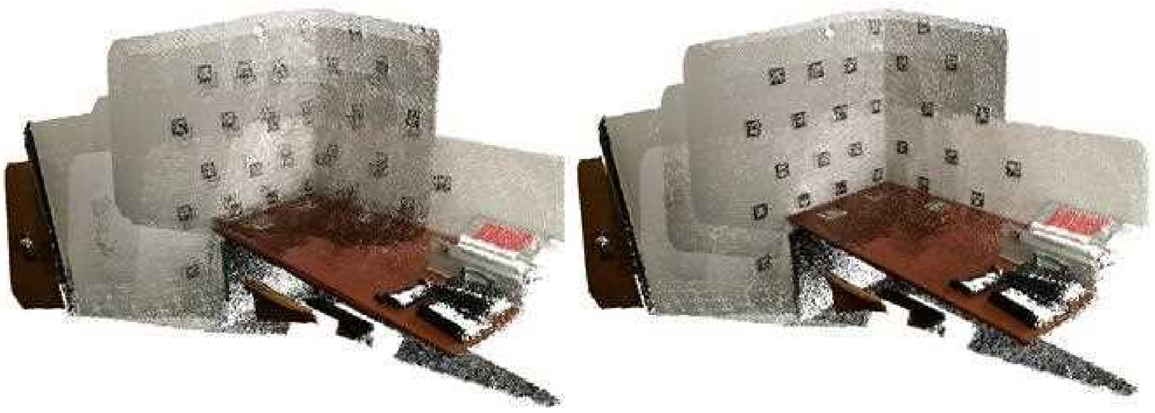

Figure 4. Merged point cloud coloured with texture obtained from images: (**left**) before optimization; (**right**) after optimization.

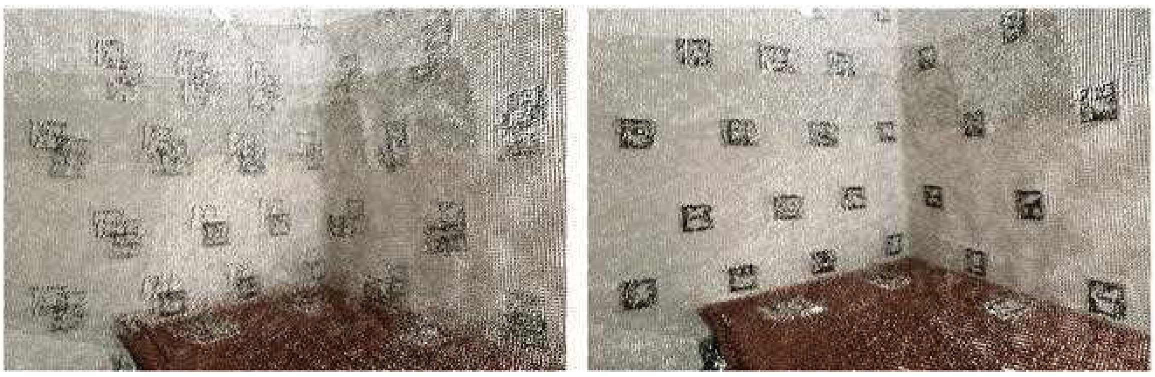

Figure 5. Merged point cloud coloured with texture obtained from images (detail): (**left**) before optimization; (**right**) after optimization.

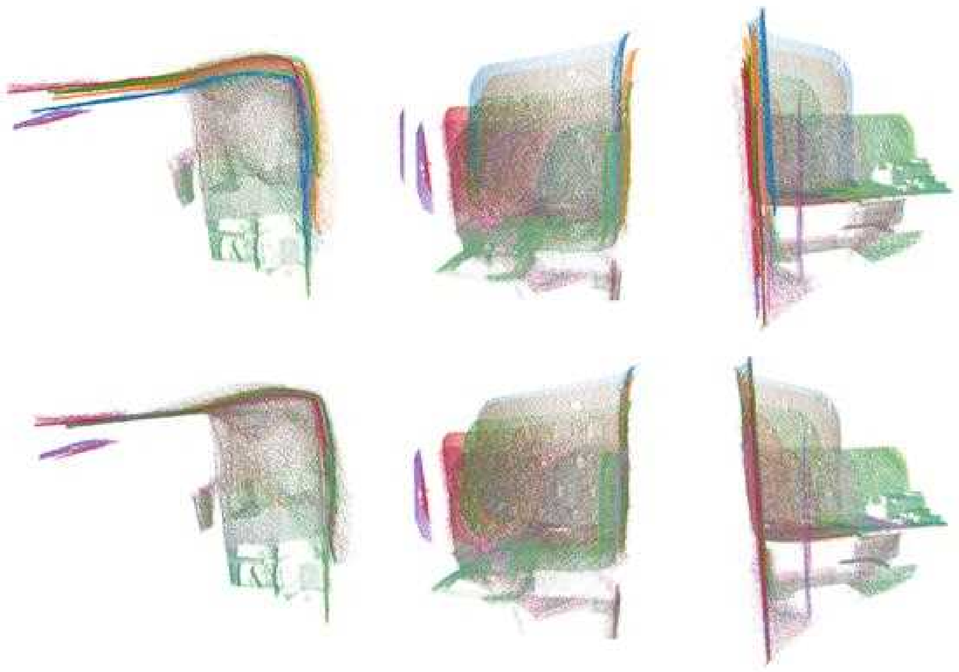

Figure 6. Several views of a reconstructed scene: (**top**) before optimization; (**bottom**) after optimization. Point clouds corresponding to each capture are colored differently for better visualization.

4. Texture Refinement and Marker Removal

This section focuses on a module developed to automatically remove fiducial markers from a scene. In Section 3, a method to improve the registration of a 3D reconstruction using fiducial markers was presented. The module showcased in the present section was created as an attempt to avoid the pollution of the acquired scene's texture with these markers. There are other scenarios, besides 3D reconstruction, where having fiducial markers in the environment is advantageous, such as Augmented Reality (AR) applications, and being able to successfully remove them from the texture may improve the final visualizations significantly.

This module is agnostic to the way the dataset is stored and loaded to memory. By using a different dataset loader, one may utilise this inpainting tool in a different context. The tool was programmed to detect and remove aruco markers, since this was the kind of fiducial marker used in Section 3. However, it could be used for different markers by adapting the detection and pose estimation functions. All the masks showcased in this section were blended with images to better showcase where they fit and what regions are the target.

4.1. Navier-Stokes-Based Inpainting

The Navier-Stokes-based Inpainting algorithm available in the OpenCV library was used in this work. Its function is to restore a region in an image using the region neighbourhood. To operate, this algorithm requires a mask representing the region to replace. In a first attempt, the masks of the fiducial markers were created using the corners provided by the aruco detector. This method was not valid since these masks did not account for the margin around the markers (see Figure 7 left), resulting in blank cards scattered through the scene. To solve this issue, the first approach taken was to dilate the masks. However, this 2D operation in the image plane did not solve the problem for markers that are close or with large angles relative to the camera (see Figure 7 center). The solution found was to create a mask as a 3D object, with the same dimensions as the markers (including thickness), and projecting the 3D masks, corresponding to each detected marker, to the 2D images. This ensures the whole intended area is masked at any distance and angle (see Figure 7 right).

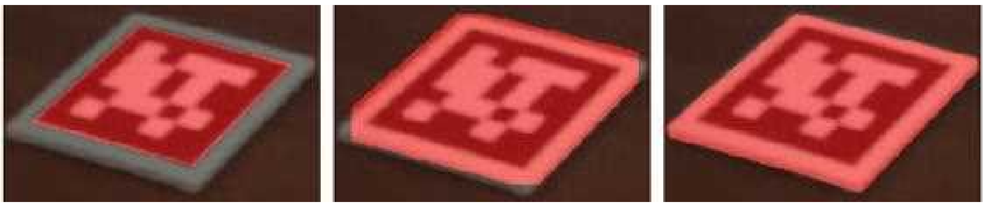
Figure 7. Methods for mask creation: (**left**) detection; (**center**) 2D dilation; (**right**) 3D projection.

Having an accurate representation of the region covered by the marker, the challenge was to define the colours that will be used to restore that region. The results of the Navier-Stokes-based Inpainting from OpenCV were underwhelming, as shown in Figure 8. The examples found online for the testing of this algorithm are usually about removing lines from the images or restoring damage from folding photographs. Inpainting in this manner is usually done in small areas, while the size of a fiducial marker in the scene is substantial.

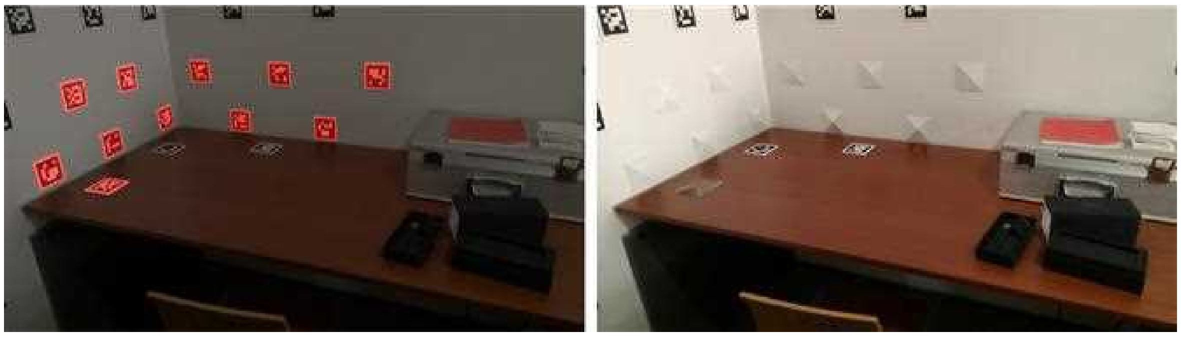
Figure 8. Example dataset image inpainting: (**left**) mask used for marker removal; (**right**) results of Navier-Stokes-based inpainting from `OpenCV`.

4.1.1. Inpaiting of Homogeneous Regions through Image Blurring

Given the limitation of the first implementation of the inpainting, and taking into account the placement of the markers over homogeneous texture regions, the blurring of the area where the inpainting is applied was performed with high aperture linear size (median blur with kernel size of 201), as seen in Figure 9.

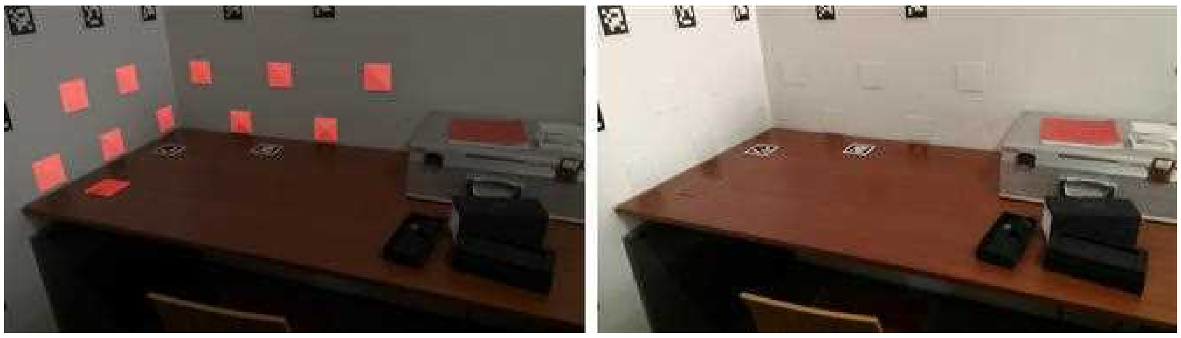

Figure 9. Example dataset image restoration attempt: (**left**) mask for the pixels to be replaced; (**right**) results with blurring.

The texture obtained after blurring (Figure 9) presents some improvements, when compared to the inpainting restoration (Figure 8). However, the limits of the inpainted areas are still sharp and visible. To reduce this effect, blurring is applied once again, this time in a bigger area, defined by enlarging the 3D masks of the markers and projecting them in the obtained images (Figure 9), now using a smaller aperture linear size for the blur (median blur with with kernel size of 51), to avoid mixing colour of the surrounding objects, see (Figure 10).

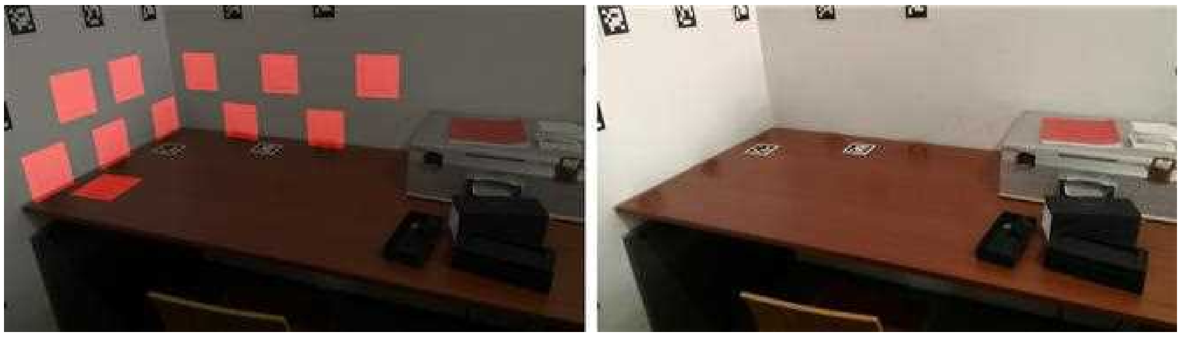

Figure 10. Example dataset image restoration improvement: (**left**) mask for the pixels to be replaced (based on enlarged 3D masks); (**right**) results with second blurring.

At the end, a bilateral filtering is applied to preserve the edges in the image, avoiding the mixing of colours, while smoothing over the surfaces, making the grainy pattern of the walls less noticeable for example, see Figure 11.

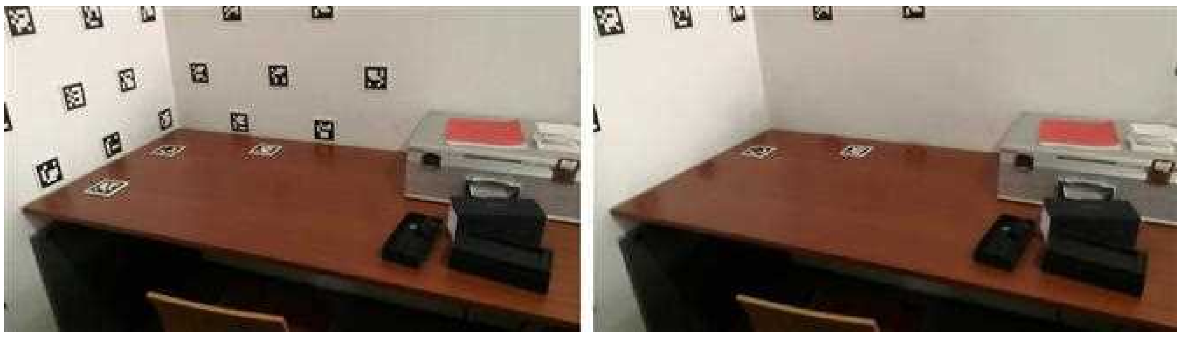

Figure 11. Example dataset image restoration: (**left**) original image; (**right**) restored image.

The values for the blurring factor, along with the size of the marker objects and patch size around them, can be modified through command line arguments or assigned easily in the code. These values were empirically set because they depend on various distinct factors, the characteristics of the scene, and the sensors used.

4.1.2. Inpainting Non-Detected Markers: Cross-Inpainting

It is frequent to miss fiducial marker detections in images, when the marker is not fully visible (for example in the border of the image), or when blurring occurs due to motion, for instance. Since the transformations from the aruco markers to cameras and from the cameras to the world are known (see Figure 12), it is possible to evaluate the presence of a marker in a given image, even if the marker is not detected in that particular image.

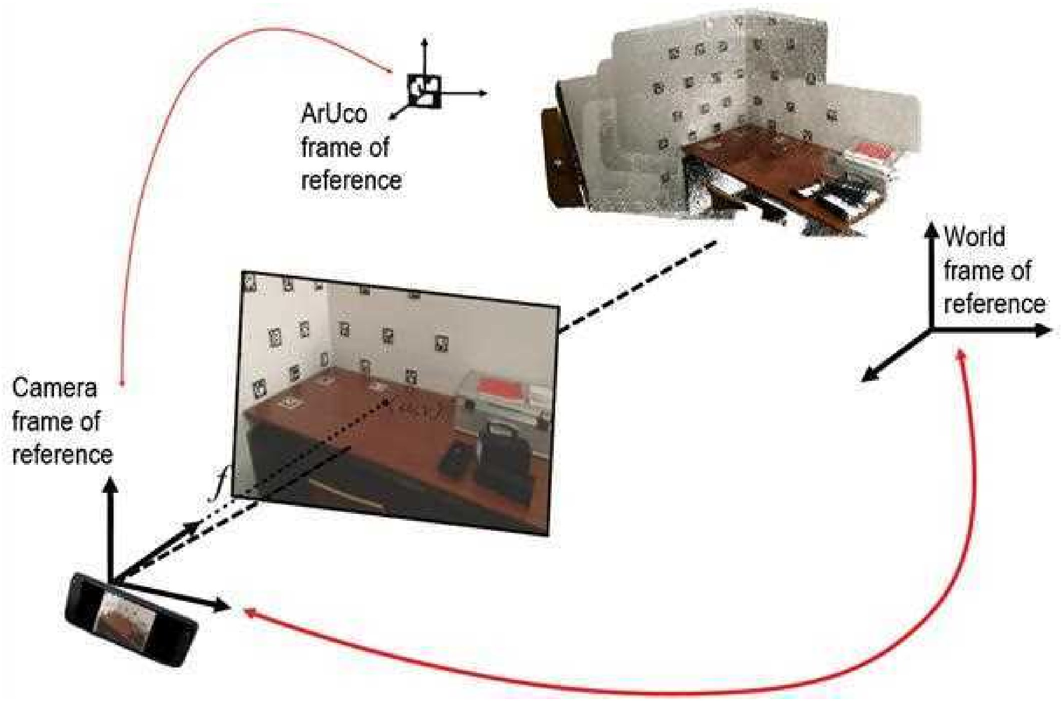

Figure 12. Conversion between different reference frames is used to perform cross-inpainting.

The solution implemented, named "Cross-Inpainting", takes advantage of transformations from the aruco markers to the world. Every time a new marker is detected in one of the images of the dataset, the transformation of the marker to the world is known. This transformation is computed using the transformation of the marker to the camera where it was detected, the *i*-th camera, and the transformation from that camera to the world:

$$ {}^{A_j}\mathbf{T}_W = {}^{C_i}\mathbf{T}_W \cdot {}^{A_j}\mathbf{T}_{C_i} \,, \tag{8} $$

where A_j refers to the *j*-th aruco marker detected, C_i refers to the *i*-th camera, and W refers to the world reference frame. Given an image captured by a camera *k*, when this image is being inpainted, it is possible to access the information for all markers and check if a marker should be present in the image even if not detected. This is done by applying the transformation from the aruco marker reference frame to the world, as in Equation (8), followed by the transformation from the world to camera *k*:

$$ {}^{A_j}\mathbf{T}_{C_k} = {}^{W}\mathbf{T}_{C_k} \cdot {}^{A_j}\mathbf{T}_W \,, \tag{9} $$

where A_j refers to the *j*-th aruco marker, C_k refers to the *k*-th camera, and W refers to the world reference frame. A 3D object representing the marker is transformed using Equation (9) and then projected to the image captured by camera *k*. This idea was applied to non-detected aruco markers before optimization (Figure 13 left) and after optimization (Figure 13 right), showing the viability of this approach after optimization to improve the inpainting of partially visible or non detected aruco markers, see Figure 14.

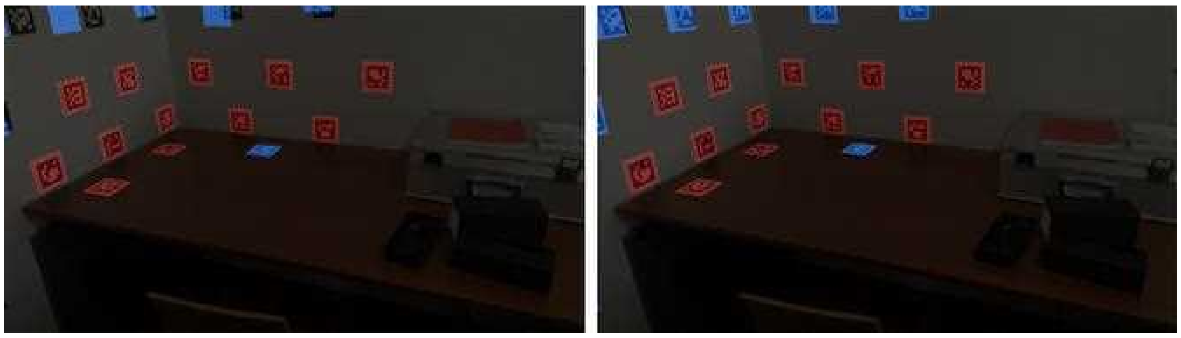

Figure 13. Masks obtained from aruco marker detection in red and masks obtained from projections from other cameras (cross-Inpainting) in blue: (**left**) before optimization; (**right**) after optimization.

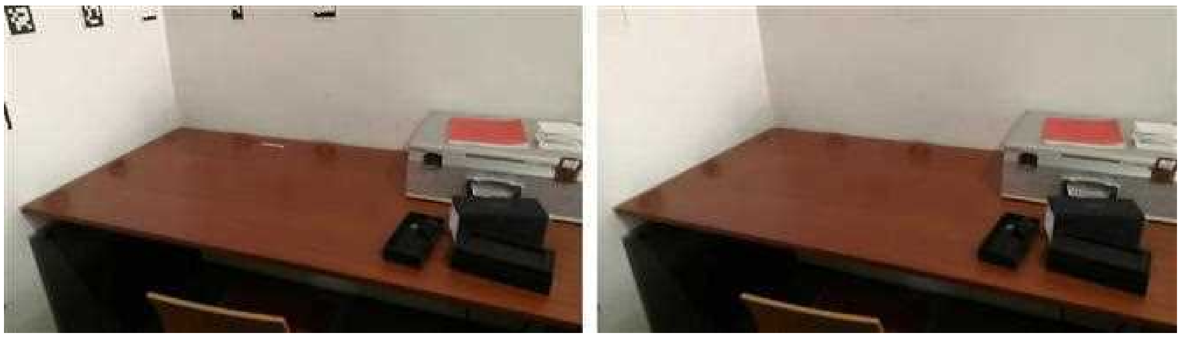

Figure 14. Restoration with cross-inpainting result: (**left**) before optimization; (**right**) after optimization.

4.1.3. 3D Point Cloud with RGB Visualisation

To better visualise and evaluate the results of the registration and inpainting operations, we developed operations to allow the visualization of fused RGB coloured 3D point clouds. For each point cloud, the coordinates of the 3D points are projected into the corresponding colour images to extract RGB pixel values. Finally, a new .ply file is created with the XYZ-RGB information for the fused point cloud that can be visualized in any 3D viewer (Meshlab, PCL viewer, PPTK library, etc.). An example of the registered visualization of 10 coloured point clouds with inpainting, before and after optimization, is presented in Figure 15.

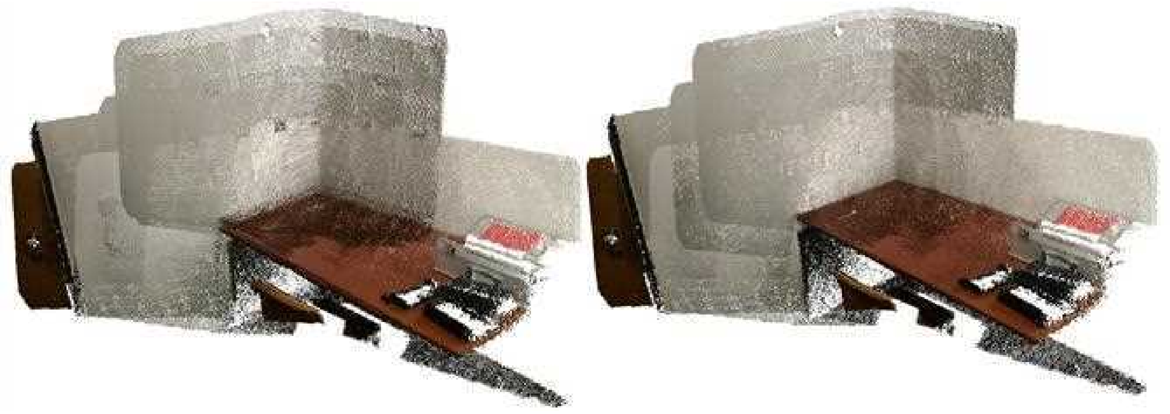

Figure 15. Point cloud coloured with texture obtained from restored images, using cross-inpainting: (**left**) before optimization; (**right**) after optimization.

For comparison purposes, in Figure 16 a merged point cloud, coloured with images restaured using only the detected markers can be observed. At first glance, the texture applied before optimization seems better, this effect is caused by the fact that the points closest to the viewer happen to correspond to images where more markers were detected. After the optimization, the merged point clouds are better aligned and the undetected markers, which in Figure 15 (right) were removed using cross-inpainting, here show through in Figure 16 (right).

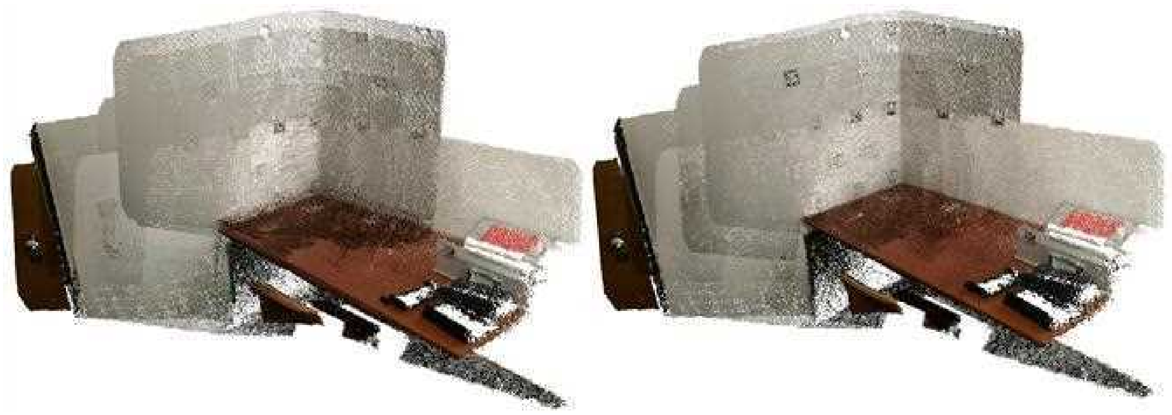

Figure 16. Point cloud coloured with texture obtained from restored images, no cross-inpainting: (**left**) before optimization; (**right**) after optimization.

5. Results

This section contains the evaluation methodology and the description of its process, going over the collection of datasets and showcasing the results obtained in various ways.

5.1. Evaluation Methodology

Visual comparison of the point clouds may sometimes prove to be somewhat subjective and inconclusive. Particularly in datasets with a very large number of point clouds. The reporting of results in a document also means that the changes must be observed in a side by side manner, which makes them harder to perceive than in a 3D visualization software such as Meshlab, for instance, where it is possible to instantly switch between point clouds placed in the same exact place. In order to provide some quantitative results, we performed experiments in a meeting room of the Department of Mechanical Engineering (DEM) at the University of Aveiro, for which a laser scan was acquired with a FARO Focus Laser Scanner [62]. Given the high precision of the FARO laser sensor, a point cloud of the scan was considered as the ground truth to evaluate the proposed process of registration based on fiducial markers, and compare it to a Google Tango reconstruction [63], as well as a reconstruction with registration refinement using ICP [30]. As the meeting room was larger than the previously used datasets, it was also an opportunity to create a more difficult challenge for the optimizer.

5.2. Datasets

The datasets were collected using the ZenFone AR, a Google Tango enabled Android phone, equipped with an RGB-D camera. The example dataset, used in previous sections, was collected in office 22.2.18 of DEM: a relatively small dataset with a large density of fiducial markers. Two additional, more complex datasets were captured: the meeting room mentioned in Section 5.1, and a lobby, just outside the meeting room. Unfortunately, due to logistic problems, the FARO laser scan and the RGB-D datasets were not acquired in the same day, resulting in some differences in the scene (namely some furniture had been moved within the room, such as couches and chairs). Despite this limitation, it was possible to validate our results using an area of the model with little change and focusing on walls and ceilings.

5.3. Collecting Ground Truth

The laser scan of the meeting room was acquired with a FARO Focus Laser Scanner [62]. A single point cloud was obtained from the merging of several scans, in Cloud Compare. However, given its very high density (50698121 vertices), unnecessary for the purpose of this study, a downsampled and clipped version was generated using Cloud Compare. The downsampling of the FARO laser acquisition result (3722614 vertices) can be observed in Figure 17, and the final result (1777550 vertices) in Figure 18.

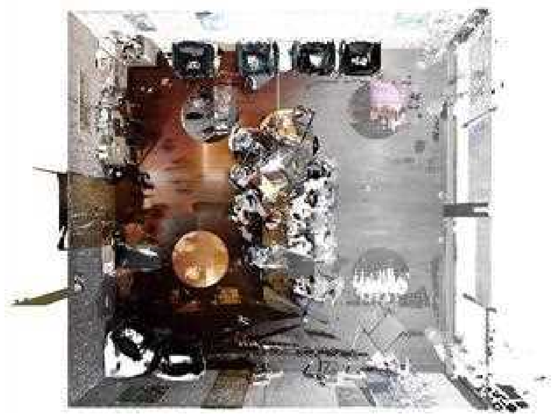
Figure 17. FARO laser scanned point cloud after downsampling.

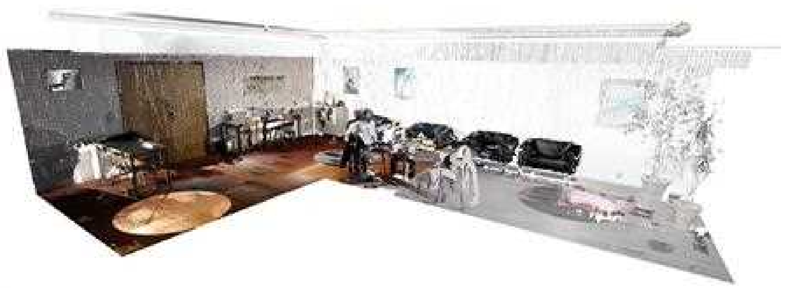
Figure 18. FARO laser scanned point cloud after downsampling and clipping.

5.4. Evaluation Procedure

The key metric used for the evaluation procedure was the Hausdorff distance. It measures how far one subset of a metric space is from the other, and is often applied to the measurement of distance between point clouds. It can be thought of as the maximum distance of a set to the nearest point in the other set [64]. The Hausdorff distance from set A to set B may be defined as

$$h(A,B) = \sup_{a\in A} \inf_{b\in B} d(a,b) \ , \tag{10}$$

where sup represents the supremum, inf represents the infimum, and $d(a,b)$ is the Euclidean distance between a and b.

The tools available in Meshlab, allow us to compute, not only an absolute distance, but also the distance at each point of the cloud. This distance can then be mapped visually through a colour. ICP was used in Cloud Compare to align processed point clouds with the FARO laser ground truth point cloud. Then, Meshlab's Haudorff distance was used to compute the distance between the given point cloud and ground truth, providing a coloured visualization of the distance. A red-yellow-green-blue colourmap was used, where red maps a small distance and blue a large distance. This visualization allows for the understanding of which areas have significant error compared to the ground truth. It also enables the identification of areas where the scene had physically changed between the capture of the ground truth and our datasets (namely furniture movements).

We used Google Tango's reconstruction as baseline and compare it with a cumulative Iterative Closest Point (ICP) approach, and our proposal. The cumulative ICP approach was applied in a pairwise fashion from cloud i to i+1, these captures were then merged and became cloud i for the next

ICP calculation. The starting point of the reconstruction for the ICP procedure was the same as the one used for our approach.

5.5. *MeetingRoom* Dataset

In this subsection, the results obtained using the *MeetingRoom* dataset are presented. This dataset takes advantage of the fact that there is the possibility of analysing it in comparison with the FARO laser scanned point cloud, see Section 5.3. The results are compared with Google Tango and ICP, first using the texture of the original images in Figures 19–21, then using the Hausdorff distance based color mapping in Figures 22 and 23, and finally, the fiducial marker removal results are showcased in Figure 24.

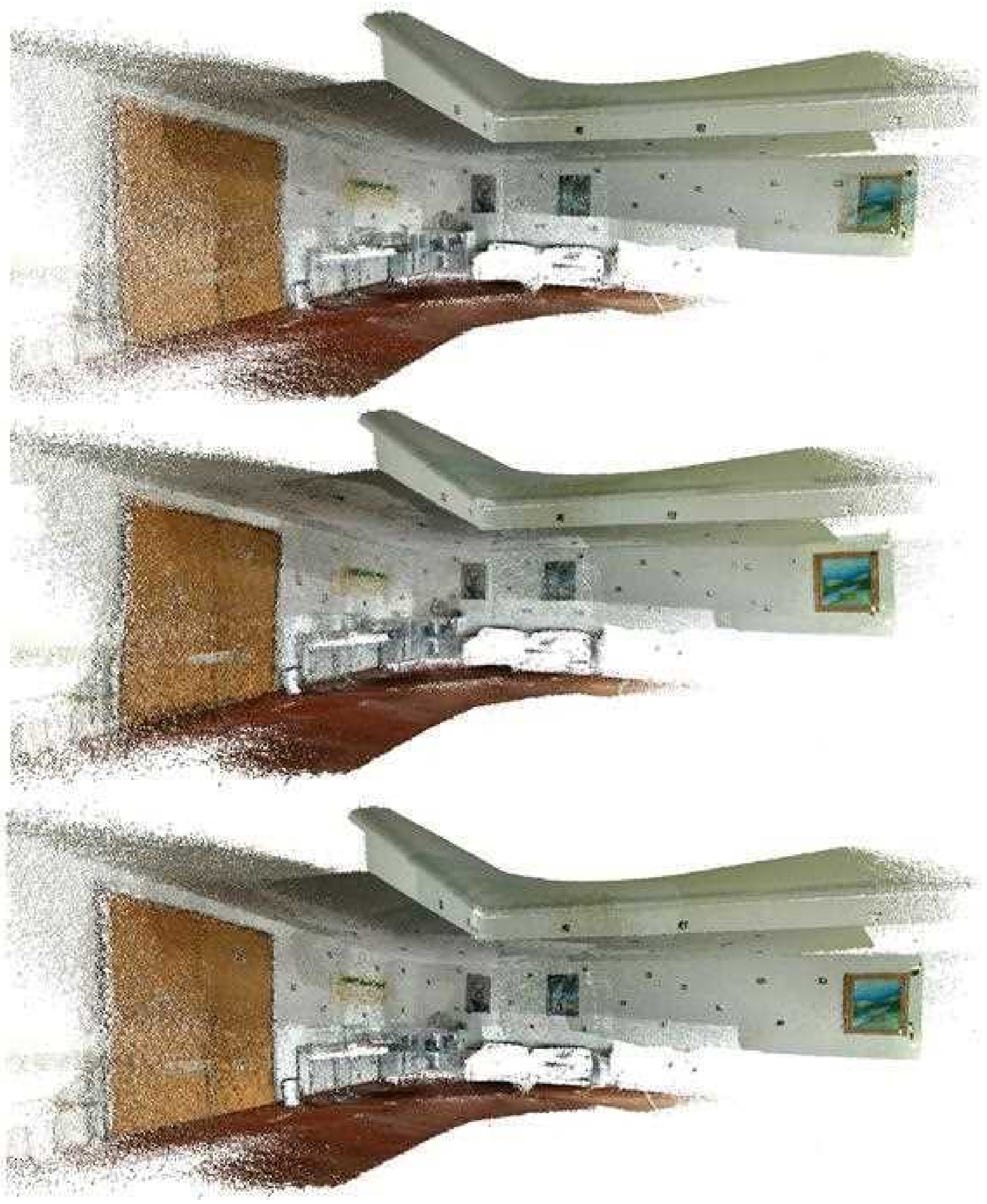

Figure 19. *MeetingRoom* dataset coloured with texture obtained from original images: (**top**) Google Tango; (**middle**) Iterative Closest Point (ICP); (**bottom**) Our approach.

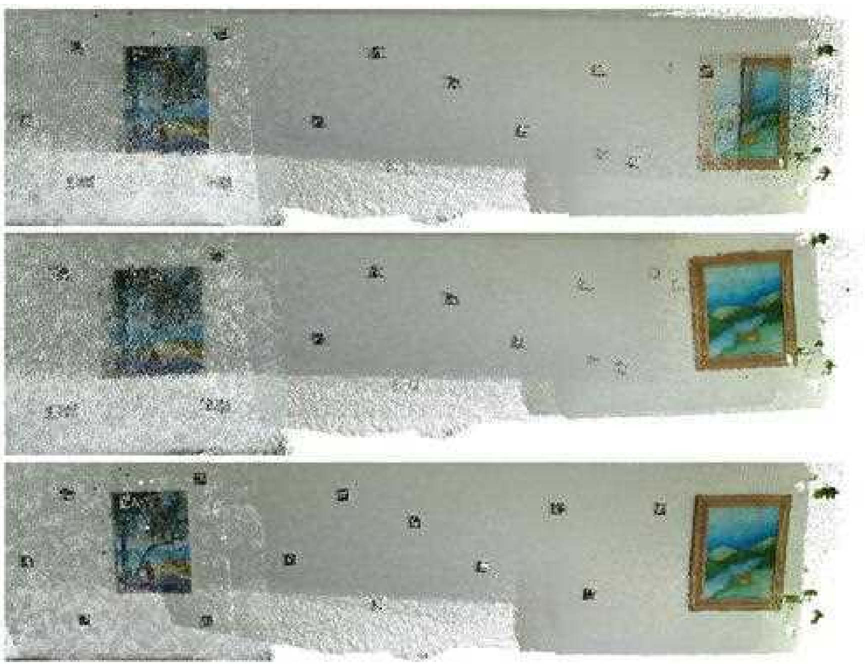

Figure 20. *MeetingRoom* dataset coloured with texture obtained from original images (detail 1): (**top**) Google Tango; (**middle**) ICP; (**bottom**) Our approach.

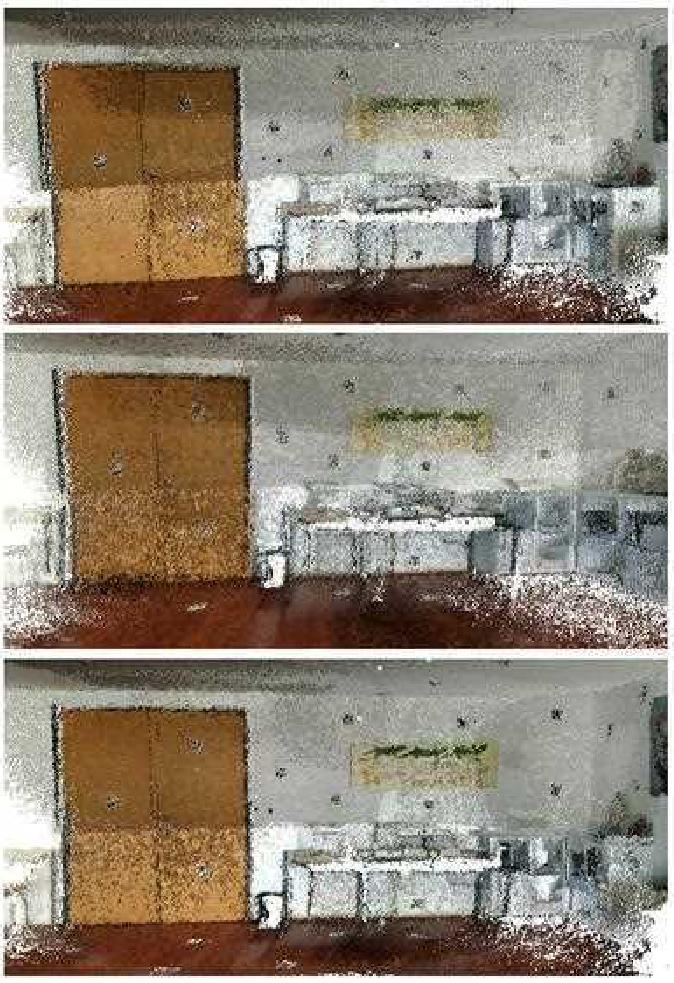

Figure 21. *MeetingRoom* dataset coloured with texture obtained from original images (detail 2): (**top**) Google Tango; (**middle**) ICP; (**bottom**) Our approach.

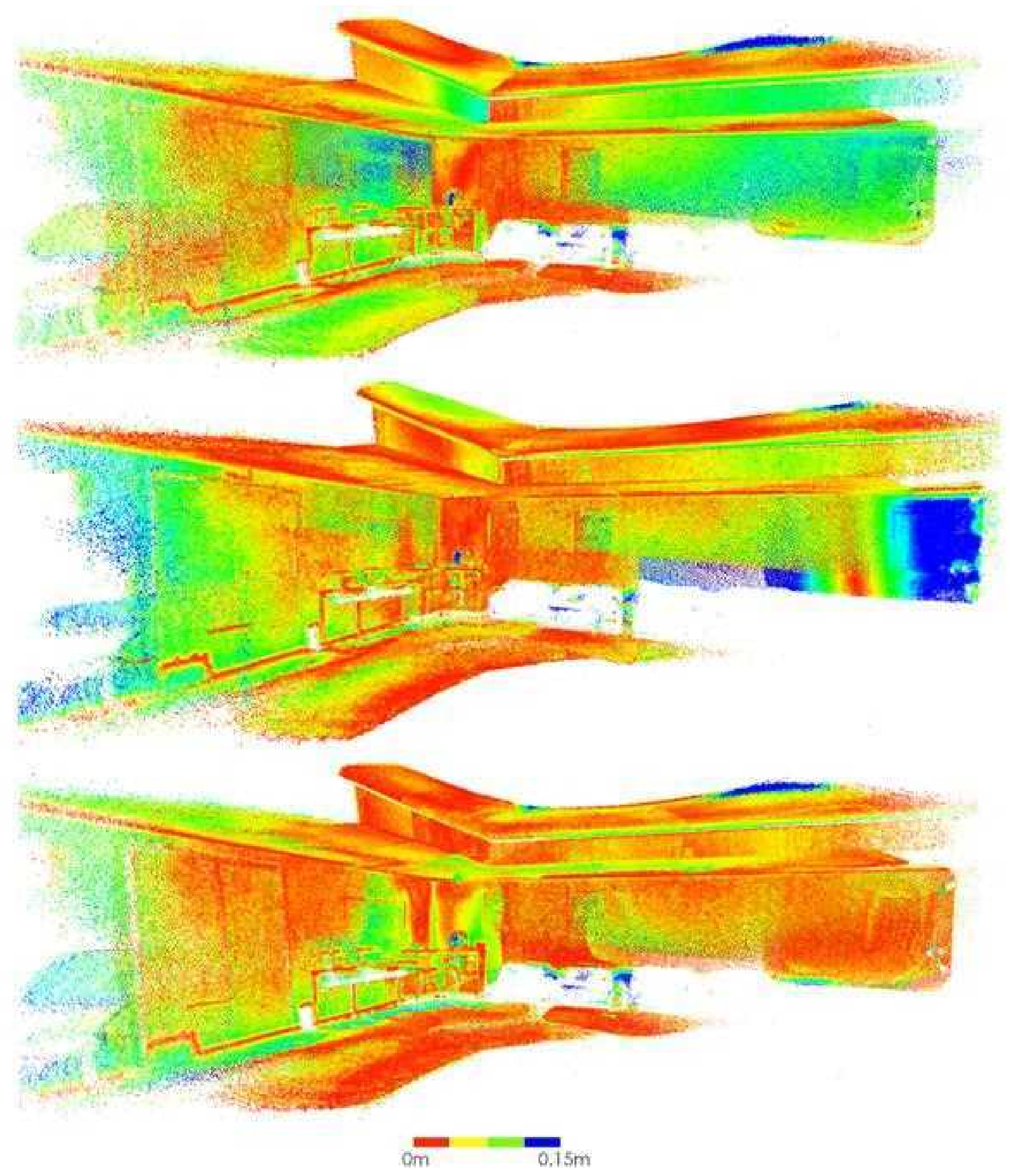

Figure 22. *MeetingRoom* dataset colourmapped with Hausdorff distance to FARO laser scanned point cloud: (**top**) Google Tango; (**middle**) ICP; (**bottom**) Our approach.

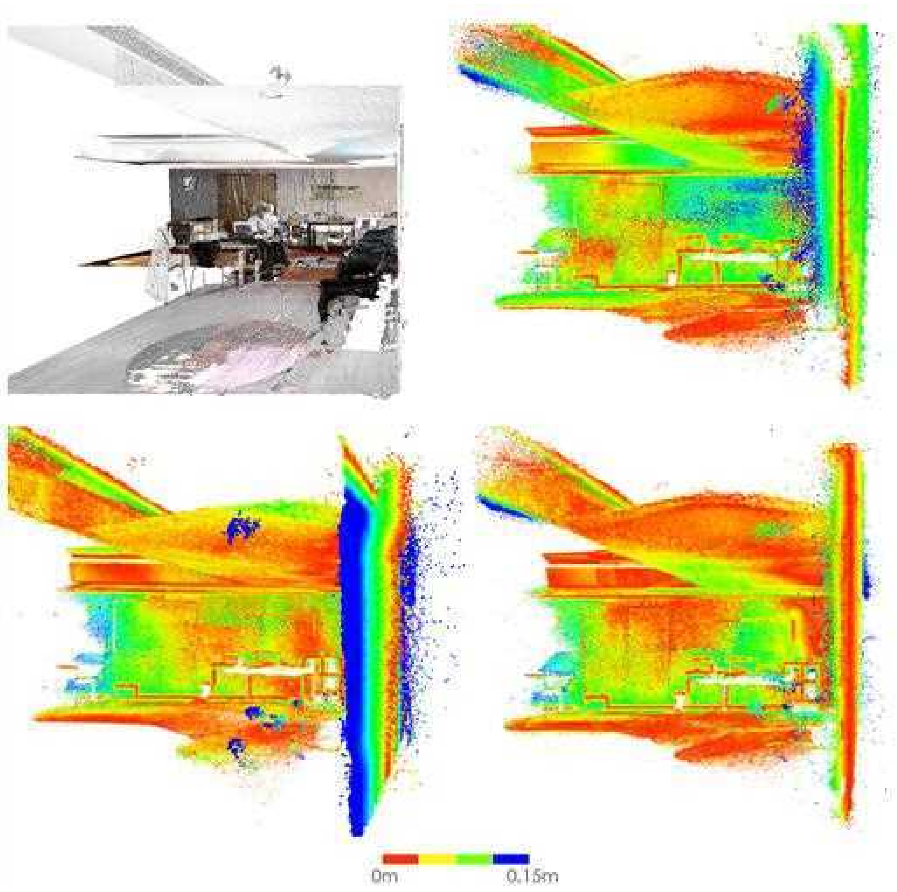

Figure 23. *MeetingRoom* dataset colourmapped with Hausdorff distance to FARO laser scanned point cloud (detail): (**top-left**) Groud truth laser scan; (**top-right**) Google Tango; (**bottom-left**) ICP; (**bottom-right**) Our approach.

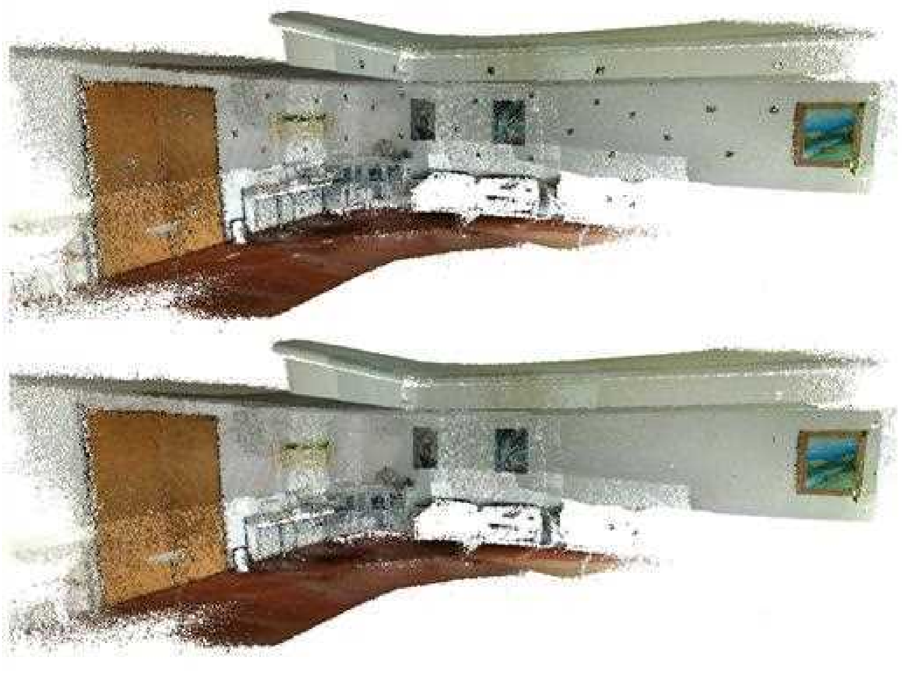

Figure 24. *MeetingRoom* dataset coloured with texture: (**top**) obtained from original images; (**bottom**) obtained from restored images.

The *MeetingRoom* dataset includes 136 RGB-D images and the alignment optimization was performed in 3m50s on an Ubuntu 18.04 laptop equipped with an Intel Core i7 7500U CPU and 8GB of RAM, see Table 1.

Table 1. 3D alignment processing times for the *MeetingRoom* dataset.

	Time (mm:ss)
ICP	03:02
Our Approach	03:50

The Hausdorff distance was calculated from each of the methods' produced reconstruction to the FARO laser generated point cloud. The local values were used to colourmap the point clouds, see Figures 22 and 23. The mean and Root Mean Square (RMS) values, see Table 2, indicate an improvement of approximately 27% of our approach over Google Tango. Regarding the comparison of our approach with ICP, it shows an improvement of approximately 13% in RMS and 5% in mean. The maximum distance considered for the measurement was empirically set to 15 cm. These values should be considered with care because they are computed globally for a very large number of points, many of which were not moved significantly during the optimization since multiple areas are already close to the ground truth before the optimization. As a result, areas that improved significantly may not have a very high impact on the value of the mean and RMS. The RMS is usually considered a more meaningful metric in the context of 3D reconstruction, since it gives a relatively high weight to large errors, which appear in the form of visual artifacts (see Figure 23).

Table 2. Hausdorff distance from the *MeetingRoom* dataset to the FARO laser scanned point cloud.

	Mean (m)	**RMS (m)**	**min (m)**	**max (m)**
Google Tango	0.0337	0.0442	0.0001	0.1500
ICP	0.0259	0.0372	0.0001	0.1500
Our Approach	0.0247	0.0324	0.0001	0.1500

The main identified limitations of our approach are: the overhead of time and logistics associated with the marker placement in the scene, when compared with methods that do not require the preparation of the environment previous to the capturing process, which is very dependent on the user, nature of the scene, and the density of markers desired; the need for at least one marker to be detected in a capture, ideally more, to allow for its registration refinement; and the pollution of the scene's texture. Some of the observed texture problems in the ICP results may be attributed to the fact that a number of captures contain points mostly belonging to the same plane, and in a pair of those captures, an ICP procedure may slide them (within the plane), while maintaining a low distance between the captures. Our approach does not suffer from this problem.

Table 1 presents the 3D alignment processing times for the MeetingRoom dataset using ICP and our approach. Google Tango processing times are not included, since it performs bundle adjustment optimization during the capture process and thus it is not directly comparable with our method that only executes the optimization after all the data has been captured. It is noteworthy that Google Tango also requires additional time after data capture to produce the final reconstructed model. Unfortunately, it is difficult to evaluate precisely tango processing time since the system is not open source and the time we could evaluate would also include other processes besides the optimization, such as mesh creation and processing of stored data changes.

5.6. Lobby Dataset

In this subsection, the results obtained using the *Lobby* dataset are showcased. This dataset is analysed through visual comparison with the Google Tango reconstruction, starting with the texture of the original images in Figures 25 and 26, then showing some detail using a colourmap, see Figure 27, and finally showcasing the fiducial marker removal results in Figure 28.

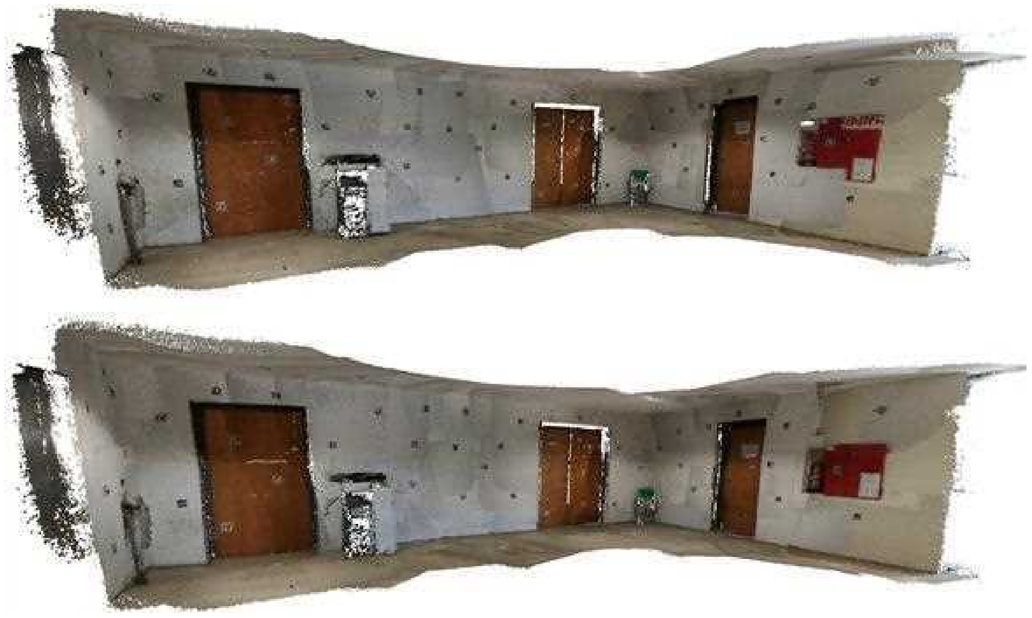

Figure 25. *Lobby* dataset coloured with texture obtained from original images: (**top**) Google Tango; (**bottom**) Our approach.

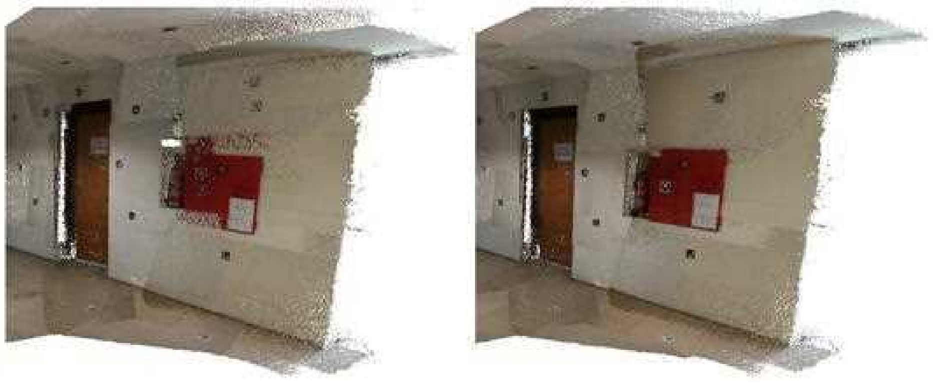

Figure 26. *Lobby* dataset coloured with texture obtained from original images (detail): (**left**) Google Tango; (**right**) Our approach.

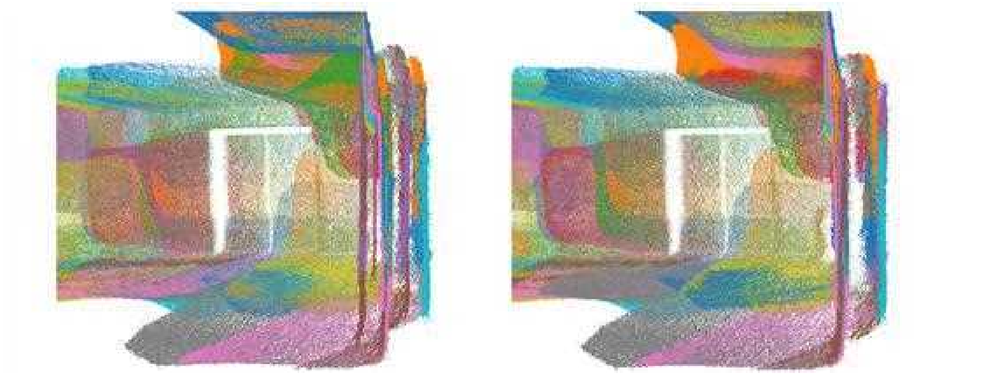

Figure 27. *Lobby* dataset point clouds coloured using a colourmap: (**left**) Google Tango; (**right**) Our approach.

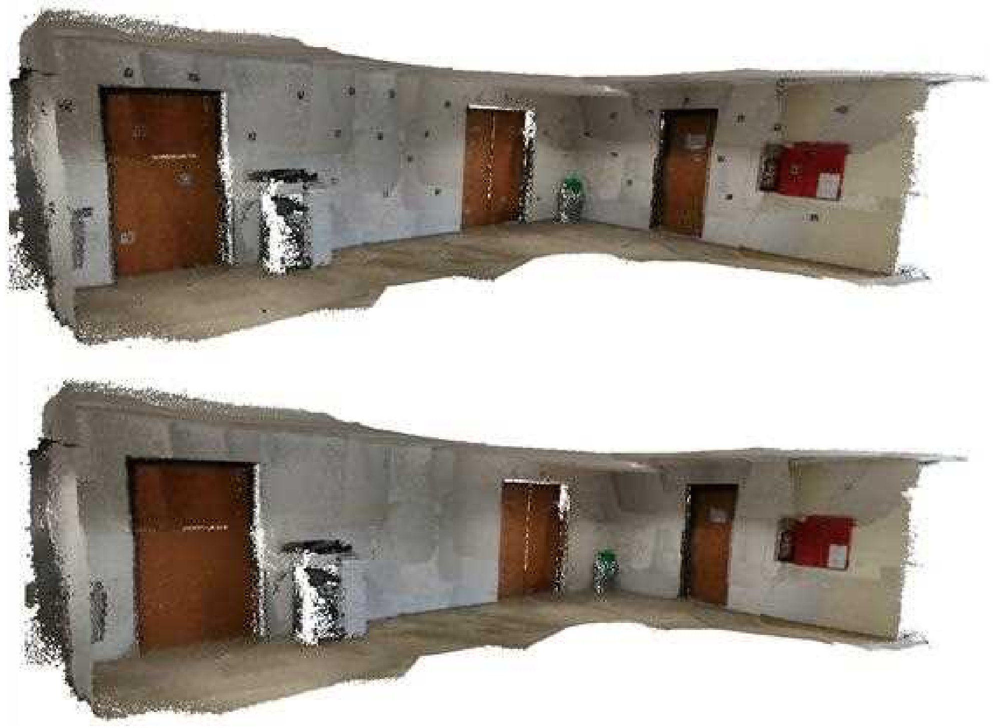

Figure 28. *Lobby* dataset coloured with texture: (**top**) obtained from original images; (**bottom**) obtained from restored images.

The *Lobby* dataset includes 84 RGB-D images and the alignment optimization was performed in 1m05s on an Ubuntu 18.04 laptop equipped with an Intel Core i7 7500U CPU and 8GB of RAM.

6. Conclusions

It is possible to observe a significant improvement in the registration of the 3D points clouds using our marker based optimization approach. Nevertheless, there are some limitations derived from the fact that this optimization only targets the registration problem, while having no influence on the geometry of each point cloud. In other words, the alignment of the point clouds may be improved, but if the point clouds are themselves distorted, which might occur with RGB-D data, the geometry of the scene may not improve significantly from this optimization.

The optimization implemented attempts to improve the results obtained from a 3D reconstruction using RGB-D cameras. If the dataset obtained from the reconstruction has poor alignment of the point clouds, even if only on some areas, then we should expect the optimization to improve the scene's geometry significantly. However, if a particular dataset happens to have very good alignment, the improvement will probably prove to be minor.

The API developed provides useful abstractions and an environment for future implementation of optimizations. It facilitates building upon and future work, allowing for the writing of more intuitive code. It also provides structure for a systematic approach to this kind of problem. We have already made use of these tools in other projects that require an optimization to be performed, such as the calibration of a set of sensor in an autonomous vehicle [65] and colour consistency correction in 3D reconstructions.

The removal of the fiducial markers from the texture of the scene was achieved, though it is important to note that the tool developed was meant for restoring homogeneous texture areas only, without a recognisable or sharp pattern. This tool allows for the creation of point clouds with texture free of markers, by utilising RGB-D information. The cross-inpainting technique was successful at removing non-detected aruco markers from the images, if there is good registration of the point clouds. For this reason, the technique also serves as a way to confirm that the optimization is working as intended, since its results, which depend on the good alignment of the point clouds, improve after the optimization is performed.

Results presented include several qualitative and quantitative assessments in three different datasets. Thus, we are convinced that the approach is robust and should in principle work for other datasets.

In future work, it would be interesting to generate meshes from the final point clouds, coloured with the texture free of fiducial markers. These meshes may be created through the utilization of ROS-based CHISEL [63], for instance.

There are some questions about the ideal density of fiducial markers in the scene, that is, how many markers should be spread, on average, to be seen in each image, to obtain the best results from the optimization. In the future, this could be analysed, producing guidelines regarding marker density and placement to optimize preparation time and alignment results.

Regarding the removal of fiducial markers, it would be interesting to investigate techniques that would allow the restoration of texture containing distinctive patterns, to enable its use in a wider range of environments and applications.

Author Contributions: Conceptualization, T.M., M.O. and P.D.; Investigation, T.M.; Methodology, T.M.; Project administration, P.D.; Resources, M.O. and P.D.; Software, T.M. and M.O.; Supervision, M.O. and P.D.; Validation, M.O. and P.D.; Writing—original draft, T.M; Writing—review & editing, T.M., M.O. and P.D.

Funding: This Research Unit is funded by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UID/CEC/00127/2019, as well as by project CYTED/TICs4CI - Aplicaciones TICS para Ciudades Inteligentes.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Achille, C.; Adami, A.; Chiarini, S.; Cremonesi, S.; Fassi, F.; Fregonese, L.; Taffurelli, L. UAV-Based Photogrammetry and Integrated Technologies for Architectural Applications—Methodological Strategies for the After-Quake Survey of Vertical Structures in Mantua (Italy). *Sensors* **2015**, *15*, 15520–15539. doi:10.3390/s150715520. [CrossRef]
2. Pérez, L.; Rodríguez, Í.; Rodríguez, N.; Usamentiaga, R.; García, D.F. Robot Guidance Using Machine Vision Techniques in Industrial Environments: A Comparative Review. *Sensors* **2016**, *16*, 335. doi:10.3390/s16030335. [CrossRef] [PubMed]
3. Zhang, Y.; Chen, H.; Waslander, S.; Yang, T.; Zhang, S.; Xiong, G.; Liu, K. Toward a More Complete, Flexible, and Safer Speed Planning for Autonomous Driving via Convex Optimization. *Sensors* **2018**, *18*, 2185. doi:10.3390/s18072185. [CrossRef]

4. Trinidad-Fernández, M.; Beckwée, D.; Cuesta-Vargas, A.; González-Sánchez, M.; Moreno, F.A.; González-Jiménez, J.; Joos, E.; Vaes, P. Validation, Reliability, and Responsiveness Outcomes Of Kinematic Assessment With An RGB-D Camera To Analyze Movement In Subacute And Chronic Low Back Pain. *Sensors* **2020**, *20*, 689. doi:10.3390/s20030689. [CrossRef] [PubMed]
5. Vázquez-Arellano, M.; Griepentrog, H.; Reiser, D.; Paraforos, D. 3-D Imaging Systems for Agricultural Applications—A Review. *Sensors* **2016**, *16*, 618. doi:10.3390/s16050618. [CrossRef] [PubMed]
6. Di Angelo, L.; Di Stefano, P.; Guardiani, E.; Morabito, A.E.; Pane, C. 3D Virtual Reconstruction of the Ancient Roman Incile of the Fucino Lake. *Sensors* **2019**, *19*, 3505. doi:10.3390/s19163505. [CrossRef] [PubMed]
7. Fan, H.; Yao, W.; Fu, Q. Segmentation of Sloped Roofs from Airborne LiDAR Point Clouds Using Ridge-Based Hierarchical Decomposition. *Remote Sens.* **2014**, *6*, 3284–3301. doi:10.3390/rs6043284. [CrossRef]
8. Henn, A.; Gröger, G.; Stroh, V.; Plümer, L. Model driven reconstruction of roofs from sparse LIDAR point clouds. *Int. J. Photogramm. Remote Sens.* **2013**, *76*, 17–29. doi:10.1016/j.isprsjprs.2012.11.004. [CrossRef]
9. Newcombe, R.A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A.J.; Kohi, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, Basel, Switzerland, 26–29 October 2011; pp. 127–136. doi:10.1109/ISMAR.2011.6092378. [CrossRef]
10. Han, J.; Shao, L.; Xu, D.; Shotton, J. Enhanced Computer Vision With Microsoft Kinect Sensor: A Review. *IEEE Trans. Cybern.* **2013**, *43*, 1318–1334. doi:10.1109/TCYB.2013.2265378. [CrossRef]
11. Remondino, F.; Nocerino, E.; Toschi, I.; Menna, F. A critical review of automated photogrammetric processing of large datasets. *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *XLII-2/W5*, 591–599. doi:10.5194/isprs-archives-XLII-2-W5-591-2017. [CrossRef]
12. Mousavi, V.; Khosravi, M.; Ahmadi, M.; Noori, N.; Haghshenas, S.; Hosseininaveh, A.; Varshosaz, M. The performance evaluation of multi-image 3D reconstruction software with different sensors. *Measurement* **2018**, *120*, 1–10. doi:10.1016/j.measurement.2018.01.058. [CrossRef]
13. Westoby, M.; Brasington, J.; Glasser, N.; Hambrey, M.; Reynolds, J. 'Structure-from-Motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology* **2012**, *179*, 300–314. doi:10.1016/j.geomorph.2012.08.021. [CrossRef]
14. Tsai, C.Y.; Huang, C.H. Indoor Scene Point Cloud Registration Algorithm Based on RGB-D Camera Calibration. *Sensors* **2017**, *17*, 1874. doi:10.3390/s17081874. [CrossRef] [PubMed]
15. Liu, H.; Li, H.; Liu, X.; Luo, J.; Xie, S.; Sun, Y. A Novel Method for Extrinsic Calibration of Multiple RGB-D Cameras Using Descriptor-Based Patterns. *arXiv* **2018**, arXiv:eess.IV/1807.07856.
16. Chen, C.; Yang, B.; Song, S.; Tian, M.; Li, J.; Dai, W.; Fang, L. Calibrate Multiple Consumer RGB-D Cameras for Low-Cost and Efficient 3D Indoor Mapping. *Remote Sens.* **2018**, *10*, 328. doi:10.3390/rs10020328. [CrossRef]
17. Diakité, A.; Zlatanova, S. First experiments with the tango tablet for indoor scanning. *ISPRS Anna. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *III-4*, 67–72. doi:10.5194/isprsannals-III-4-67-2016. [CrossRef]
18. Li, X.; Kesavadas, T. Surgical Robot with Environment Reconstruction and Force Feedback. In Proceedings of the 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Honolulu, HI, USA, 18–21 July 2018, pp. 1861–1866. doi:10.1109/EMBC.2018.8512695. [CrossRef]
19. Naseer, M.; Khan, S.; Porikli, F. Indoor Scene Understanding in 2.5/3D for Autonomous Agents: A Survey. *IEEE Access* **2019**, *7*, 1859–1887. doi:10.1109/access.2018.2886133. [CrossRef]
20. Li, L.; Su, F.; Yang, F.; Zhu, H.; Li, D.; Xinkai, Z.; Li, F.; Liu, Y.; Ying, S. Reconstruction of Three-Dimensional (3D) Indoor Interiors with Multiple Stories via Comprehensive Segmentation. *Remote Sens.* **2018**, *10*, 1281. doi:10.3390/rs10081281. [CrossRef]
21. Zhou, Y.; Zheng, X.; Chen, R.; Hanjiang, X.; Guo, S. Image-Based Localization Aided Indoor Pedestrian Trajectory Estimation Using Smartphones. *Sensors* **2018**, *18*, 258. doi:10.3390/s18010258. [CrossRef]
22. Pan, S.; Shi, L.; Guo, S. A Kinect-Based Real-Time Compressive Tracking Prototype System for Amphibious Spherical Robots. *Sensors* **2015**, *15*, 8232–8252. doi:10.3390/s150408232. [CrossRef]
23. Jamali, A.; Boguslawski, P.; Abdul Rahman, A. A Hybrid 3D Indoor Space Model. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *XLII-2/W1*, 75–80. doi:10.5194/isprs-archives-XLII-2-W1-75-2016. [CrossRef]

24. Zollhöfer, M.; Patrick Stotko, A.G.; Theobalt, C.; Nießner, M.; Klein, R.; Kolb, A. State of the Art on 3D Reconstruction with RGB-D Cameras. *Comput. Graph. Forum* **2018**, *37*, 625–652. [CrossRef]
25. Gokturk, S.; Yalcin, H.; Bamji, C. A Time-Of-Flight Depth Sensor - System Description, Issues and Solutions. In Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop, Washington, DC, USA, 27 June–2 July 2004.
26. Minou, M.; Kanade, T.; Sakai, T. Method of time-coded parallel planes of light for depth measurement. *IEICE Trans.* **1981**, *64* 521–528.
27. Will, P.; Pennington, K. Grid coding: A preprocessing technique for robot and machine vision. *Artif. Intell.* **1971**, *2*, 319–329. doi:10.1016/0004-3702(71)90015-4. [CrossRef]
28. Curless, B.; Levoy, M. A Volumetric Method for Building Complex Models from Range Images. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 4–9 August 1996; pp. 303–312.
29. Rusinkiewicz, S.; Hall-holt, O.; Levoy, M. Real-Time 3D Model Acquisition. *ACM Trans. Graph.* **2002**, *21*. doi:10.1145/566570.566600. [CrossRef]
30. Minguez, J.; Montesano, L.; Lamiraux, F. Metric-based iterative closest point scan matching for sensor displacement estimation. *IEEE Trans. Rob.* **2006**, *22*, 1047–1054. doi:10.1109/TRO.2006.878961. [CrossRef]
31. Manjunath, B.; Shekhar, C.; Chellappa, R. A New Approach to Image Feature Detection With Applications. *Pattern Recognit.* **1996**, *29*, 627–640. doi:10.1016/0031-3203(95)00115-8. [CrossRef]
32. Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision* **2004**, *60*, 91–110. doi:10.1023/B:VISI.0000029664.99615.94. [CrossRef]
33. Bay, H.; Tuytelaars, T.; Van Gool, L. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2006.
34. Patwary, M.J.A.; Parvin, S.; Akter, S. Significant HOG-Histogram of Oriented Gradient Feature Selection for Human Detection. *Int. J. Comput. Appl.* **2015**, *132*, 20–24. doi:10.5120/ijca2015907704. [CrossRef]
35. Canny, J. A Computational Approach To Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698. doi:10.1109/TPAMI.1986.4767851. [CrossRef]
36. Sobel, I. An Isotropic 3x3 Image Gradient Operator. Presentation at Stanford A.I. Project 1968, 2014.
37. Harris, C.; Stephens, M. A Combined Corner and Edge Detector. *Proc. 4th Alvey Vis. Conf.* **1988**, *1988*, 147–151. doi:10.5244/C.2.23. [CrossRef]
38. Smith, S.; Brady, M. SUSAN—A new approach to low level image processing. *Int. J. Comput. Vis.* **1997**, *23*, 45–78. doi:10.1023/A:1007963824710. [CrossRef]
39. Kenney, C.; Zuliani, M.; Manjunath, B. An Axiomatic Approach to Corner Detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005.
40. Lindeberg, T.; Li, M.X. Segmentation and Classification of Edges Using Minimum Description Length Approximation and Complementary Junction Cues. *Comput. Vis. Image Underst.* **1997**, *67*, 88–98. doi:10.1006/cviu.1996.0510. [CrossRef]
41. Rosten, E.; Drummond, T. Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2006.
42. Lindeberg, T. Feature Detection with Automatic Scale Selection. *Int. J. Comput. Vis.* **1998**, *30*, 77–116. doi:10.1023/A:1008045108935. [CrossRef]
43. Matas, J.; Chum, O.; Urban, M.; Pajdla, T. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. *Image Vis. Comput.* **2004**, *22*, 761–767. doi:10.1016/j.imavis.2004.02.006. [CrossRef]
44. Deng, H.; Zhang, W.; Mortensen, E.; Dietterich, T.; Shapiro, L. Principal Curvature-Based Region Detector for Object Recognition. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007.
45. Lindeberg, T. Discrete Scale-Space Theory and the Scale-Space Primal Sketch. Ph.D. Thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, 1991.
46. Jakubovic, A.; Velagic, J. Image Feature Matching and Object Detection Using Brute-Force Matchers. In Proceedings of the 2018 International Symposium ELMAR, Zadar, Croatia, 16–19 September 2018.
47. Muja, M.; Lowe, D. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *VISAPP* **2009**, *1*, 331–340.

48. Mount, D.; Netanyahu, N.; Le Moigne, J. Efficient Algorithms for Robust Feature Matching. *Pattern Recognit.* **2003**, *32*. doi:10.1016/S0031-3203(98)00086-7. [CrossRef]
49. Chen, C.S.; Hung, Y.P.; Cheng, J.B. RANSAC-Based DARCES: A new approach to fast automatic registration of partially overlapping range images. *IEEE Trans. Pattern Anal. Mach. Intell.* **1999**, *21*, 1229–1234. doi:10.1109/34.809117. [CrossRef]
50. Martínez, J.; González-Jiménez, J.; Morales, J.; Mandow, A.; Garcia, A. Mobile robot motion estimation by 2D scan matching with genetic and iterative closest point algorithms. *J. Field Rob.* **2006**, *23*, 21–34. doi:10.1002/rob.20104. [CrossRef]
51. Autodesk. ReCap: Reality capture and 3D scanning software for intelligent model creation. Available online: https://www.autodesk.com/products/recap/overview (accessed on 5 February 2020).
52. Alicevision. Meshroom: Open Source Photogrammetry Software. Available online: https://alicevision.org/#meshroom (accessed on 5 February 2020).
53. Thrun, S.; Leonard, J.J. Simultaneous Localization and Mapping. *Springer Handb. Rob.* **2008**, 871–889. doi:10.1007/978-3-540-30301-5_38. [CrossRef]
54. Jorge Nocedal, S.J.W. *Numerical Optimization*; Springer: Berlin/Heidelberg, Germany, 2000.
55. Agarwal, S.; Snavely, N.; M. Seitz, S.; Szeliski, R. Bundle Adjustment in the Large. In *Computer Vision—ECCV 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 29–42. doi:10.1007/978-3-642-15552-9_3. [CrossRef]
56. Harltey, A.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2006.
57. Romero Ramirez, F.; Muñoz-Salinas, R.; Medina-Carnicer, R. Speeded Up Detection of Squared Fiducial Markers. *Image Vision Comput.* **2018**, *76*. doi:10.1016/j.imavis.2018.05.004. [CrossRef]
58. Garrido-Jurado, S.; Muñoz-Salinas, R.; Madrid-Cuevas, F.; Medina-Carnicer, R. Generation of fiducial marker dictionaries using Mixed Integer Linear Programming. *Pattern Recognit.* **2015**, *51*. doi:10.1016/j.patcog.2015.09.023. [CrossRef]
59. Hornegger, J.; Tomasi, C. Representation issues in the ML estimation of camera motion. In Proceedings of Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999.
60. Schmidt, J.; Niemann, H. Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization. In *Vmv*, Aka GmbH: Stuttgart, Germany, 2001.
61. Oliphant, T. Python for Scientific Computing. *Comput. Sci. Eng.* **2007**, *9*, 10–20. doi:10.1109/MCSE.2007.58. [CrossRef]
62. FARO. Focus Laser Scanner Series. Available online: https://www.faro.com/products/construction-bim/faro-focus (accessed on 5 February 2020).
63. Klingensmith, M.; Dryanovski, I.; Srinivasa, S.; Xiao, J. Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially Hashed Signed Distance Fields. In *Robotics: Science and Systems XI*, RSS: Rome, Italy, 2015.
64. Rote, G. Computing the minimum Hausdorff distance between two point sets on a line under translation. *Inf. Process. Lett.* **1991**, *38*, 123–127. [CrossRef]
65. Oliveira, M.; Castro, A.; Madeira, T.; Dias, P.; Santos, V. A General Approach to the Extrinsic Calibration of Intelligent Vehicles Using ROS. In *Iberian Robotics Conference*; Springer: Cham, Switzerland, 2019.