STAG: Smart Tools and Applications in Graphics (2019)
M. Agus, M. Corsini and R. Pintus (Editors)

3DReg-i-Net: improving deep learning based 3D registration for a robust real-time alignment of small-scale scans

M. Lombardi[1,2], A. Riccardi[2], M. Savardi[1] and A. Signoroni[1]

[1]University of Brescia, Department of Information Engineering – Brescia, Italy
[2]Open Technologies Srl – Rezzato (BS), Italy

Abstract

We present 3DReg-i-Net, an improved deep learning solution for pairwise registration of 3D scans, which evolves the recently proposed 3DRegNet technique by Pais et al. This is one of the very first learning based algorithm aiming at producing the co-registration of two 3D views starting solely from a set of point correspondences, which is able to perform outlier rejection and to recover the registration matrix. We evolve the original method to face the challenging scenario of quick 3D modelling at small scales through the alignment of dense 3D views acquired at video frame-rate with a handheld scanner. We improve the system tracking robustness and alignment performance with a generalized input data augmentation. Moreover, working on suboptimal aspects of the original solution, we propose different improvements that lead to a redefinition of the training loss function. When tested on the considered scenario, the proposed 3DReg-i-Net significantly outperforms the prior solution in terms of accuracy of the estimated aligning transforms.

CCS Concepts

• ***Computing methodologies*** → *Artificial intelligence; Point-based models;* • ***Hardware*** → *Emerging technologies;*

1. Introduction

Pairwise as well as multiview 3D registration of point-based data (point clouds or range images) from RGBD sensors and 3D optical scanners define a large family of techniques widely studied in Computer Vision and Graphics [CM91, BM92, AMCO08, BSL11, PDS16]. However, progresses in the data acquisition technologies and the availability of new tools for data analysis result into the constant need to evolve and explore new solutions for new challenging needs. In particular, up to date handheld devices allow performing 3D scanning in a real-time fashion [INK*11], with higher flexibility and an improved user-experience with respect to a classic setup. Ideally, 3D real-time scanning should be robust to tracking loss and to the resulting misalignment, maintaining responsiveness at a high frame rate and ensuring metric accuracy and high density of the model. Although for pairwise view alignment only 3 points in common are enough, many solutions have been proposed for 3D keypoint detection and estimation of a redundant set of correspondences to counter the various non-idealities and challenges characterizing 3D point data acquisition and processing. To this aim, feature-based pipelines can provide generality of use, flexibility, robustness and speed. However, in a real working scenario, the presence of corrupted data typically affects the creation of the set of correspondences (*i.e.* the input of the network) and so there is a high percentage of outliers that must be detected in order to properly estimate the roto-translation transform **T** between 2 non-aligned point clouds or range images. To respond to the most advanced needs, approaches based on deep learning (DL) are becoming viable solutions. In the field of feature detection and matching some interesting solutions have been proposed both for images and 3D data [ZSN*17], while even more recently, a DL approach called 3DRegNet [PMR*19] has been proposed for direct estimation of **T** given a set of estimated input matches.

In the present work, starting from the potentialities of the 3DRegNet solution, we introduce and test improved solutions for real-time dense scan alignment scenarios, where the data flow comes from a handheld scanner and where fast estimation of the camera movement and robust tracking are main needs. More specifically, we highlight some problems and sub-optimalities of the original method and we propose alternative data handling and training solutions that lead to significant improvements of the co-registration accuracy.

1.1. Related Work

Unless some prior assumptions can be usually made on the range of pose variations between consecutive views, the computational complexity of unconstrained 3D view alignment, especially when dense sets of points are involved, can be prohibitive without exploiting a method for data reduction. To this aim, techniques that allow to extract a set of *features* which properly describe the dataset are particularly crucial. Compared to the 2D image domain, the extraction of feature points and the definition of informative and ro-

DOI: 10.2312/stag.20191363

bust 3D descriptors is more challenging because of the occlusions, clutters, missing structures, and noise affecting both structured and unstructured acquisitions of point sets. Many handcrafted feature based solutions have been proposed aiming at 3D view alignment, such as Spin Images [JH99] or Persistent Feature Histograms PFH [RBMB08] and its variants [RMBB08, RBB09]). Some of them additionally addressed the problem of the rotation invariance of the features such as RIFT [LSP05] and methods based on Point Pair Features (PPF) [DUNI10]. In [BSL11] a pairwise registration based on SIFT-like 3D features has been proposed for robust alignment of dense range scans, while in [PDS16] alignments driven by pose invariant Local Reference Frames, along with other different solutions, have been tested on different kinds of scan data. These are only few among many other works in the field of feature-based 3D scan alignment and the reader can take the literature review parts in [BSL12, PDS16] as a starting point for a broader overview.
More recently, learned feature descriptors based on deep representation learning solutions have emerged as an intriguing alternative, and some of them already outperformed the state-of-the-art of the handcrafted approaches. The main proposed solutions so far are 3DMatch [ZSN*17] and the PPF-based DL versions, *i.e.* PPFNet [DBI18b] and PPF-FoldNet [DBI18a].
Eventually, regardless the data type and the selected feature extraction approach, the result of a feature matching stage is a collection of correspondences to exploit for the alignment. A good method to deal with these sets of 3D points is the well known Iterative Closest Point algorithm [BM92], which tries to align two views by minimizing the point-to-point distance by iterative steps. A significant contribution was given by Chen and Medioni [CM91] with the introduction of a meta-view registration between a frame and a model and the alternative point-to-plane cost function. ICP, and its variants [RL01], is used as a direct alignment technique if the views are known to be sufficiently close, while they can have a refinement role after a so called "coarse" alignment is obtained after the filtering of the initial pool of correspondences that, in many practical applications, can be affected by the presence of many (sometimes a majority portion of) outliers. This is due to the noisy nature of the acquisitions, to the non sufficient geometric specificity and uniqueness of the geometric keypoints or a poor performance of the feature descriptors. A typical algorithm used to prune the outliers is RANSAC [FB81], despite other ranking based approaches proved to be good alternatives [BSL12]. These very effective methods are also quite time consuming (despite the existence of some faster variants [CMK03], [CM08]) because they are iterative and typically needs many iterations to work properly.
Deep learning solutions are currently emerging also to work on this final stage. Elbaz *et al.* [EAF17] were among the firsts to propose to feed a neural network auto-encoder with point clouds to register. More recently, an auto-encoder has been proposed also by Deng *et al.* [DBI19], relying on PPF features [DBI18b], [DBI18a] and PointNet [CSKG17] to be robust to the order of points in the input set. Finally, [PMR*19] suggested to use Multi Layer Perceptrons and ResNet [HZRS16] to classify a set of correspondences from two views and to directly co-register them. These methods, despite promising in terms of execution speed, still have to prove that they can be competitive to classical methods in terms of registration accuracy in real and challenging application scenarios, such as the one of interest here concerning small-scale object modeling by means of handheld real-time 3D scanners.

1.2. Contribution

We focus on the classification of a set of correspondences and the automatic pairwise registration using a deep learning based approach which promises to be fast and robust with respect to the amount of data to analyze.
Starting from the 3DRegNet approach [PMR*19] we characterize its behaviour and develop improved solutions in the context of data generated by a handheld scanning. At first we describe the device in use and the characteristics of our dataset. We also reflect on a critical aspect affecting a supervised learning approach in this context and present a data augmentation strategy to solve it. Then, we analyze the characteristics and evaluate the performance of the original 3DRegNet and we discuss the suboptimalities encountered during such assessment, especially related to the accuracy of the registration. Our main improvements concern a different output of the registration sub-block of the network for which we use a cost function involving quaternion-based representations for the inference of the rigid alignment transformation. Furthermore, we introduce an additional cost factor, working on the coherence among estimated surface normals, to help the network to rapidly learn how to regress the registration matrix. Finally we test and compare the new solutions with the original one and we evaluate how the different components concur to the final result.

2. Device and dataset

The launch of the first Microsoft Kinect [Zha12], followed in thxamples years by other affordable devices capable to acquire 3D depth maps at video frame rate, not only changed radically entertainment and video game experiences, but also opened new opportunities in many application fields and raised high interests in the Computer Vision research community at large [HSXS13]. In particular, real-time 3D object/scene reconstruction and modeling, pioneered by Izadi et al. [INK*11], developed toward solutions aiming to handle the whole set of problems and requirements characterizing real-time 3D scanning [DNZ*17]. Nowadays, depending on the accuracy, the dimensional scale, the performance, the engineering of the hardware, the software and many other application related aspects, the market can offer several solutions for real-time 3D scanning at different price ranges, from few hundreds to tens of thousands of euros. In our research activity we had the opportunity to work with the data flow produced by the *Insight 3* scanner, a handheld scanner by Open Technologies – FARO†, conceived for real-time object scanning. This is an unstructured-light optical scanner that, working in the infrared region, performs a stereo reconstruction of 3D points in space by means of two 1280×1024 pixel cameras (see Fig.1). The field of view of the scanner is between 150 and 500 mm, which is relatively narrow with respect to both the original structured-light Kinect and other ToF-based devices. It is conceived for close-range object scanning, producing

† https://www.opentechnologies.it/en/products/scanner-3d-eng/industrial-line/insight/

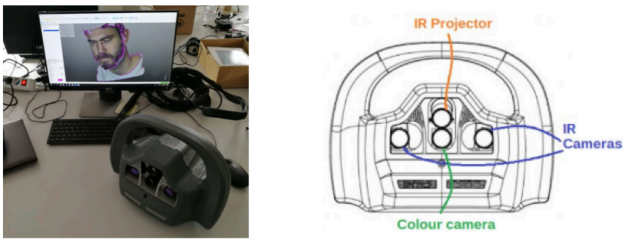


Figure 1: *The Insight scanner.*

a quite accurate dense data, with a point spacing in the range of $0.12-0.4$ mm and a depth resolution under 1 mm. Each frame is a Range Image (RI) and the 3D model is obtained by real-time fusion of the incoming RIs into a sparse volume by means of the Truncated Signed Distance Function (TSDF) [CL96]. This makes this scanner well targeted to many different applications, as reverse engineering, cultural heritage, entertainment (movies, videogames), medical CAD (orthotics and dentistry) and CAM.

Trying to improve the computational and robustness performance of these kind of devices is challenging and since our aim is to explore RI alignment solutions based on deep-learning approaches we need enough amount of data well representative of the device and of the task. Dataset creation is critical in this context, especially in the case of a supervised method, which usually requires application expertise and skilled personnel time to build the ground-truth. Moreover, working in a 3D domain increases the overall complexity of the building process. For this reasons, wherever possible, people usually prefer to work with available and already pre-processed data, which are then commonly used as reference by the research community. Two of them, the ICL-NUIM [HWMD14] and the SUN3D [XOT13] datasets, are in fact considered for the development of the 3DRegNet solution in [PMR*19]. The former is a big set of 8500 different pairs of connected point-clouds, synthetically created. The latter is a relatively smaller set, containing 3700 different pairs of real-scene connected point-clouds acquired with a commercial RGB-D infrared-based camera (Asus Xtion). Such a device is characterized by a wide detection range, a dense point-cloud production and, like others affordable RGB-D sensors, it is commonly used for performance capture or for environments of the scale of people and rooms. Since the main target of our scanner is different, more oriented to a smaller-scale object acquisition, we needed to create a new dataset made by data coming from the Insight scanner and representing typical acquisition targets, still referring to a *scene* every time we talk about one of the 3D model we produced. Our dataset presents different subjects (some examples are shown in Figure 2), from simple objects (*e.g.* a couple of helmets, a dummy, a piece of marble carved to have some geometric shapes on top, plastic models of mechanical components, etc.) to human body parts (head, hands, arms, bust). Each scene is composed by multiple RIs, also called *fragments*, typically acquired in a next to each other sequence, but possibly including pose discontinuities due to tracking failures or to the user need to acquire more than one sequence of fragments to complete the 3D model. The ground truth is created by pairwise alignment of subsequent fragments using a reference feature based alignment technique [BSL11,BSL12], which has been recognized to be particularly suited for the alignment of dense point sets as the ones generated by the Insight scanner. As well as helping in creating the alignment ground truth, this method directly generates the set of features and correspondences which are required by our approach, similarly to what happens for 3DRegNet. Feature-points are extracted according to a robust multiscale Difference-of-Gaussians (DoG) processing of the RIs that takes into account various data non-idealities, and feature descriptors consists of 192-dimensional signatures related to both local geometric saliency and normal field measures. Then feature matching and correspondence selection mechanisms operates on pairs of fragments allowing to filter out the wrong matches in an iterative fashion guided by distance-based scores, until a set of (decidable) N best ranked correspondences $(\mathbf{X},\mathbf{Y}) \in \mathbb{R}^{N\times 3\times 3}$ is extracted, where $\mathbf{X} \in \mathbb{R}^{N\times 3}$ is a set of 3D coordinates taken from the first fragment and $\mathbf{Y} \in \mathbb{R}^{N\times 3}$ is taken from the second one. We consider the latter as the *floating* fragment that we want to align to the former, the *fixed* one. Relying on these correspondences, a rigid transformation $\mathbf{T}_{gt} \in \mathbb{R}^{4\times 4}$, defined as the combination of a rotation $\mathbf{R}_{gt} \in \mathbb{R}^{3\times 3}$ and a translation vector $\mathbf{t}_{gt} \in \mathbb{R}^3$, can be computed. In [BSL11, BSL12] this is done by a fast skimming approach able to effectively skim outliers and that, in this context, proved faster and better performance compared to the widely adopted RANSAC approach. We adopt $\mathbf{T}_{gt}$ as the ground-truth rigid transformation while, aiming at estimating it through a Deep Learning approach, by starting from a set of N correspondences, we take as input the N best correspondences generated by the first ranking operated in [BSL11, BSL12]. Even proceeding this way, we have to stress the fact that, despite in [PMR*19] the inlier ratio at the input is considered to be 50%, which is a good assumption to not bias the training, this is however quite far from what happens in real working scenario, where instead the outliers percentage can be much higher.

By construction the stream of pairwise fragments, which is directly taken from the scanner, is mainly composed by nearby fragments (taken few tens of milliseconds apart) characterized by small viewpoint rotations and translations. Since our goal is to learn the system to quickly recover even from tracking failure (which may involve relatively larger camera viewpoint changes), we then apply a data augmentation to our dataset. To do this, we randomly rotate and translate respectively with a span of $\pm 30^\circ$ and of ± 30 mm the set of points $\mathbf{Y}$, *i.e.* $\mathbf{Y}' = \mathbf{T}_{rnd}\mathbf{Y}$ and we recompute the ground truth rigid transformation $\mathbf{T}_{gt}$ as the composition of the original transformation with this new random one:

$$\mathbf{T}'_{gt} = \mathbf{T}_{gt} \cdot \mathbf{T}^{-1}_{rnd} \tag{1}$$

In such a way, we are now able to feed the network with a set of fragments that are misaligned from few millimeters up to some centimeters and eventually highly rotated on the three axes, so that to learn to handle both incremental alignments and tracking loss recovery instances.

3. Methods

Our main goal is to design and test DL-based solutions for faster estimation of pairwise alignment transforms. The only and very recent contribution by Pais (et al.) that works on this idea is called

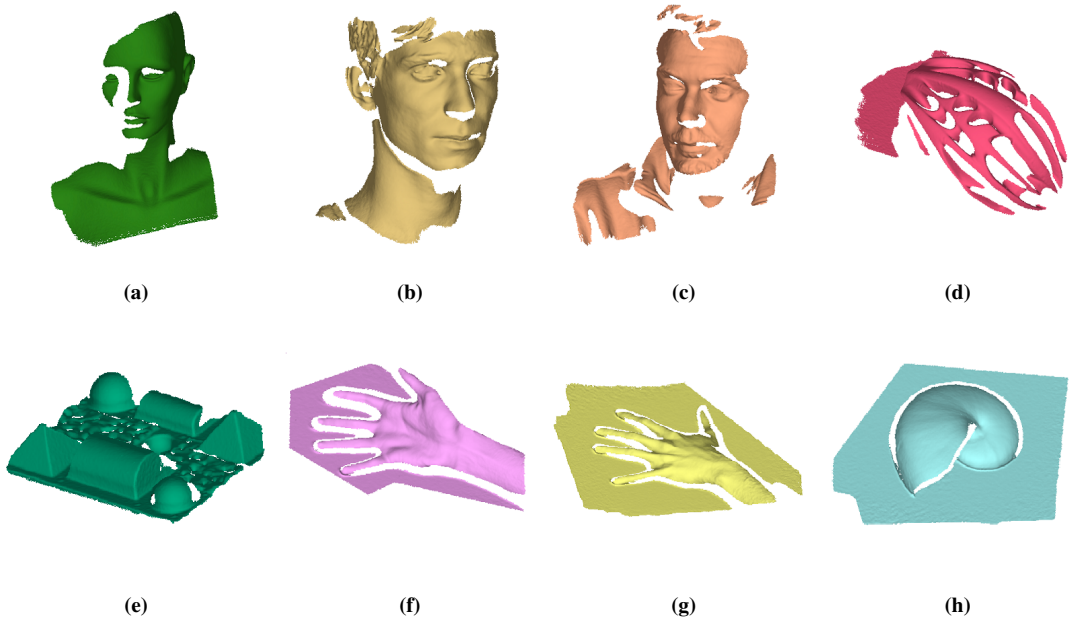


Figure 2: *Examples of scenes in our dataset. The targets are many, both objects, for instance a dummy (a), a piece of marble with 3D solids on top (e), a helmet (d) and a seashell (h) and human body parts like heads (b), (c) and hands (f), (g).*

3DRegNet [PMR*19] and has been tested on benchmark datasets which are only partially representative of our working scenario. Thus, while adopting 3DRegNet as a starting point for our work, we also critically analyze possible weakness of the method and propose new and improving solutions. In its turn, 3DRegNet is inspired and extends a previous work by Yi *et al.* [YTO*18], where a neural network is fed with sets of 2D pixel correspondences extracted from many couples of images with the aim to classify each correspondence as a valid or invalid one. This is done grounding on the *ResNet* architecture [HZRS16], a state-of-the-art deep learning solution allowing to grow the number of inner layers without increasing the training complexity. Notably, other than introducing mere 2D to 3D extension rearrangements, 3DRegNet [PMR*19] expands the network with a new registration block, with the purpose of inferring the 3D alignment transformation in a straightforward way. Here we confine ourselves in giving an essential overview of 3DRegNet, which is functional to the understanding of our contribution, while referring the readers to the original paper for more details. A schematic representation of the network is depicted in Figure 3. The input consists of K pairs of scans, each one coming with N pairs of possibly corresponding 3D points, *i.e.* $\{(\mathbf{x},\mathbf{y})_i\}^k$, where $i \in \{1,\dots,N\}$ and $k \in \{1,\dots,K\}$. The first inner block is demanded to the classification task: each correspondence is passed through a fully connected layer with *ReLU* activation functions (where $ReLU(x) = \max(0,x)$) that returns a $N \times 128$ output. Then a 12 ResNet block sequence with weight-shared fully connected layers processes the data returning again a $N \times 128$ output. Finally, a fully connected layer activated with *ReLU* and a $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ unit produces the classification output, such that for each correspondence at the input we end up with an output value interpretable as the probability for that correspondence to be a valid one (that is an inlier).

Additionally, while traversing the chain of ResNets that we mentioned above, every block returns a temporary $N \times 128$ output that acts as input both for the next block and for the skip connection inside ResNet workflow. 3DRegNet takes such outputs, performs a max pooling and concatenates the features. As explained both in [YTO*18] and [PMR*19], such *Context Normalization* helps to normalize and to fix the necessary number of features. This new set of data is then passed through a convolutional layer and to two fully connected layers, before returning the final output of the registration block. The output represents the parametrization of the learned rigid transformation that aligns the two views, $\hat{\mathbf{T}}$, consisting in the combination of a rotation matrix $\hat{\mathbf{R}}$ and a translation vector $\hat{\mathbf{t}}$. Its size depends upon the desired representation of $\hat{\mathbf{R}}$: in Lie algebra only three parameters are required, four in case of quaternion representation and nine for the linear matrix. In [PMR*19] the author works with Lie algebra, because of more compact representation and better experimental alignment performance. After the design of the network, the second critical aspect regards the choice of the loss function. Since 3DRegNet is a combination of classification and registration sub-blocks, the loss reflects this structure. The *classification loss* is defined via the commonly used cross-entropy function $H(y,\hat{y})$:

$$L_c^k = \frac{1}{N}\sum_{i=1}^{N} \gamma_i^k H\left(l_i^k, \sigma(o_i^k)\right) \tag{2}$$

which, for the k-th fragments pair, computes the mean of the cross-entropy over the N correspondences, evaluated for the ground truth label l_i^k and the output of the classification block (o_i^k) activated with a sigmoid function σ. The *registration loss* instead is a simple formulation of the ℓ_1 distance metric function in 3D space. An ℓ_1 dis-

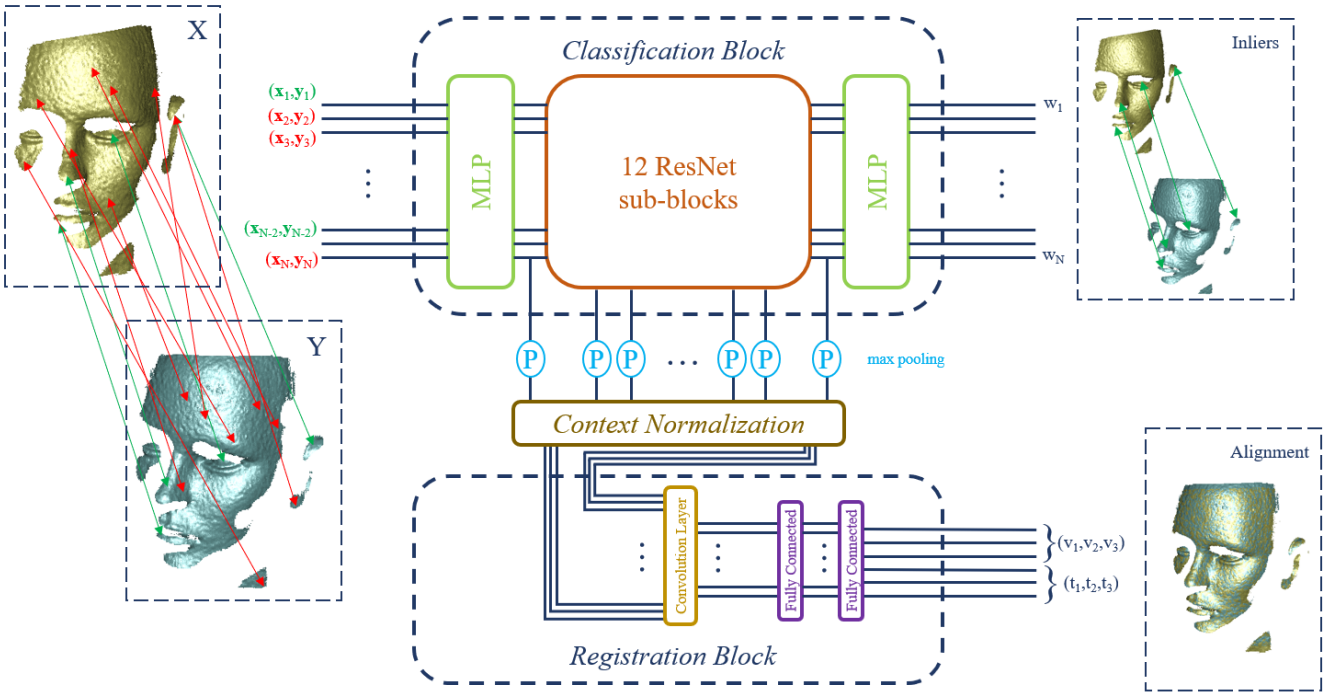


Figure 3: *The original 3DRegNet [PMR*19] architecture, where in our implementation the input set of N correspondences extracted from a pair of range images (on the left) are obtained by the method described in [BSL11]. The output is split in two sub-blocks: the first block is devoted to classification and produces N weights that represent the likelihood of the relative correspondence to be valid or not. The second block is devoted to the inference of the rigid transformation that is necessary to align the two scans. The size of such output depends on the parametrization of the alignment transform. In [PMR*19] it is a 6-tuple (3 components for the rotation matrix represented with Lie algebra and 3 for the translation vector).*

tance was experimentally chosen among other metrics because it produced best results. The loss is then defined as the point-wise distance between $\mathbf{x}$ and the transformed version of the corresponding one $\mathbf{y}$ after applying the learned transformation $\hat{\mathbf{T}}$:

$$L_r^k = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i^k - (\hat{\mathbf{R}}\mathbf{y}_i^k + \hat{\mathbf{t}})\|_1 \tag{3}$$

However, the choice of such registration loss seems to be sub-optimal to us, since by considering the distance between all corresponding point pairs, we also include all the outliers, which could likely mislead the learning process, given that the network keeps trying to minimize the distance between not corresponding points. In the end, the resulting alignment could easily suffer from drift issues. A first alternative could be using a weighted least square solution (although according to the author of 3DRegNet it performs worse than the ℓ_1 distance) or a filtered version of the point-to-point distance with removal of the outliers by means of the ground-truth labels. However, while testing alternatives, we observed for all solutions some issues related to the learning goodness of the translation vectors of the rigid transformation. In particular, they rapidly fall into a local minimum during the gradient descent and do not recover from it. Since the reasons for this could be many, comprising an ill posed definition of the loss, to break it down, we modified the way we compute L_r. We leverage the fact that we also have the ground truth transformation $\mathbf{T}_{gt}$ available, and in particular the rotation. Thus, as in [DBI19], we can define a cost function to directly learn to regress the rotation. Similarly to [DBI19] and [KC17] we use a quaternion-based parametrization of the rotation, because of its continuous and differentiable formulation, which fits well for deep learning training algorithms, and because of the reduced number of parameters to regress. So the *rotation loss* for k-th pair of fragments is now formulated as:

$$L_q^k = \left\| \mathbf{q}_{gt} - \frac{\hat{\mathbf{q}}}{\|\hat{\mathbf{q}}\|} \right\| \tag{4}$$

where $\|\cdot\|$ is an ℓ_2 norm. Here the normalization of the inferred $\hat{\mathbf{q}}$ is required for guaranteeing valid quaternion rotations. The output of the registration sub-block is then reduced to four parameters only, as depicted in Figure 4.

Furthermore, since quaternions are non-injective ($-\mathbf{q}$ is identical to $\mathbf{q}$), we prefer to constrain $\hat{\mathbf{q}}$ to one hemisphere by adding a penalty every time the control statement $\mathbf{q}_{gt} \cdot \hat{\mathbf{q}} > 0$ returns false:

$$L_p^k = \begin{cases} 0, & \text{if } \mathbf{q}_{gt}^k \cdot \hat{\mathbf{q}}^k > 0. \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

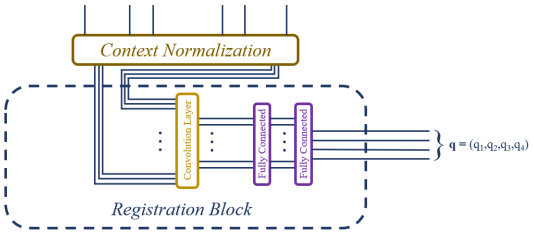


Figure 4: *Our output at the registration sub-block which now is focused only on the inference of the rotation matrix, parametrized with a 4-tuple quaternion.*

We focus our attention on reducing the distance to the ground-truth rotation, since learning the translation vector can be avoided and $\mathbf{t}$ can be easily retrieved by looking at the definition of the rigid transformation itself (as also pointed out in [DBI19]). Indeed, once we have a point $\mathbf{x_i}$, the corresponding point $\mathbf{y_i}$ and the estimated rotation $\mathbf{R}$ that aligns $\mathbf{y_i}$ to $\mathbf{x_i}$, we can retrieve the translation for this point as $\mathbf{t_i} = \mathbf{x_i} - \mathbf{R}\mathbf{y_i}$. Therefore, by restricting ourselves to solely the correspondences that are estimated to be inliers (with sufficiently high confidence level) by the network, the translation can be computed as the average translation over the set of N_I inliers x_j^I:

$$\hat{\mathbf{t}} = \frac{1}{N^I}\sum_{j=1}^{N^I}(\mathbf{x_j^I} - \hat{\mathbf{R}}\mathbf{y_j^I}) \tag{6}$$

This allows reducing the network complexity and we can focus to learn the rotation as best as possible. In this perspective, we add a last component to the registration loss by also considering a constraint on the normals, since we have them ready from the acquisition session. In particular, we define the *normals rotation loss* for the generic k-th pair as:

$$L_n^k = \frac{1}{N^I}\sum_{j=1}^{N^I}(- < \mathbf{q}_{gt}\mathbf{n}_{y_j} \cdot \hat{\mathbf{q}}\mathbf{n}_{y_j} >) \tag{7}$$

In such a way, the total cost increases whenever the angle between the normals that have been rotated respectively with the ground-truth and the learned rotations, is getting further from zero.
Eventually, the final registration loss is defined as:

$$L_r^k = L_q^k + L_p^k + L_n^k \tag{8}$$

Together with (2), by computing the mean over the K different pairs, they form the *total loss*:

$$L = \alpha L_c + \beta L_r \tag{9}$$

where α and β are two hyper-parameters of the network, in the same fashion as 3DRegNet, that are used to properly tune the impact of each sub-loss on the overall cost. Moreover, as described in [PMR*19], we also use the *Curriculum Learning* data augmentation strategy: in each training epoch we rotate every incoming batch of data by increasing angles from 0° to 50° until we get to the half running epoch and then we keep rotating but backward, with decreasing angle from 50° to 0°. This strategy is useful to further add new rotations to the network and to not repeat the data one epoch after the other, so that it allows to run more iterations over the dataset avoiding the rapid overfitting.

4. Experiments

Our aim is the performance comparison of the new version of the network with the original 3DRegNet in the perspective of its possible integration within the scanner modeling pipeline.

Evaluating metrics

In order to perform the assessment, we need to define some scores. Since the network contains two sub-blocks, we use different metrics for both the two components. For the classification test we rely on two well known parameters, *i.e.* its *accuracy* and *precision*, typically defined as:

$$accuracy = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}\ ,\ precision = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

where TP, TN, FP, FN stands for true positives, true negatives, false positives and false negatives respectively.
Similarly to 3DRegNet, we also use two metrics for the registration test. Such metrics describe the distances of the learned rotation and the learned translation from their ground-truth counterpart. In particular, the rotation accuracy is defined in [MSKS03] as:

$$\delta_{\mathbf{R}} = \cos^{-1}\left(\frac{\mathrm{trace}(\mathbf{R}^{-1}\mathbf{R}_{gt})-1}{2}\right) \tag{10}$$

which basically represents, via Lie algebra, the angle (in degrees) required to align the two rotations under comparison. For translations we use the euclidean distance between the two vectors:

$$\delta_{\mathbf{t}} = \|\mathbf{t}_{gt} - \hat{\mathbf{t}}\| \tag{11}$$

Finally, since we are also interested into the evaluation of the alignment process, we additionally compute the point-to-point euclidean distance for the set of true correspondences (inliers), having cardinality $N^I \leq N$. In particular, differently from (3) we do not use both the points $\mathbf{x^I}$ and $\mathbf{y^I}$ of each inlier correspondence. In fact, since we have an intrinsic error in the re-projection, which is due to the scanner depth resolution (around $1mm$), we prefer to consider only the floating points (*i.e.* those belonging to the second scan of the pair), and rotate them with both the ground-truth and the learned rigid transformations and eventually compute their euclidean distance. In such a way, the registration distance

$$\delta_{\mathrm{reg}} = \frac{1}{N^I}\sum_{j=1}^{N^I}\|(\mathbf{R}_{gt}\mathbf{y}_j^I + \mathbf{t}_{gt}) - (\hat{\mathbf{R}}\mathbf{y}_j^I + \hat{\mathbf{t}})\| \tag{12}$$

is more accurate and informative to us.

Training details

We used Tensorflow [AAB*16] (version 1.13 with GPU support) to implement the network, together with Tensorboard to check the evolution of the learning process. In total we acquired 29 scenes, containing 4682 range images. We selected 25 of them (4150 fragments) for training, while we kept the other 4 (532 fragments) for testing. We augmented the training dataset up to 20000 pairs of

fragments, taking from each of them the best $N = 1000$ correspondences generated with the method of [BSL12]. We trained the network with a batch size of 16 for 1200 epochs (so a total of 1.5M iterations). We used Adam Optimizer [KB14] with a learning rate equal to 10^{-4}. For the first 20000 iterations the registration loss was set to 0 in order to let the network focus on the classification task. We used the same batch normalization strategy of 3DRegNet and we selected the hyper-parameters α and β such that the losses had the same order of magnitude in both the two setups. Then for the original 3DRegNet we changed β from 10^{-3} to 10^{-5}, while we set to 1 both the parameters in our setup. Each training was performed with an INTEL i7-8700K and a NVIDIA GeForce 1080 with 8Gb of memory and it took around 2 days of computation (8 iterations per second).

4.1. Comparison with 3DRegNet

The first test that we run is finalized to compare the behaviour of our 3DReg-i-Net implementation against the original 3DRegNet solution [PMR*19]. We use the dataset presented in Section 2 and the training setup mentioned above. Table 1 shows the results. Given the equivalence (architecture and loss) of the correspondence

Method	Classification		Rotation [deg]		Translation [mm]		Time [ms]
	Accuracy	Precision	Mean	Median	Mean	Median	
[PMR*19]	0.98	0.75	7.35	5.73	11.75	10.33	5.8
3DReg-i-Net	0.98	0.75	3.75	2.97	5.95	3.44	5.4

Table 1: *Comparison between 3DRegNet original implementation [PMR*19] and our improved version.*

classification part between the two solutions, we can see, as expected, that the inlier/outlier classification results are almost the same for the two implementations. In particular, the accuracy is high and very similar to the values reported in the original study. We do not have a correspondent reference score for the precision to compare with, but we evaluate it because we want to know how good the network is at detecting the inliers only. The score is remarkable to us if we consider the very low percentage of inliers within the test set (see Figure 5 for the inliers distribution), well below the 50% and much closer to a real case, combined with the high processing speed (around 5ms) to obtain both this classification and the alignment estimation. The difference between the two solutions emerges by looking at the registration result. The values this time are dissimilar to the ones reported in [PMR*19] but we assume that this is due to the different resolutions of the scanners adopted for the creation of the dataset (we also recall the fact that they use a synthetic dataset for the training which by definition has a perfect ground-truth to rely on). Nevertheless, by comparing the two versions with our data it is evident how the change of the cost function had a benefit in the inference of the transformation matrix: both the registration and translation errors are halved. As expected, the two methods have a similar processing speed, which is remarkably fast and could be also compliant with the real-time constraint of the Insight 3 scanner. In Figure 6 two test cases are depicted. We have two pairs of initially misaligned fragments representing a dummy portion (Subfigs.(6a) - (6c)) and a piece of marble carved with geometric figures (Subfigs.(6d) - (6f)). It is clear how in both cases 3DReg-i-Net outperforms the original 3DRegNet at registering the

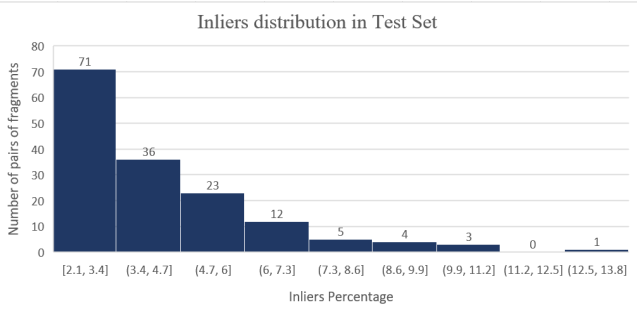


Figure 5: *Inliers distribution in the Test dataset. On the vertical axis we see how many pair of scans have an inliers percentage (among 1000 correspondences) in the range which is expressed on the horizontal axis. The minimum value is* 2.1% *while the maximum is* 12.6%*.*

two views. In particular, the latter seems to not recover properly the translation vector. Furthermore, in Table 2 we report the average values of eq.(12) over the entire test set. Again, our method is clearly the most accurate.

Method	Registration Distance [mm]	
	Mean	Median
[PMR*19]	12.940	11.911
3DReg-i-Net	5.859	3.614

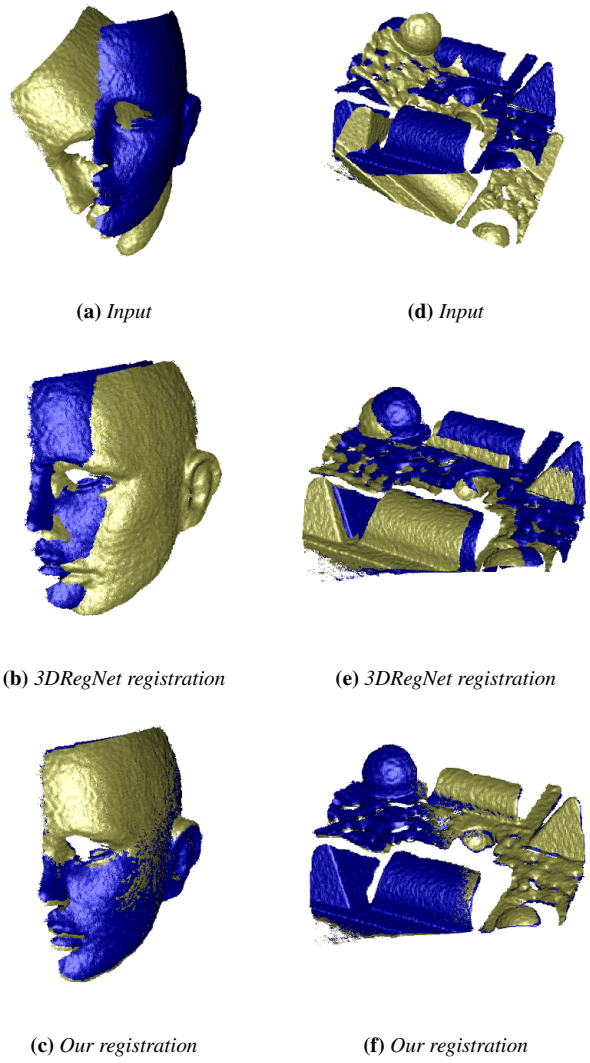
Table 2: *Comparison between the original 3DRegNet and our solution by evaluating the registration distance as defined in (12).*

4.2. 3DReg-i-Net ablation study

Now we want to evaluate how the new different components affects the overall performance of the network. First we look at the loss function in the registration sub-block, then we check the influence of the augmentation introduced in the dataset to feed the network with more transformations and finally we run a stress test on the network by varying the number of correspondences to work with.

Registration loss

We train again our network two times, turning off one by one the two main components of the registration loss, *i.e.* the costs defined respectively on the quaternions distance minimization (L_q and L_p) and the normals rotation coherence (L_n). In Table 3 we report the results after testing these setups. As we can see, the accuracy of the classifier did not change and this, again, can be explained by the fact that we did not modified anything in that specific sub-block. Instead the registration presents some variations: in particular, it is highlighted how the rotation loss component is more effective than the one using the normals . This is quite predictable, since the former loss is specifically oriented to directly minimize the rotation, while the latter is working as an additional constraint and so it reduces the searching space for the optimizer to find the minimum of the cost. In fact, during the training sessions we observed, via Tensorboard, a more rapid convergence to the optimal result for the full implementation, with respect to the version without the constraint



(a) *Input* **(d)** *Input*

(b) *3DRegNet registration* **(e)** *3DRegNet registration*

(c) *Our registration* **(f)** *Our registration*

Figure 6: *Alignment results on two pairs of fragments (6a), (6d) using both the two solutions, i.e. 3DRegnet (6b), (6e) and our version with a new loss (6c),(6f). The new version is clearly more accurate with respect to 3DRegNet, which in particular is lacking a good estimation of the translation vector for the alignment.*

on the rotation of the normals. The full implementation get a better result overall while decreasing at a faster rate, as reported in Table 4.

Dataset augmentation

In Section 2 we explained the need of augmenting the original dataset, which was lacking of relatively large rotations and translations, by applying random rigid transformations to the set of floating points. In Table 5 we report the result after we trained the network using the same setup as above but with a smaller dataset that does not have any synthetic augmentation. The test set is the same from the previous section and so it has instead large camera movements to infer. The results show how the trained network missing the augmented data failed to recover challenging rotations and translations, endorsing the idea that a supervised method needs an input as generic as possible to properly tackle every scenario.

Losses	Classification		Rotation [deg]		Translation [mm]		Time [ms]
	Accuracy	Precision	Mean	Median	Mean	Median	
No (L_q, L_p)	0.98	0.73	8.17	7.10	8.509	6.445	5.4
No L_n	0.98	0.73	4.23	3.43	6.22	3.66	5.5
All losses	0.98	0.75	3.75	2.97	5.95	3.44	5.4

Table 3: *Comparison between training sessions with different losses. The full implementation i.e. the one with all the cost functions considered, produces the best result overall.*

Iterations	Rotation Mean [deg]	
	No L_n	**Full** L_r
100K	6.57	5.59
200K	5.97	4.59
400K	4.96	4.19
600K	4.64	4.1
800K	4.49	3.96
1000K	4.39	3.91

Table 4: *Mean of the rotation error evaluated through the learning process at different iterations. Comparison between the implementation without the constraint on the rotation of the normals and the loss containing all the components.*

Dataset	Rotation [deg]		Translation [mm]	
	Mean	Median	Mean	Median
Augmented	3.75	2.97	5.95	3.44
Not Augmented	23.63	27.43	21.29	15.13

Table 5: *Results for the registration after training the network either with or without the augmented dataset.*

Number of correspondences

Eventually, we are interested to know how the network is sensitive to a varying number of correspondences. We first run a stress test by removing a fixed number of correspondences from the original test set but keeping fixed the inlier and we report the results in Table 6. It can be noticed how the behaviour is very similar to the one presented in [PMR*19]. Indeed we have a degradation of the registration accuracy when the number of correspondences decreases. On the contrary the accuracy of the classifier is not affected. In few cases with the smaller set, it happened that there were too few inliers at the input (around 5 over 250 total correspondences) such that the classifier failed to find any inlier. So this appears to be a lower bound limit for our network at the moment. Then, we tried to remove only the outliers and consequently to evaluate how the network responds to an increasing inliers ratio. As reported in Table 7, we see that the precision scores increase coherently with such ratio, as expected from a dataset with less outliers in it.

5. Conclusions

We investigated the problem of the co-registration of 3D views in a real-time 3D scanning scenario. Our 3DReg-i-Net grounds on

Correspondences	Rotation [deg]		Translation [mm]		Classification	Time [s]
	Mean	Median	Mean	Median	Accuracy	
25%	5.63	4.27	7.857	5.555	0.97	5.22
50%	4.39	3.44	6.755	4.473	0.98	5.24
75%	3.96	3.16	6.202	3.719	0.98	5.27
100%	3.75	2.97	5.955	3.438	0.98	5.4

Table 6: *Results after testing our network with an increasing number of correspondences (ratio over the total number available, which is $N = 1000$).*

Inliers percentage	Classification Precision
4%	0.74
8%	0.81
12%	0.85
16%	0.88

Table 7: *Variation of inliers percentage (on average) in the test set and corresponding evaluation of the precision by the classifier.*

a recently proposed deep neural network solution called 3DRegNet [PMR*19]. Given a set of 3D point correspondences, the method aims to solve both the outlier rejection and the inference of the rigid transformation matrix that aligns the views. We highlighted the critical aspect regarding the accuracy of the process on a new dataset created using a handheld 3D optical scanner. This new dataset was also augmented by applying a random set of rotations and translations, in order to be as generic as possible with respect to the possible movements of the scanner. This way 3DReg-i-Net demonstrated to better recover the pose even facing large misalignment. Moreover, we deduced some weakness of 3DRegNet for the target application which were tackled by reformulating the registration cost function of the network, enhancing the focus on the rotation matrix inference and adding a constraint based on the space of rotations of the correspondence point normals. The results highlighted how the new loss is more informative to the network which is now more accurate when performing the registration.

References

[AAB*16] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I. J., HARP A., IRVING G., ISARD M., JIA Y., JÓZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D. G., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P. A., VANHOUCKE V., VASUDEVAN V., VIÉGAS F. B., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR abs/1603.04467* (2016). URL: http://arxiv.org/abs/1603.04467, arXiv:1603.04467. 6

[AMCO08] AIGER D., MITRA N. J., COHEN-OR D.: 4-points congruent sets for robust pairwise surface registration. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 85:1–85:10. doi:10.1145/1360612.1360684. 1

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (Feb 1992), 239–256. doi:10.1109/34.121791. 1, 2

[BSL11] BONARRIGO F., SIGNORONI A., LEONARDI R.: A robust pipeline for rapid feature-based pre-alignment of dense range scans. In *2011 International Conference on Computer Vision* (Nov 2011), pp. 2260–2267. doi:10.1109/ICCV.2011.6126505. 1, 2, 3, 5

[BSL12] BONARRIGO F., SIGNORONI A., LEONARDI R.: Multi-view alignment with database of features for an improved usage of high-end 3D scanners. *EURASIP Journal on Advances in Signal Processing 2012*, 1 (2012), 148. doi:10.1186/1687-6180-2012-148. 2, 3, 7

[CL96] 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 York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 303–312. doi:10.1145/237170.237269. 3

[CM91] CHEN Y., MEDIONI G.: Object modeling by registration of multiple range images. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation* (April 1991), pp. 2724–2729 vol.3. doi:10.1109/ROBOT.1991.132043. 1, 2

[CM08] CHUM O., MATAS J.: Optimal randomized RANSAC. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 8 (Aug 2008), 1472–1482. doi:10.1109/TPAMI.2007.70787. 2

[CMK03] CHUM O., MATAS J., KITTLER J.: Locally optimized RANSAC. In *Pattern Recognition* (Berlin, Heidelberg, 2003), Michaelis B., Krell G., (Eds.), Springer Berlin Heidelberg, pp. 236–243. doi:10.1007/978-3-540-45243-0_31. 2

[CSKG17] CHARLES R. Q., SU H., KAICHUN M., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3D classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), pp. 77–85. doi:10.1109/CVPR.2017.16. 2

[DBI18a] DENG H., BIRDAL T., ILIC S.: PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. In *Computer Vision – ECCV 2018* (Cham, 2018), Ferrari V., Hebert M., Sminchisescu C., Weiss Y., (Eds.), Springer International Publishing, pp. 620–638. doi:10.1007/978-3-030-01228-1_37. 2

[DBI18b] DENG H., BIRDAL T., ILIC S.: PPFNet: Global context aware local features for robust 3D point matching. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (June 2018), pp. 195–205. doi:10.1109/CVPR.2018.00028. 2

[DBI19] DENG H., BIRDAL T., ILIC S.: 3D local features for direct pairwise registration. *CoRR abs/1904.04281* (2019). URL: http://arxiv.org/abs/1904.04281, arXiv:1904.04281. 2, 5, 6

[DNZ*17] DAI A., NIESSNER M., ZOLLHÖFER M., IZADI S., THEOBALT C.: Bundlefusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph. 36*, 3 (May 2017). doi:10.1145/3054739. 2

[DUNI10] DROST B., ULRICH M., NAVAB N., ILIC S.: Model globally, match locally: Efficient and robust 3D object recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (June 2010), pp. 998–1005. doi:10.1109/CVPR.2010.5540108. 2

[EAF17] ELBAZ G., AVRAHAM T., FISCHER A.: 3D point cloud registration for localization using a deep neural network auto-encoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), pp. 2472–2481. doi:10.1109/CVPR.2017.265. 2

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (June 1981), 381–395. doi:10.1145/358669.358692. 2

[HSXS13] HAN J., SHAO L., XU D., SHOTTON J.: Enhanced computer vision with Microsoft Kinect sensor: A review. *IEEE Transactions on Cybernetics 43*, 5 (Oct 2013), 1318–1334. doi:10.1109/TCYB.2013.2265378. 2

[HWMD14] HANDA A., WHELAN T., MCDONALD J., DAVISON A.: A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA* (Hong Kong, China, May 2014). 3

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision*

and Pattern Recognition (CVPR) (June 2016), pp. 770–778. doi:10.1109/CVPR.2016.90. 2, 4

[INK*11] IZADI S., NEWCOMBE R. A., KIM D., HILLIGES O., MOLYNEAUX D., HODGES S., KOHLI P., SHOTTON J., DAVISON A. J., FITZGIBBON A.: KinectFusion: Real-time dynamic 3D surface reconstruction and interaction. In *ACM SIGGRAPH 2011 Talks* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 23:1–23:1. doi:10.1145/2037826.2037857. 1, 2

[JH99] JOHNSON A. E., HEBERT M.: Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. Pattern Anal. Mach. Intell. 21*, 5 (May 1999), 433–449. doi:10.1109/34.765655. 2

[KB14] KINGMA D., BA J.: Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014). 7

[KC17] KENDALL A., CIPOLLA R.: Geometric loss functions for camera pose regression with deep learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), pp. 6555–6564. doi:10.1109/CVPR.2017.694. 5

[LSP05] LAZEBNIK S., SCHMID C., PONCE J.: A sparse texture representation using local affine region. *IEEE transactions on pattern analysis and machine intelligence 27* (09 2005), 1265–78. doi:10.1109/TPAMI.2005.151. 2

[MSKS03] MA Y., SOATTO S., KOSECKA J., SASTRY S. S.: *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003. 6

[PDS16] PETRELLI A., DI STEFANO L.: Pairwise registration by local orientation cues. *Computer Graphics Forum 35*, 6 (2016), 59–72. doi:10.1111/cgf.12732. 1, 2

[PMR*19] PAIS G. D., MIRALDO P., RAMALINGAM S., NASCIMENTO J. C., GOVINDU V. M., CHELLAPPA R.: 3DRegNet: A deep neural network for 3D point registration. *arXiv:1904.01701* (2019). 1, 2, 3, 4, 5, 6, 7, 8, 9

[RBB09] RUSU R. B., BLODOW N., BEETZ M.: Fast point feature histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation* (May 2009), pp. 3212–3217. doi:10.1109/ROBOT.2009.5152473. 2

[RBMB08] RUSU R. B., BLODOW N., MARTON Z. C., BEETZ M.: Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sep. 2008), pp. 3384–3391. doi:10.1109/IROS.2008.4650967. 2

[RL01] RUSINKIEWICZ S., LEVOY M.: Efficient variants of the ICP algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling* (May 2001), pp. 145–152. doi:10.1109/IM.2001.924423. 2

[RMBB08] RUSU R., MARTON Z., BLODOW N., BEETZ M.: Persistent point feature histograms for 3D point clouds. *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany 16* (01 2008). doi:10.3233/978-1-58603-887-8-119. 2

[XOT13] XIAO J., OWENS A., TORRALBA A.: SUN3D: A database of big spaces reconstructed using SfM and object labels. In *2013 IEEE International Conference on Computer Vision* (Dec 2013), pp. 1625–1632. doi:10.1109/ICCV.2013.458. 3

[YTO*18] YI K. M., TRULLS E., ONO Y., LEPETIT V., SALZMANN M., FUA P.: Learning to find good correspondences. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (June 2018), pp. 2666–2674. doi:10.1109/CVPR.2018.00282. 4

[Zha12] ZHANG Z.: Microsoft Kinect sensor and its effect. *IEEE MultiMedia 19*, 2 (Feb 2012), 4–10. doi:10.1109/MMUL.2012.24. 2

[ZSN*17] ZENG A., SONG S., NIESSNER M., FISHER M., XIAO J., FUNKHOUSER T.: 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), pp. 199–208. doi:10.1109/CVPR.2017.29. 1, 2