Correspondence matching in unorganized 3D point clouds using Convolutional Neural Networks

Pujol-Miró, Alba[a], Casas, Josep R.[a], Ruiz-Hidalgo, Javier[a]

[a] *Universitat Politècnica de Catalunya, Jordi Girona 1-3, Barcelona 08034, Spain*

Abstract

This document presents a novel method based in Convolutional Neural Networks (CNN) to obtain correspondence matchings between sets of keypoints of several unorganized 3D point cloud captures, independently of the sensor used. The proposed technique extends a state-of-the-art method for correspondence matching in standard 2D images to sets of unorganized 3D point clouds. The strategy consists in projecting the 3D neighborhood of the keypoint onto an RGBD patch, and the classification of patch pairs using CNNs. The objective evaluation of the proposed 3D point matching based in CNNs outperforms existing 3D feature descriptors, especially when intensity or color data is available.

Keywords: matching; point cloud; convolutional neural networks

*Corresponding author: Tel.: +34-934-011-627
Email address: `alba.pujol@upc.edu` (Pujol-Miró, Alba)

1. Introduction

Correspondence matching allows to establish which parts of one image correspond to which parts of another image. This information is widely used to perform data fusion from different sensors in order to obtain an enriched representation of scene observations. Use cases are, for instance, the matching of stereo cameras to obtain depth information, the combination of data from several camera types (visible light, infrared, x-ray...) into a single source, or the computation of optical flow (local displacement between frames) in a video sequence.

The present work focuses on feature matching algorithms in a different domain: unorganized 3D point clouds, popular nowadays with several types of 3D sensors. Range sensors like Kinect or Asus Xtion use an infrared sensor to acquire, for each pixel, the distance from the object to the camera. This data can be combined with the RGB information to obtain an RGBD image of the scene. In these sensors, the 3D points are organized into a lattice. Other sensors, like LiDAR scanners, provide a spherical image with the radial distance to the capture point. There are also line scanners, which scan the scene line-by-line and provide streams of depth information.

When data from several sensor types is combined, the information can no longer be represented in a single regular lattice. The data from the multiple sensors is rather combined in a non-ordered (or unorganized) 3D point cloud with intensity or color information for each point.

Our work aims to obtain a reliable method to match several keypoints between two unorganized 3D point clouds exploiting both photometry and geometry. Current techniques consist in computing a feature vector using local and global information on the neighborhood of the query and target points. Comparing these two vectors, the keypoints can be classified as matches or non-matches. Several methods are available to obtain these feature vectors for 2D and 3D scenarios, but recent contributions in the case of 2D images [19] propose to classify image patches centered on the keypoints by means of Convolutional Neural Networks (CNNs) instead of using a descriptor.

The work presented in this document aims to extend deep learning techniques to match relevant keypoints in the 3D domain. To do so, a two-stage algorithm is proposed. As shown in Figure 1, first the 3D neighborhood around a potential keypoint is projected into the 2D domain, and then the image patches obtained are classified using CNN's.

In the next section, we introduce a brief review of state-of-the-art for 3D correspondence matching. Then, we describe the proposed procedure in

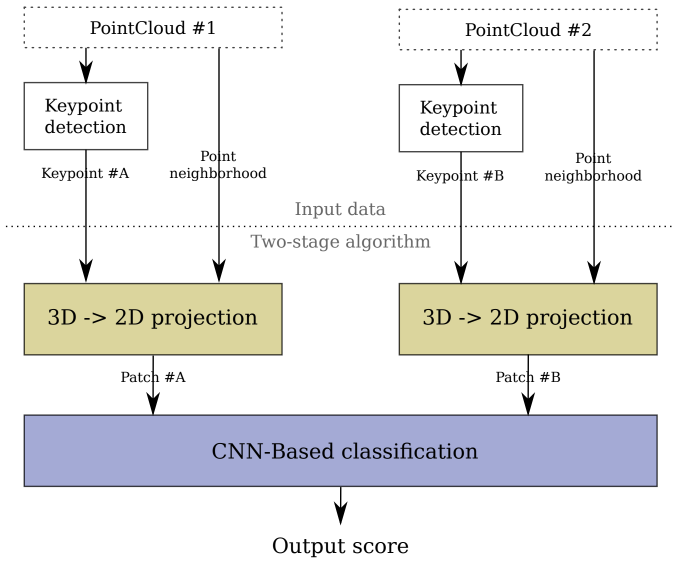


Figure 1: The proposed pipeline is divided in two parts: Projection of the neighborhood of a query point into a 2D patch, and classification of these patches related to keypoints as matches/non-matches using CNNs

section 3, and evaluate its performance against existing methods in section 4. Finally, our conclusions are drawn in section 5.

2. State of the art

Correspondence matching is a widely developed field in computer vision. The goal of this procedure can be stated as follows: given two points on two different captures of the same scene, establish if they correspond to the same point in the scene. Classical matching methods establish correspondences by extracting characteristics of the local neighborhood of the keypoints and comparing them. Thus, on most applications, the decision about which points will be query points is also an important factor to take into account. Although the procedure is out of the scope of this work, the selection of which points will be queried may have an effect on the performance of the algorithm.

The most relevant characteristics in a keypoint detector are the specificity, defined as the capability to differentiate the point from other keypoints on different sources, and the repeatability, defined as the ability to accurately detect the keypoint on all the sources. The specificity depends on which features are observed on the correspondence matching steps, but general characteristics requirements should be fulfilled in most cases. For instance, areas with high textures or sharp contours are favored over flat surfaces, where the location of keypoints cannot be established accurately.

We will use Intrinsic Shape Signatures (ISS) [20] as keypoint detector. The review in [16] shows that ISS provide the highest repeatability on most datasets. In addition, the detected points have good performance when matched using state-of-the-art 3D descriptors, as shown in [3].

Once the keypoints are established, classical correspondence matching methods consist in computing a feature vector on each keypoint and performing a distance-based matching [3]. One of the best performing algorithms is Rotational Projection Statistics (RoPS) [4]. RoPS projects the point neighborhood into various rotated patches in several planes, and extracts statistics about the projection. Another algorithm with good performance is Point Fea- ture Histogram (PFH) [10]. PFH computes a histogram describing the local geometry around a query point.

For a comprehensive evaluation of the proposed algorithm, other descriptors are used in the benchmarking we present. Spin Images [5] compute a histogram representing the point distribution in the neighborhood of a center point. Intensity Spin Images [6] use the same idea but representing the distribution of intensity in the surroundings. Rotation-Invariant Feature Transform (RIFT) [6] takes some concepts from the 2D descriptor SIFT. This descriptor constructs histograms from neighborhoods of increasing circular radius. Signature of Histograms of Orientations (SHOT) [14] encodes information about the topology within a spherical neighborhood. SHOT-Color [15] is an improved version that also uses color information to increase the performance.

Recent developments exploit Convolutional Neural Networks (CNN's) to perform a matching between two keypoints. Zagoruyko [19] presents a method to establish correspondences by querying if two keypoints match based on their surrounding pixels on the image. For each query keypoint, a 64x64 image patch is extracted. Then, these patches are classified using a CNN that provides a score indicating how similar the patches are.

However, unorganized point clouds can not be directly analyzed using a CNN, since no lattice structure is available. Recent developments presented in [8, 9] propose a method to segment unorganized point clouds using Neural Networks. The method consists in applying a transformation that makes the input invariant to the point order.

In our case, the proposed strategy consists in obtaining a local lattice structure around the query keypoints, analogously to the image patches in [19], using a technique similar to the RoPS computation [4]. The local lattice structure allows obtaining an RGBD representation of the surrounding patches that can be classified using existing 2D CNNs.

3. Methodology

In this section a method to establish correspondence between two sets of 3D data is explained. Given one 3D point in one capture and another 3D point in another capture this method is focused on establishing if these two points are from the same point in the 3D scene. The procedure used to obtain this correspondence is to extract information about the surrounding of each point and then use Convolutional Neural Networks to classify the points in matches or non-matches.

Figure 1 shows the two main steps of the proposed procedure. In the first step, for each query keypoint, an RGBD image patch representing its surroundings is extracted. The second step consists in the classification of the obtained RGBD patches using a convolutional neural network structure. The networks used are based on the ones presented in [19].

3.1. Transform point clouds to image patches

The first step of this procedure consists on mapping the surroundings of a keypoint into an RGBD image. A 3D point cloud of a scene can be seen as a sampling of the 3D surfaces that are contained in the scene. If the sampling is done from a single viewpoint (for example, a single static Kinect or LiDAR sensor), then the scene can be projected into a 3D domain without losing any information. However, if the available point cloud is a combination of captures from several locations, when projected onto a single plane points from several surfaces will be overlapped.

When matching keypoints, however, the interest area is reduced to a small neighborhood of the query point. In this case, only the points –and thus, the surfaces– close to the query point are relevant. In this close neighborhood the likelihood of not finding an arbitrary projection plane which does not have overlapped points from several surfaces is minimized.

Additionally, another factor has to be taken into account. In a point cloud, the density of the points may change on the scene. For example, on a Kinect camera the point density decreases with the distance of the sensor to the camera. The proposed projection technique should obtain a similar projection independently of the point density on the query point neighborhood.

Taking into account these two factors the following projection strategy is presented. First of all, a small neighborhood X_k is acquired for each keypoint k using a KD-tree with a fixed radius r as shown in Equation 1, where X_k are the point coordinates and I_k their corresponding intensity values.

$$X_k = \{\vec{x_{k1}}, \cdots, \vec{x_{kN}}\}, I_k = \{i_{k1}, \cdots, i_{kN}\} \tag{1}$$

The obtained neighborhood is centered by computing the centroid $\overline{\vec{x_k}}$ and subtracting it from the query points X_k, obtaining $\underline{X_k}$ as shown in Equation 2.

$$\overline{\vec{x_k}} = \frac{1}{N}\sum_{i=1}^{N} \vec{x_{ki}}$$
$$\underline{X_k} = X_k - \overline{\vec{x_k}} = \left\{\underline{\vec{x_{k1}}}, \cdots, \underline{\vec{x_{kN}}}\right\} \tag{2}$$

Using PCA, the principal components of the neighborhood $P_k = [\vec{p_{kx}}, \vec{p_{ky}}, \vec{p_{kz}}]$, ordered by decreasing eigenvalue, are obtained. These vectors define the optimal projection plane to minimize the overlapping between points, being the plane with normal vector $\vec{p_{kz}}$.

This projection provides a transformation that is invariant to the camera viewpoint and to the point density of the area. In most cases the PCA projection gives a strong plane vector, but the rotation within the plane is not well-defined. To obtain a better performance the orientation of the patches is normalized to a common direction.

To define a common orientation between all the patches the intensity values of the points contained in the neighborhood are used. The direction of maximum intensity $\vec{\eta_k}$ is computed using the location information $\underline{X_k}$ and the intensity information I_k, as shown in Equation 3.

$$\vec{\eta_k} = \sum_{i=1}^{N} \underline{\vec{x_{ki}}} \cdot i_{ki} \tag{3}$$

This information allows the definition of a new base $P'_k = [\vec{p'_{kx}}, \vec{p'_{ky}}, \vec{p'_{kz}}]$. This base has the same third vector as the original base, but the vectors in the plane are defined based on the direction of maximum intensity $\vec{\eta_k}$, as shown in Equation 4. The vector $\vec{p'_{kx}}$ is computed as the projection of $\vec{\eta_k}$ in the plane defined by $\vec{p_{kz}}$, and $\vec{p'_{ky}}$ as the vector orthogonal to $\vec{p'_{kx}}$ and $\vec{p_{kz}}$

$$\begin{aligned} \vec{p'_{kx}} &= \vec{\eta_k} - \langle \vec{\eta_k}, \vec{p_{kz}} \rangle \cdot \vec{p_{kz}} \\ \vec{p'_{ky}} &= \vec{p'_{kz}} \times \vec{p'_{kx}} \\ \vec{p'_{kz}} &= \vec{p_{kz}} \end{aligned} \tag{4}$$

With this new base, the points $\underline{X_k}$ are transformed into the base P'_k, obtaining $\underline{X'_k}$. This projection provides a representation invariant to the camera location and with a common orientation on all the query neighborhoods.

The next step is to obtain an RGBD image from these oriented points. This image is obtained by projecting the transformed point set into a regular $N \times N$ lattice. This lattice is defined within the range of the projected points, from $min\{\underline{X'_k}\}$ to $max\{\underline{X'_k}\}$ for the x and y axis.

To obtain the RGB information Each 3D point is projected into a cell, assigning their RGB value to the cell. If more than one point is assigned to the same cell, their values are averaged.

For the projection of the depth channel, we follow a similar procedure. In this case, the value d_{ik} assigned to the depth channel projection for each point in $\underline{X_k}$ is computed as the distance between the point and the projection plane (defined by the principal vector $\vec{p'_{kz}}$): $d_{ik} = \langle \underline{\vec{x_{ik}}}, \vec{p'_{kz}} \rangle$. If more than one point is assigned to the same cluster on the lattice, the final value is the average of the distance of the assigned points.

To summarize, the acquisition of an RGBD patch for each query point of an unorganized 3-D point cloud can be done following these steps, shown in Figure 2.

1. For each given query point, obtain a subset of the point cloud given by the points which are within distance r of the query point (Figure 2.a)
2. Compute the PCA decomposition P_k of the point subset (Figure 2.b).
3. Compute the direction of maximum intensity $\vec{\eta_k}$ to make the transformation invariant to rotation (Figure 2.c).
4. Rotate the base P_k to have the projection of $\vec{\eta_k}$ as the main plane vector (Figure 2.d).
5. Define a lattice in the projection plane of size $N \times N$. This lattice should be scaled to the range of the projected points (Figure 2.e).
6. Project the points into the lattice. For each cluster, if more than one point is assigned their RGB values are averaged. The depth value is defined as the distance of each point to the projection plane. In a same manner, if more than one point is assigned their depth values are averaged (Figure 2.f).
7. The final result is an RGBD image which is independent to the point density of the underlying surface and has a defined orientation (Figure 2.g).

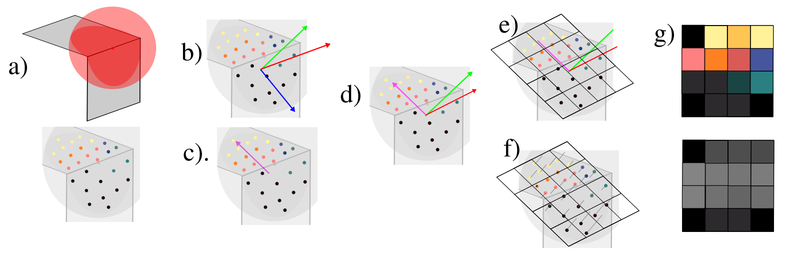


Figure 2: Schematic of the computation of an RGBD image from a 3D point cloud keypoint: a) Underlying structure and sampled points, b) PCA decomposition, c) Direction of maximum intensity, d) Rotation of the PCA vectors to match the direction of maximum intensity, e) Projection lattice (4x4), f) Projection of the points to the lattice, g) Obtained RGB (top) and depth (bottom) images

3.2. Classification using CNN

Once the RGBD patches are obtained, we classify the patches in matches and non-matches in a second step. We consider the CNN's proposed in [19] for scoring the similarity between patches.

The data used in [19] consists in a ground truth of true and false matches of 64×64 patches with the same content orientation. To construct this dataset, keypoints have been detected on several images and a distance threshold established to determine if they are matching. A balanced dataset (same number of matches and non-matches) is constructed for the training process. The training data consist on 64×64 image patches which have been rotated to have the same orientation between matches.

The data obtained in the previous step is very similar: image patches of the keypoint surroundings invariant to rotation. However, there are three key differences: the represented data, the size of the image and the shape of the patch.

In our case, the obtained data is not only the RGB information, but also the depth information. Therefore, to take advantage of this new information the network structures will be modified to allow the inclusion of this data type.

Another key difference is the image size. The images used in [19] are 64×64. The images obtained by a traditional camera have much higher pixel counts that the number of valid points that can be obtained using state-of-the-art range sensors. Therefore, if the lattice is sized to 64×64, with small neighborhoods there will be not enough points in the neighborhood to fill the lattice and there will be random blank spaces in the interior, which affects negatively the performance. One possible solution is to increase the neighborhood radius. However, this solution has the negative effect

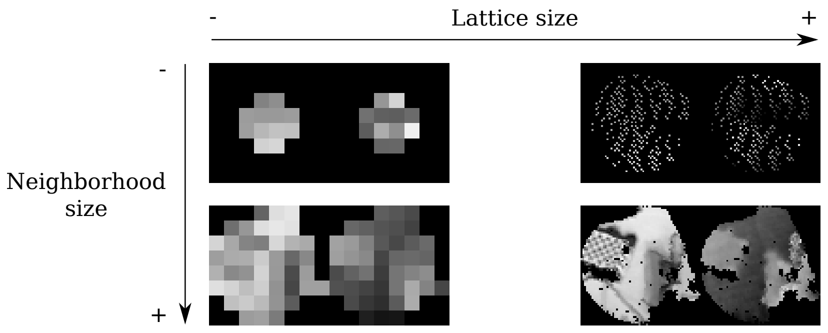


Figure 3: Illustrative representation of the effects of patch size and neighborhood query radius. In each case, the left image corresponds to the intensity information, whereas the right image corresponds to the depth information.

of including more surroundings that might not be relevant to the scene. Another possible solution, selected in this work, is to use smaller lattice sizes and then up-sample the data before feeding it to the network.

The third relevant difference is the information contained in the patch. Since the neighborhood is acquired thresholding with an euclidean distance from the query point, the resulting image contains a circle-shaped patch with black corners.

To better illustrate the effects explained above, Figure 3 provides four illustrative examples of the obtained patch for various neighborhood and lattice sizes. As it can be seen, the three differences mentioned above are present. First of all, each patch has associated a depth information (right) for each RGB information (left). Another relevant factor is the image size. As it can be seen, increasing the lattice size without gathering more points (top right) produces undesirable artifacts on the image, produced by blank clusters inside the main area. It can also be observed that lowering the patch size produces a blurred image due to several points being averaged in the same cluster (bottom left). Finally, it can also be observed that the patch has a roundish shape, produced by the query of the points using a ball distance.

Having into account this information, the following structures are proposed. It should be noted that on all the network structures instead of using the RGB channels a single intensity channel is used, leaving us with two possible channels, intensity and depth. Additional information for the presented networks can be seen in Figure 4.

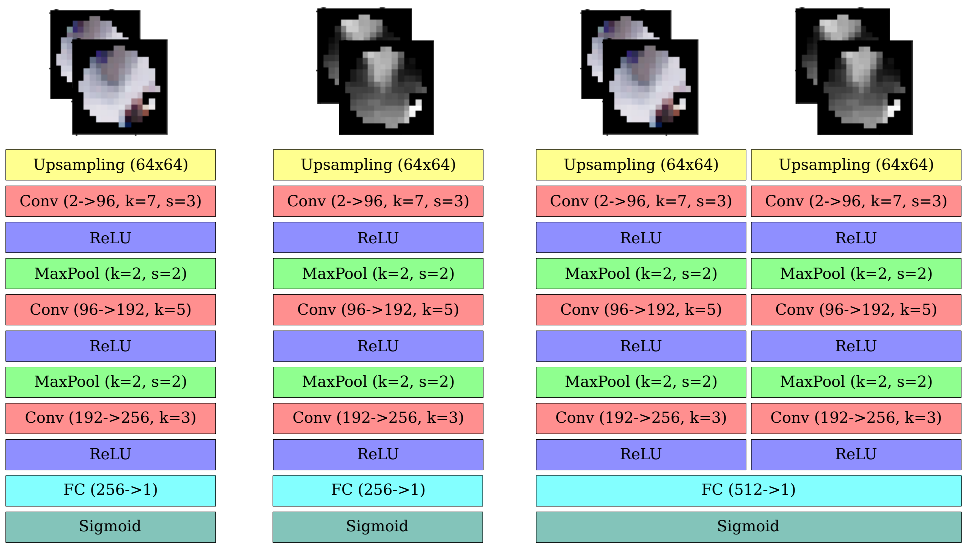
2ch-intensity Network structure using only intensity information. The two $N \times N$ single-channel images (intensity only) from each keypoint are concatenated into the third dimension producing a two-channel image $N \times N \times 2$. This two-channel patch scaled to $64 \times 64 \times 2$. A network with 3 convolutional layers is used. This network finishes with a fully connected layer and sigmoid layer that normalizes the output of the fully connected layer between 0 –non-match– and 1 –full match.

2ch-depth This network uses the same structure as before, but in this case the single-channel image used per keypoint contains the depth information instead of the intensity information. The structure also scales the input image to $64 \times 64 \times 2$, has 3 convolutional layers and also outputs a confidence value.

2ch-2stream This network is a combination of the two networks presented before. It consists on a two-stream network, one for the intensity and one for the depth information. The streams have the same structure presented before and are joined on the final fully connected and sigmoid layers. This network also outputs a single confidence value.

This network structures are derived from the network structures presented in [19], used with intensity patches. The structures that provide best results in their test case are the ones that input the patches as a two-channel image, closely followed by the central-surround approach. This approach consists on dividing the network in two streams. One stream is feed with the central part of the patches as a two-channel image, whereas the other stream is feed with a downsampled version of the same patches, also in a two-channel image. Both streams have the same structure but do not share any parameters. Having into account the round shapes of the patches used in this work, it has been considered that using the central-surround system will not provide a significant advantage given the large blank areas around the circle patch.

Therefore, our first test uses the two channel network with only the intensity information. The second straightforward test consists in using the same network structure, but only with the depth channel as input. Then, in order to take advantage of the intensity and depth information jointly, a new network structure is presented. As stated in [1], mixing intensity and depth channels in the same input gets worse results than providing the different channels as input on separate network streams. Therefore, the proposed structure has two streams with the same structure as the '2ch' network, and concatenates the features on the fully connected layer, without sharing parameters.



a) 2ch-intens   b) 2ch-depth   c) 2ch-2stream

Figure 4: Network structures used: a) U(64) - C(2->96,k=7,s=3) - ReLU - P(k=2,s=2) - C(96->192, k=5, s=1) - ReLU - P(k=2,s=2) - C(192->256, k=3, s=1) - ReLU - F(256->1) - Sigmoid, b) same structure as a) but input with depth images. c) Two branches U(64) - C(2->96,k=7,s=3) - ReLU - P(k=2,s=2) - C(96->192, k=5, s=1) - ReLU - P(k=2,s=2) - C(192->256, k=3, s=1) - ReLU, joined with F(512->1) - Sigmoid, Notation : U(N) is the initial up-sampling to size $N \times N$ C($i- > n$, k=k, s=s) is a convolutional layer with i input channels, n filters of spatial size $k \times k$ applied with stride s, P(k=k, s=s) is a max-pooling layer of size $k \times k$ applied with stride s, and F($i- > n$) denotes a fully connected linear layer with i input units and n output units.

4. Experiments

In this chapter a comprehensive evaluation of the proposed method is presented. First of all, the datasets and measures used to evaluate this work are presented. Several datasets with different characteristics are used to obtain a wide benchmark of the performance of the method an the effect of the training dataset on the final performance. This method is also benchmarked against several state-of-the-art hand crafted descriptors.

In addition to the final performance, a study on the method parameters is presented. The effects of the radius size, the patch size and the network structure will also be studied on the training and validation tests.

The code used to perform the development and the results obtained is available from our webpage[1].

4.1. Datasets

The procedure proposed in the previous section aims to, given two keypoints from different captures and their surroundings, establish if these two keypoints are from the location in the scene. Therefore, the training and testing dataset should contain information about matching keypoints and their surroundings.

For 2D images, there are datasets available, such as [18], that providing a set of patches together with the information about which patches correspond to the same scene point.

For 3D data, no such dataset is readily available. Therefore, we have selected several 3D datasets with multiple captures of the same scene (either multi-viewpoint or scenes with camera motion), which have the camera location available for each capture. For the ground-truth labeling, we detect several keypoints on different captures, and then label the matching points in the other captures with the help of the known camera location parameters.

A factor that will be relevant during the development of these experiments is the point cloud resolution. The point cloud resolution is defined as the average distance between neighboring points in a point cloud. To compute this value on a capture, each point is matched with its nearest point and the distance is computed. The average of this distance between each point of the capture defines the point cloud resolution.

[1]https://imatge.upc.edu/web/resources/correspondence-matching-unorganized-3d-point-clouds-using-convolutional-neural-networks

This parameter is useful to be able to not be dependent on the size of the distance of the capture and the resolution of the sensor. In this work, for each sequence a global resolution is captured, averaging the point cloud resolution of each frame into a single value, called the dataset resolution.

To obtain the location-labeled captures some datasets reviewed in [2] have been used. Two sequences –`freiburg1/desk` and `freiburg1/teddy`– from the Freiburg RGBD dataset [13] and two sequences from the CoRBS Dataset [17] –`corbs/desk` and `corbs/e.cabinet`– have been used. Additionally, a depth-only sequence from the Microsoft 7 scenes dataset [12] –`m7scenes/fire`– has been used.

For train and validation only the sequence `freiburg1/desk` has been used. This sequence contains a capture of a handheld Kinect 1 camera moving around a common office desktop desk.

For test and validation four different sequences have been used. Sequence `freiburg1/teddy` contains a capture of a teddy bear recorded in a similar fashion than `freiburg1/desk`. `corbs/desk` also contains a capture of an office desk, but in this case is recorded using a Kinect 2 camera. Sequence `corbs/e.cabinet` is the most different capture from `freiburg1/desk`: contains a different scene –an electrical cabinet– and is captured using a different sensor – Kinect 2 rather tan Kinect 1. Finally, the sequence `m7scenes/fire` contains a capture of a fire extinguisher. This sequence do not have color information, sincethe registration between the depth and the intensity data is not available in the Microsoft 7 scenes dataset.

The `freiburg1/desk` dataset is divided into training and validation split. The first 416 images are used for training, and the remaining 179 are used for validation. This implies that the matches between images on the train split and images on the validation split are discarded to avoid having repeated patches on the train and validation split.

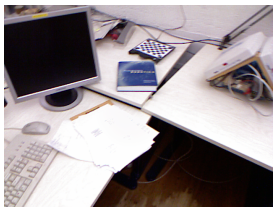
Additional information about the datasets used can be found in Table 1. Additionally, a visual representation of the scene shown in each dataset can be seen in Figure 5.

4.2. Measures

In this project a balanced binary classification project is evaluated. The output of this method is a confidence value between 0 –non-match– and 1 –full match. The thresholding of this confidence value to obtain two sets of matches and non-matches is left outside the scope of this document, since the thresholding value is application-dependent: if the application can handle a high number of false positives the threshold will be lower.

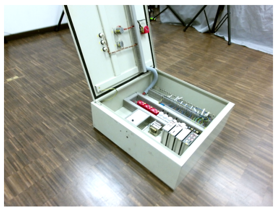
Dataset	Sensor	Frames	Resolution	Kpts/frame	T.matches	Color	Train	Test
`freiburg1/desk`	Kinect 1	595	2.29e-3 m	279	327039	x	x	
`freiburg1/teddy`	Kinect 1	301	2.29e-3 m	1102	438860	x		x
`corbs/desk`	Kinect 2	100	6.58e-3 m	1206	98130	x		x
`corbs/e.cabinet`	Kinect 2	100	5.47e-3 m	901	52007	x		x
`m7scenes/fire`	Kinect 1	400	2.65e-3 m	390	450072			x

Table 1: Datasets used for evaluation: frames used, average keypoints per frame found in each dataset and total number of true matches are stated. For evaluation purposes, an equal number of false matches are randomly selected from the non-matching pairs.


(a) freiburg1/desk

(b) freiburg1/teddy

(c) corbs/desk


(d) corbs/e.cabinet

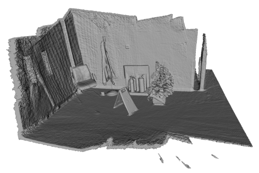

(e) m7scenes/fire

Figure 5: Visual examples of the scene shown in each dataset

`gamma_21`	`gamma_32`	`min_neigh`	`salient_rad`	`non_max_rad`	`normal_rad`	`border_rad`
0.975	0.975	5	0.15	0.05	0.0.01	0.0025

Table 2: Parameters used for the ISS detector. These parameters have been tuned from the default parameters assuming a cloud resolution of 2.5e-3.

Reference [3] evaluates 3D local feature descriptors with Precision-Recall curve and the AUC measure. AUC is stated to have better performance when the positive and negative datasets are balanced. Therefore, the measures selected for the evaluation are the ROC curve for the graphical results and the AUC for the numerical results.

To perform this evaluation, for each frame in each dataset a set of keypoints is detected. As stated in Section 2, for the detection of keypoints we use Intrinsic Shape Signatures (ISS) [20]. The high repeatability shown in [16] allows to detect a substantial amount of matched keypoints and have a large dataset of matched keypoints using only a few frames. The same ISS detector parameters, shown in Table 2, are used on all the datasets to perform the detection. Since in this context the performance of the keypoint detector is not evaluated, the parameters have not been fine-tuned.

To obtain the ground truth of matching and non-matching pairs, for each sequence all the possible combinations between keypoints of different frames are taken into account. Since the keypoint location is a non-discrete value, a certain distance threshold must be allowed between keypoint locations to consider them a match or a non-match. In this case, the selected threshold is $2 * dataset\_resolution$, which allows to take into account the errors obtained in the capture and the point density.

To obtain a balanced dataset, for each pair of matched keypoints, a random pair of non-matching keypoints is selected on the same capture pair.

4.3. Descriptor benchmark

The performance of the proposed patch matching algorithm is evaluated against some of the most relevant state-of-the-art algorithms according to [3]. The descriptors tested are Spin Images [5], Intensity Spin Images [6], RIFT [6], SHOT [14], SHOTColor [15], Fast Point Feature Histogram [10] and RoPS [4], as stated in Section 2. The selected implementation for the stated descriptors is available from the Point Cloud Library (PCL)[11].

4.4. Image patches

The first part of the proposed pipeline computes image patches from 3D keypoint neighborhoods. As stated in the previous section, creating the

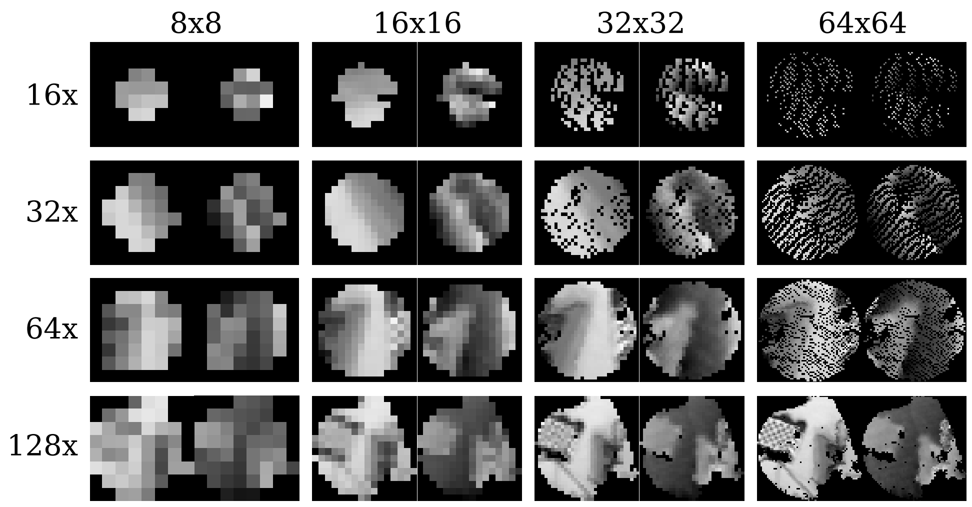


Figure 6: Representation of patches of several neighborhood sizes (rows) and lattice sizes (columns). In each case, the left image corresponds to the intensity information, whereas the right image corresponds to the depth information.

patch consists in obtaining a set of neighboring points using a KD-Tree, and then projecting these points into a 2D lattice.

In this procedure two main parameters can be tuned: the neighborhood size and the lattice size. The size of the K-neighborhood has been set according to the resolution parameter in Table 1. Several

Figure 6 shows a visual example of the constructed patches from the `freiburg1/desk` dataset using a 8×8, 16×16, 32×32 and 64×64 patch. For each patch size, several neighborhood sizes are taken into account: $16\times$,$32\times$, $64\times$ and $128\times$. This neighborhood sizes are the maximum distance in the terms of the dataset resolution ($16\times$ the dataset resolution, etc.).

When input to the network, these patches are upsampled to 64×64, the same patch size used in [19]. Figure 6 also shows that the rotation applied to the points before the projection in order to normalize in the direction of maximum intensity change (Equations 3– 4 in Section 3) effectively makes the patches invariant to rotation.

Whilst smaller lattices with large neighbor radius produce a blurry image of the surroundings, too large lattices with small neighbor radius provide patches with holes that will hinder the evaluation. A trade-off between lattice size and neighborhood size can be observed. For each neighborhood radius, there is a patch size that provides the sharpest image without having blank spaces inside the patch. For example, for a neighborhood radius 32× the dataset resolution the clearest patch size is 16×16, whereas for a neighborhood radius 64× the dataset resolution the clearest patch size is 32×32.

Each network structure will be trained with the presented combinations of patch size and neighborhood radius.

4.5. Training parameters

The parameters used to train the network are the same for all network configurations. In all cases, the dataset used in the training step has been `freiburg1/desk`, with 70% of the matches (228.927 true pairs and 228.927 false pairs) in the training split and 30% of the matches (98.112 true pairs and 98.112 false pairs) in the validation split. The network has been trained for 90 epochs, with a learning rate of 0.1, a momentum of 0.9, a weight decay of 0.0004 and a batch size of 256 pairs. The validation split is used to stop the training and select the epoch with best *AUC* measure. The software framework used is PyTorch [7].

4.6. Performance

The system has been trained using three different network structures: `2ch-intens`, `2ch-depth` and `2stream`, as explained in the previous section. Each network structure has been trained with 4 different patch sizes: 8×8, 16×16, 32×32 and 64×64; and 4 different patch resolutions: 16×,32×, 64× and 128× the dataset resolution.

Table 3 shows the evaluation using the *AUC* of the different structures with the different patch sizes and resolutions on the validation split on the dataset `freiburg1/desk`.

In these tests, the best performing network is `2ch-2stream`, closely followed by `2ch-intens`. The `2ch-2stream` structure combines intensity and depth information, whereas `2ch-intensity` uses only the intensity information and `2ch-depth` uses only the depth information.

Comparing the performance of the `2ch-intensity` and `2ch-depth` networks it can be seen that the intensity information is more relevant for the patch matching than the depth information. However, when both data are combined in the `2ch-2stream` network the performance is slightly increased.

Observing all the network structures globally it can be seen a correlation between the patches seen in Figure 6 and the performance of the different configurations. It can be observed that the best performing combinations of patch size and neighborhood radius are the ones that provide a sharp image and include a relevant amount of information of the surroundings.

Radius \ Size	8	16	32	64
16×	0,956	0,961	0,956	0,931
32×	0,964	0,970	0,969	0,961
64×	0,972	**0,978**	0,977	0,977
128×	0,962	0,976	0,976	0,975

(a) 2ch-intensity

Radius \ Size	8	16	32	64
16×	0,861	0,899	0,906	0,906
32×	0,909	0,931	0,925	0,926
64×	0,934	**0,948**	0,946	0,946
128×	0,916	0,923	0,933	0,933

(b) 2ch-depth

Radius \ Size	8	16	32	64
16×	0,960	0,964	0,961	0,944
32×	0,971	0,973	0,972	0,965
64×	0,976	**0,981**	0,980	0,978
128×	0,965	0,976	0,975	0,976

(c) 2ch-2stream

Table 3: AUC for different patch sizes, k-neighbor radius and network configurations on `freiburg1/desk` dataset (validation split). The best value for each network structure is highlighted in bold face.

However, having a high neighborhood radius is not always beneficial: when the neighborhood radius is too large, it gathers surroundings of the keypoint that are not relevant to the local neighborhood. These points have high variability between captures and hinder the results of the algorithm.

A combination of patch size and radius has been selected for each network. On all network configurations, the best radius is $64\times$ the dataset resolution. This radius gives enough context of the surrounding of the patch without gathering non-relevant information of the neighborhood. The best lattice size on all the datasets is 16×16. This lattice size allows to represent the data on the surroundings without losing important information and without having blank spaces inside the patches.

Observing the Figure 6 it can be seen that with radius $64\times$ the dataset resolution the information contained in the patch have enough texture and distinctive features to be able to match different keypoints. With radius $32\times$, only a small surrounding is represented and it is not big enough to provide relevant information. With radius $128\times$, the neighborhood includes points that are too distant to the keypoint and do not feed relevant information of the immediate surroundings.

Regarding the patch size with radius $64\times$ we can see that both 16×16 and 32×32 provide a good representation of the surroundings. Although 32×32 provides a sharper representation some small holes are showing in the patch representation. This shows that the network is more sensitive to blank points inside the patch than to blurry representations of the surroundings. Therefore, the selection of the patch size has to be taken into account when selecting a lattice size.

From the previous results, the network parameters selected for benchmarking against the stated descriptors are radius $64\times$ and patch size 16×16, used on the network structures `2ch-intens`, `2ch-depth` and `2stream-8`. These networks are benchmarked using the `freiburg1/teddy`, `corbs/desk`, `corbs/e.cabinet` and `m7scenes/fire` datasets. The results obtained are included in Table 4 and Figure 7.

In Table 4 all the presented network structures outperform the existing 3D descriptors in the benchmark. The results are specially outstanding when color information is available (`freiburg1/teddy`, `corbs/desk`, `corbs/e.cabinet` datasets), but the proposed global methods also outperform the existing descriptors when only depth information is used (in the network `2ch-depth`).

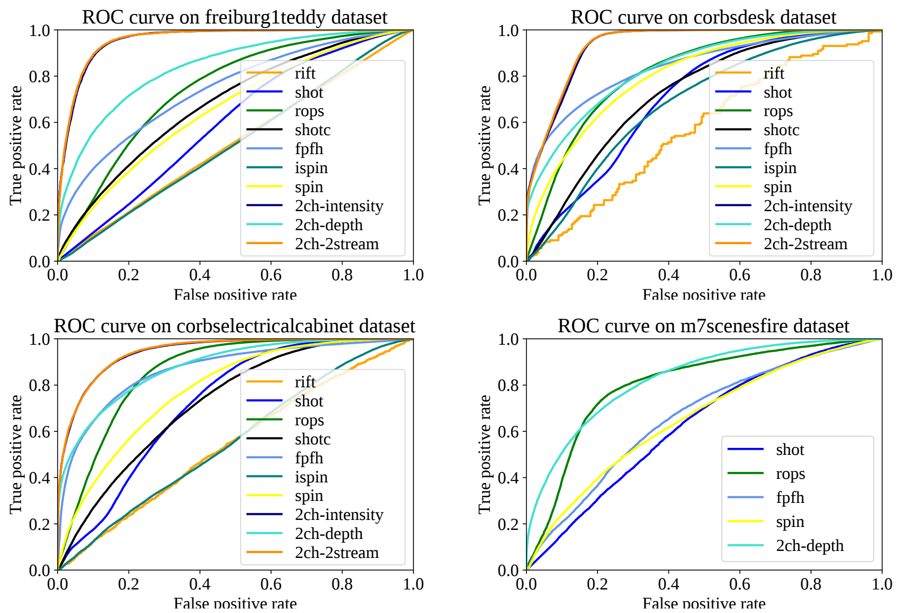
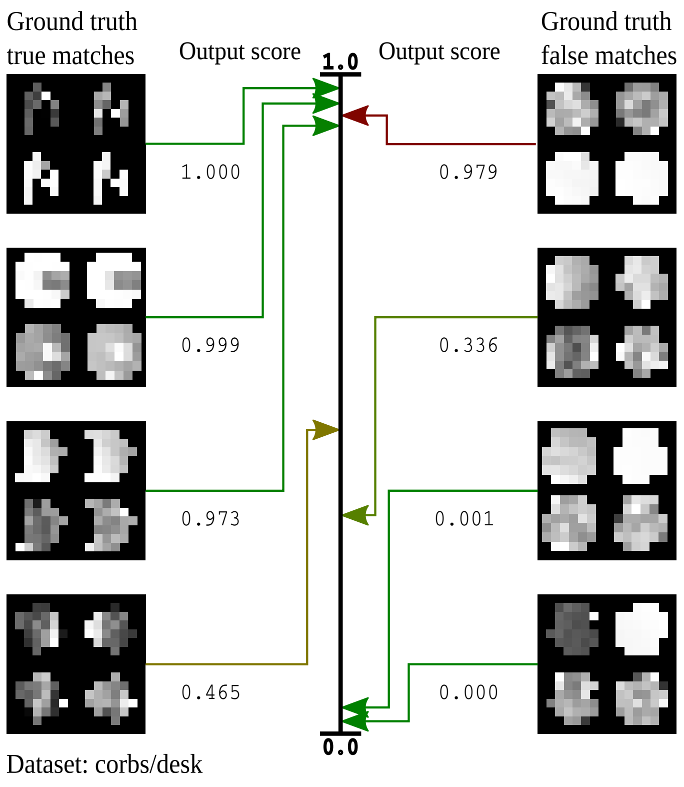


Figure 7: ROC curves for the benchmark and tested descriptors on the different datasets. Above (left to right): `freiburg1/teddy` and `corbs/desk`, below: `corbs/e.cabinet` and `m7scenes/fire`.

Dataset	RIFT	SHOT	SHOTC	RoPS	FPFH	SI	iSI	*2ch-i*	*2ch-d*	*2ch-2str*
`freiburg1/teddy`	0.51	0.60	0.68	0.74	0.74	0.66	0.51	**0.96**	0.84	**0.96**
`corbs/desk`	0.57	0.71	0.72	0.82	0.84	0.80	0.68	0.93	0.84	**0.94**
`corbs/e.cabinet`	0.54	0.73	0.73	0.86	0.87	0.78	0.56	0.94	0.88	**0.95**
`m7scenes/fire`	–	0.63	–	0.80	0.66	0.65	–	–	**0.83**	–

Table 4: AUC values for the 7 state of the art descriptors tested and the three network structures presented. For each dataset, best global method is marked in bold and best benchmarking descriptor underlined. For the sequence `m7scenes/fire` only those methods that do not need intensity information have been evaluated.



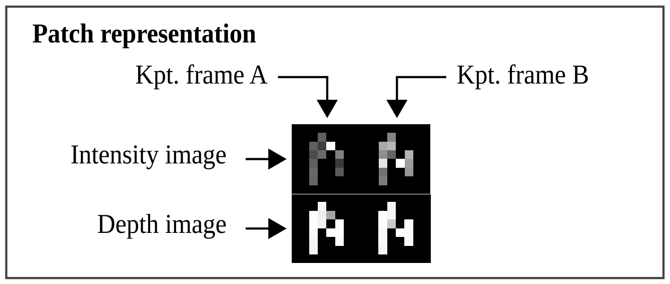


Figure 8: Visual results. The patches are separated in ground-truth matches (left) and ground truth non-matches (right). The matches are sorted in descending score order. Patch size 8×8 is used for clarity.

4.7. Visual results

In addition to the numerical results, some visual results of the output of the neural network are shown in Figure 8. The output obtained from the network is a value indicating the confidence that two patches are matching or non-matching. Score ranges for each dataset are provided.

We present the results in ascending score order (non-matching patches) and descending score order (matching patches) for each dataset. Pairs of patches are shown for intensity and/or depth, separated in matches (top) and non-matches (bottom) according to the ground-truth. Therefore, patch pairs easiest to classify by the network are shown on the left (true patches with high score and false patches with low score). Patch pairs more difficult to classify are located on the right (true patches with low score and false patches with high score) for each dataset group.

Please note that the network offers very good performance to detect similar or non-similar patches when enough structure is present, either using intensity or depth information.

In Figure 8 some results obtained with the `corbs/desk` dataset are shown. In this case, four matching points and four non-matching points are randomly selected. In the image, the patches corresponding to the true matches are shown in the left column and the images corresponding to the non-matches are shown in the right column. The numerical score obtained by the neural network for each pair shown. A high score indicates that the two patches are similar, whereas a low score indicates that the two patches are different. The patches are sorting in descending score order. Therefore, the correctly classified matching patches are shown on the top left, whereas the correctly classified non-matching patches are shown on the bottom right.

For each match or non-match four images are shown: the top row correspond to the intensity image and the bottom row correspond to the depth image. Column-wise, the images on the left column correspond to the information for one keypoint and the images on the right column correspond to the information of the other keypoint.

Observing the scores obtained on the true matches it can be seen that high scores are given when the patches present distinctive structures on the same spatial locations in the patch (third and fourth patch). However, patches with distinctive features but not correctly oriented (first patch) obtain a low score. This shows the importance of having a common orientation in the patch set, and how errors in this computation hinder the performance of the algorithm.

Regarding the false matches it can be seen that this method correctly gives a low score to patches which do not share any features and have different intensity and depth values (first patch). However, on flat patches with similar intensity textures (fourth patch) it gives an higher score.

On overall, it can be seen that this method offers a good performance classifying the patches when enough texture and contour is available on the images.

5. Conclusions

Correspondence matching between sets of keypoints is a highly relevant technique for 2D/3D multi-sensor fusion. We aim at contributing a reliable method to match keypoints for unorganized 3D point clouds, improving the available state-of-the-art 3D descriptors.

In particular, we propose a procedure to perform pair matching of keypoints between several instances of 3D point clouds. The proposed approach takes advantage of recent developments on the usage of CNNs for patch matching in 2D images. But unorganized 3D point clouds cannot be directly analyzed using CNNs, since no lattice structure is available. To overcome this limitation, we propose to obtain a local lattice by projecting the K-NN neighbors of 3D keypoints onto 2D image patches. We feed the network structure with these patches containing both intensity and geometry (depth) information of the keypoint neighborhood. The network classifies the 2D patches and provides a score indicating how similar the patches are, so that we can derive a matching/non-matching classification for each pair of the corresponding 3D keypoints.

The procedure has been evaluated in several datasets. Experimental results have shown an excellent performance on all tests, improving the results of existing 3D descriptors for keypoint matching. The proposed procedure exhibits a low dependence on the training dataset or the sensor type. Therefore, we believe the proposed methodology can be used to perform multi-modal/multi-sensor registration since no underlying lattice structure is needed on the 3D point clouds.

6. Acknowledgements

This research was supported by the Spanish Ministry of Science and Innovation via a doctoral grant to the first author (FPU2014), and developed in the framework of project TEC2016-75976-R, financed by the Ministerio de Economía, Industria y Competitividad and the European Regional Development Fund (ERDF).

References

[1] Audebert, N., Le Saux, B., Lefèvrey, S., 2017. Fusion of heterogeneous data in convolutional networks for urban semantic labeling, in: Urban Remote Sensing Event (JURSE), 2017 Joint, IEEE. pp. 1–4.

[2] Firman, M., 2016. Rgbd datasets: Past, present and future, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 19–31.

[3] Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., Kwok, N.M., 2016. A comprehensive performance evaluation of 3d local feature descriptors. International Journal of Computer Vision 116, 66–89.

[4] Guo, Y., Sohel, F., Bennamoun, M., Lu, M., Wan, J., 2013. Rotational projection statistics for 3d local surface description and object recognition. International journal of computer vision 105, 63–86.

[5] Johnson, A., Hebert, M., 1998. Surface matching for object recognition in complex 3-d scenes. to appear in. Image and Vision Computing .

[6] Lazebnik, S., Schmid, C., Ponce, J., 2005. A sparse texture representation using local affine regions. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1265–1278.

[7] PyTorch team, . PyTorch. `http://pytorch.org/`.

[8] Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2016. Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593 .

[9] Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 .

[10] Rusu, R.B., Blodow, N., Beetz, M., 2009. Fast point feature histograms (fpfh) for 3d registration, in: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE. pp. 3212–3217.

[11] Rusu, R.B., Cousins, S., 2011. 3D is here: Point cloud library (PCL), in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE. pp. 1–4.

[12] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A., 2013. Scene coordinate regression forests for camera relocalization in rgb-d images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2930–2937.

[13] Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D., 2012. A benchmark for the evaluation of rgb-d slam systems, in: Proc. of the International Conference on Intelligent Robot Systems (IROS).

[14] Tombari, F., Salti, S., Di Stefano, L., 2010. Unique signatures of histograms for local surface description, in: European Conference on Computer Vision, Springer. pp. 356–369.

[15] Tombari, F., Salti, S., Di Stefano, L., 2011. A combined texture-shape descriptor for enhanced 3d feature matching, in: Image Processing (ICIP), 2011 18th IEEE International Conference on, IEEE. pp. 809–812.

[16] Tombari, F., Salti, S., Di Stefano, L., 2013. Performance evaluation of 3d keypoint detectors. International Journal of Computer Vision 102, 198–220.

[17] Wasenmuller, O., Meyer, M., Stricker, D., 2016. CoRBS: Comprehensive rgb-d benchmark for slam using kinect v2, in: IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE. p. . URL: `http://corbs.dfki.uni-kl.de/`.

[18] Winder, S.A., Brown, M., 2007. Learning local image descriptors, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE. pp. 1–8.

[19] Zagoruyko, S., Komodakis, N., 2015. Learning to compare image patches via convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4353–4361.

[20] Zhong, Y., 2009. Intrinsic shape signatures: A shape descriptor for 3d object recognition, in: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, IEEE. pp. 689–696.