A Radial Search Method for Fast Nearest Neighbor Search on Range Images

Federico Tombari[1], Samuele Salti[1], Luca Puglia[2(✉)], Giancarlo Raiconi[3], and Luigi Di Stefano[1]

[1] Dipartimento di Informatica - Scienza e Ingegneria, University of Bologna, Bologna, Italy
{federico.tombari,samuele.salti,luigi.distefano}@unibo.it
[2] Dipartimento di Informatica, University of Salerno, Salerno, Italy
lpuglia@unisa.it
[3] D.I.E.M., University of Salerno, Salerno, Italy
gianni@unisa.it

Abstract. In this paper, we propose an efficient method for the problem of Nearest Neighbor Search (NNS) on 3D data provided in the form of range images. The proposed method exploits the organized structure of range images to speed up the neighborhood exploration by operating radially from the query point and terminating the search by evaluating adaptive stop conditions. Despite performing an approximate search, our method is able to yield results comparable to the exhaustive search in terms of accuracy of the retrieved neighbors. When tested against open source implementations of state-of-the-art NNS algorithms, radial search obtains better performance than the other algorithms in terms of speedup, while yielding the same level of accuracy. Additional experiments show how our algorithm improves the overall efficiency of a highly computational demanding application such as 3D keypoint detection and description.

1 Introduction and Related Work

In the past few years there has been a growing interest in processing 3D data for computer vision tasks such as 3D keypoint detection and description, surface matching and segmentation, 3D object recognition and categorization. The relevance of such applications has been fostered by the availability, in the consumer market, of new low-cost RGB-D cameras, which can simultaneously capture RGB and range images at a high frame rate. Such devices are either based on structured light (e.g. Microsoft Kinect, Asus Xtion) or Time-of-Flight (TOF) technology (e.g., Kinect II), and belong to the class of active acquisition methods. On the other hand, low-cost RGB-D sensors belonging to the class of passive acquisition technologies are mostly based on stereo cameras [13] or Structure-from-Motion.

Independently from the specific technology being used, each sensor acquires 3D data in the form of range images, a type of 3D representation that stores

G. Hua and H. Jégou (Eds.): ECCV 2016 Workshops, Part III, LNCS 9915, pp. 563–577, 2016.
DOI: 10.1007/978-3-319-49409-8_49

depth measurements obtained from a specific point in 3D space (i.e., sensor viewpoint) — for this reason, it is sometimes referred to as 2.5D data. Such representation is *organized*, in the sense that each depth value is logically stored in a 2D array, so that spatially correlated points can be accessed by looking at nearby positions on such grid. By estimating the intrinsic parameters of the sensors, usually available via calibration, the (x, y, z) coordinates associated to each depth value can be directly obtained. Conversely, point clouds are *unorganized* 3D representation that simply stores 3D coordinates in an unordered list.

Arguably the most ubiquitous task performed on 3D data for the aforementioned computer vision applications is represented by the Nearest Neighbors Search (NNS), i.e. given a query 3D point, find its k nearest neighbors (*kNN Search*), or, alternatively, all its neighbors falling within a sphere of radius r (*Radius Search*). This is for example necessary for computing standard surface differential operators such as normals and curvatures. In addition, NNS is a required step also for keypoint detection and description on 3D data, which are deployed, in turn, for 3D object recognition and segmentation. Another relevant example (among many others) of the use of NNS is the Iterative Closest Point (ICP) [5] algorithm, a key step for most 3D registration, 3D reconstruction and SLAM applications.

When NNS has to be solved on a point cloud, being it an unorganized type of 3D data representation, efficient indexing scheme are typically employed to speed up the otherwise mandatory linear search. Nevertheless, despite such schemes are particularly efficient, the NNS on point clouds can still be extremely time consuming, since the complexity grows with the size of the point cloud. In particular, over the years several methods have been proposed to solve optimally the NNS problem in the fastest way possible based on heuristic strategy [6], clustering techniques (e.g. hierarchical k-means, [8]) or hashing techniques [2]. Currently, the most popular approach is the kd-tree approach [7], or its 3D-specific counterpart known as octree.

In addition to exact algorithms, also approximated methods have been proposed, which trade-off a non optimal search accuracy with a higher speed-up with respect to the linear search. In [4], a modified kd-tree approach known as Best Bin First (BBF) is presented, where a priority queue with a maximum size is deployed to limit the maximum number of subtrees visited while traversing the tree bottom up, i.e. from the leaf node to the root. In [3] a similar approach is proposed, where the stop criteria is imposed as a bound on the precision of the result. More recently, [15] proposed the use of an ensemble of trees where the split on each dimension is computed randomly and that rely on an unified priority queue: such approach is known as multiple randomized kd-trees, or randomized kd-forest. In [12], a library including several approximated NNS algorithms is proposed, including multiple randomized kd-trees [15], BBF kd-tree [4] and hierarchical k-means [8]. In addition, [12] also proposes a method to automatically determine the best algorithm and its parameters given the current dataset. Such, library, known as Fast Library for Approximate Nearest Neighbors (FLANN), is one of the most used libraries for NNS on point clouds: for example, it is the

default choice for NNS within the Point Cloud Library (PCL) [1], the reference library for 3D computer vision and robotic perception.

Although all the aforementioned methods for approximated NNS on points clouds can be used also on range images simply by turning this 3D data representation into a point cloud, it is possible to leverage on the organized trait of such data representation to speed up the search. Nevertheless, exploiting the 2D grid available when dealing with range images is not trivial, since nearest neighbor on the 2D grid are not guaranteed to be nearest neighbors also in 3D space (think about two points lying nearby on the image plane but on two different sides of a depth border). Furthermore, and especially for the Radius Search case, it is not trivial to turn a metric radius into a pixel-wise radius in the general case, when calibration data are not available.

In literature, the specialization of NNS to the case of range images is almost unexplored. One of the most relevant techniques is the one implemented in the PCL library for both the Radius Search and the kNN Search, where the main idea is to adaptively define the extent in pixels of the search area on the image based on the 3D data as well as the camera parameters. In particular, in the kNN Search case, the query point is first projected onto the range image by explicitly taking into account the intrinsic camera parameters and the camera pose. In case the projected point lays outside the range image, the nearest element in the image is used as start position. Then, the first k nearest neighbors are sought for by looking in the nearby positions on the image plane. The search area in pixel is then defined based on the distance, projected on the image plane, of the query to the farthest point among the found k neighbors, which is finally searched exhaustively to refine the list of retrieved neighbors. Instead, in the Radius Search case, the intrinsic parameters and the camera pose are used also to translate the input metric radius into the pixel-wise radius of the search area on the image plane, by projecting the estimated 3D spherical neighborhood onto the image plane of the sensor. Additionally, each neighbor on the range image 2D grid is also checked with respect to its 3D distance from the query before being added to the list of neighbors. Notably, this radius search algorithm is similar to the one presented in [10], where NNS is applied to the specific task of normal estimation on range images.

In this work we present a method, dubbed *Radial Search Method* (RSM), which can be used as an alternative to the methods available in PCL [1] for fast approximate NNS on range images. The idea of our approach is, starting from the query point, to incrementally look for neighbors on the 2D grid along radial regions of increasing radius. Specific stop conditions are employed to terminate the search when the candidates obtained are estimated to approximate well enough the real set of neighbors. In particular, we propose two variants of such approach, derived for both the kNN Search and the Radius Search problems. Notably, and advantageously with respect to [1,10], our method does not require to know neither the intrinsic parameters of the sensor nor the sensor pose. By means of experimental results on 3D data obtained with a consumer RGB-D camera, we evaluate the performance of our approach against the NNS methods

in PCL and against the FLANN randomized kd-forest approach in terms of both accuracy as well as efficiency. In addition, we also show how RSM can help improving the performance of a typical 3D computer vision application of NNS such as 3D keypoint detection and description.

2 RSM Algorithm

Our approach is based on the incremental exploration of the neighborhood starting from the query point, along concentric frames. While in the NNS search for range images available in PCL [1], hereinafter referred to as *organized*, the search is carried out over an image sub-region row after row (*raster scan*), in our method the search is made in radial order as depicted in Fig. 1. This allows our algorithm to evaluate less points in the neighborhood of the query point to obtain a similar level of approximation in the search result. Another important characteristic of our exploration strategy is that it is adaptive, i.e. the size of the 2D neighborhood that is considered changes at each query point by evaluating a stop condition that depends on the improvements of the search at each step.

Let q be the query index and e an element of the range image. If the query point is only available as a point in 3D space, it is projected onto the image and the nearest element to the projection is used as starting point q. We define a non-euclidean distance $\hat{D}(q, e)$ on a range image I as the minimum number of horizontal, vertical or diagonal moves to reach e from q. We also use the classic euclidean distance $D(q, e)$ to measure the distance in 3D space between the (x, y, z) points corresponding to q and e. Furthermore, we define $\mathcal{Q}$ as a min-priority queue of (e, p) pairs in which e is the index of a range image element and $p = D(q, e)$ is the priority key. Finally, we define the frame at distance h from a query point q as:

$$f_h(q) = \{e \,|\, \hat{D}(q, e) = h\}. \tag{1}$$

Figure 2 shows the elements belonging to the sets of the first 3 frames, i.e. $f_1(q)$, $f_2(q)$ and $f_3(q)$, of a query point q.

The key idea is to explore, at every iteration, the space of 3D point candidates defined by one full frame of pixels around the query point, and to stop the search

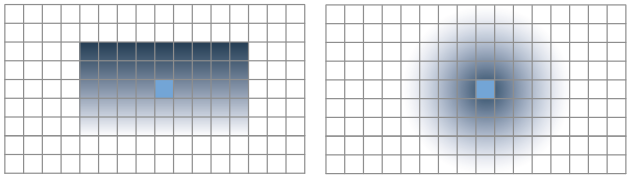

Fig. 1. Depiction of *Organized* (left) and RSM (right) search strategies: the light-colored square in the middle is the query point, while the darker a cell is the earlier it is explored. (Color figure online)

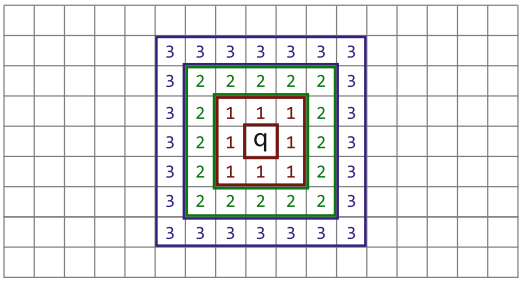

Fig. 2. Elements in the set of frames $f_1(q)$, $f_2(q)$ and $f_3(q)$.

whenever the number of *inliers* (e.g., found nearest neighbors) in the currently explored frame is too low. It is also important to make the exploration robust with respect to the presence of invalid points in the range image. In fact in many practical cases the acquired range images contains several invalid points, due to limitations in the sensing range or in dealing with specific surfaces, such as dark and reflective surfaces for active sensors, non-Lambertian surfaces and low-textured regions for passive sensors. Figure 3 shows how the algorithm has to work in the presence of invalid points. Even if an invalid point is encountered along the exploration of a frame, the search must continue beyond it. In the extreme case of one or more frames of invalid points, the stop condition should offer a setup that allows to continue the search in the next valid frames to be able to find other valid neighbors.

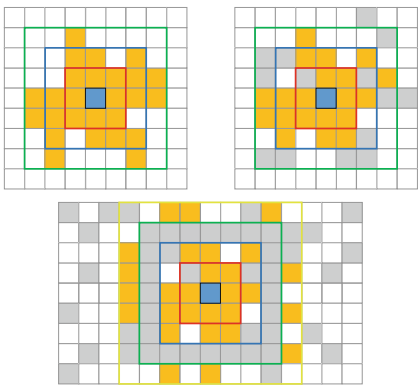

Fig. 3. The solution is represented by yellow points, while in gray we denote invalid elements. On the left, an example of the solution in the absence of invalid points; on the right, an example of the same neighborhood where some of the points are invalid, e.g. due to a change in viewpoint. Bottom: a case where an entire frame is composed of invalid points. The valid points in the neighborhood should still be evaluated and inserted in the solution.

```
function RSM_KNN(K, q, δ̃)
    i ← 1
    Q ← (q, 0)
    while f_i(q) ∈ image do
        ι ← 0
        ν ← 0
        for all e ∈ f_i(q) do
            if is_valid(e) then
                ν ← ν + 1
                if Q.size() < K then
                    Q.push(e, D(e, q))
                    ι ← ι + 1
                else if Q.top > D(e, q) then
                    Q.pop()
                    Q.push(e, D(e, q))
                    ι ← ι + 1
                end if
            end if
        end for
        if ν > 0 then
            δ ← δ + (ν − ι)/ν
        else
            δ ← δ + 1
        end if
        if δ > δ̃ then break
        if ι > 0 then δ ← 0
        i ← i + 1
    end while
    return Q
end function
```

Fig. 4. Pseudo-code for the RSM algorithm, kNN Search mode.

To meet all these goals, in our proposal two statistics are accumulated while exploring each frame: one is the number of valid candidates $\nu(f_i(q))$, i.e. all neighboring points with valid 3D coordinates, the other one is the number of inliers for the current search, $\iota(f_i(q))$. At the end of the exploration of each frame, the following stop condition is tested:

$$\delta(f_i(q)) = 1 - \frac{\nu(f_i(q)) - \iota(f_i(q))}{\nu(f_i(q))} > \tilde{\delta} \tag{2}$$

where $\tilde{\delta}$ is a user defined parameter. Intuitively, if the percentage of outliers (i.e., 1 minus the percentage of inliers) for the current frame is greater than a pre-defined threshold, the search is terminated.

To take into account the possibility that multiple frames are entirely composed of invalid points or outliers we allow the parameter $\tilde{\delta}$ to take integer values greater than 1. In this case, the parameter counts the number of frames entirely

```
function RSM_RADIUS(Radius, q, δ̃)
    i ← 1
    ℒ ← (q, 0)
    while f_i(q) ∈ image do
        ι ← 0
        ν ← 0
        for all e ∈ f_i(q) do
            if is_valid(e) then
                ν ← ν + 1
                if D(q, e) < Radius then
                    ℒ.push(e, D(e, q))
                    ι ← ι + 1
                end if
            end if
        end for
        if ν > 0 then
            δ ← δ + (ν − ι)/ν
        else
            δ ← δ + 1
        end if
        if δ > δ̃ then break
        if ι > 0 then δ ← 0
        i ← i + 1
    end while
    return ℒ.sort()
end function
```

Fig. 5. Pseudo-code for the RSM algorithm, Radius Search mode.

composed of invalid points or outliers to be consecutively met before stopping the search. Every time such kind of frames are met, we set

$$\delta\left(f_i\left(q\right)\right) = \delta\left(f_{i-1}\left(q\right)\right) + 1 \tag{3}$$

and then check the stop condition.

The pseudo-code of the kNN Search can be found in Fig. 4. For each query point, the algorithm keeps the discovered neighbors in a priority queue $\mathcal{Q}$, which holds the sorted result at the end of the search. $\mathcal{Q}$ is a min-priority queue in which the order is based on the distance between the query point and the element. Starting form the query point, we evaluate the frames in order of increasing distance, push each valid element of the frame in the priority queue if needed, and check if the termination criterion is met after processing every frame. After initialization of the data structures, the search starts from the first frame and continues until the stop criterion is met. In particular, the algorithm accumulates for each frame the number of valid examined candidates ν, as well as the number of those currently included in the nearest neighbor set, i.e. the inliers ι. With this value, it updates the percentage of outliers in the explored frames, δ, taking

Fig. 6. Examples showing one object view of each of the four datasets used in our experiments.

into account the previously exposed rules in case of fully invalid frames. When such percentage exceeds the user provided parameter $\tilde{\delta}$, the search ends.

The pseudo-code of the Radius Search can be found in Fig. 5. For each query point, the algorithm keeps all the elements that have distance less than the radius parameter R. All the points are successively stored in a list $\mathcal{L}$, which is sorted at the end of the search to return the points in distance order. The overall structure of the algorithm is similar to the kNN Search, but due to the nature of the radius search, if a point is pushed into the list it is never removed because it surely belongs to the final solution. For the same reason, the list is not kept sorted during the exploration, and is just sorted at the end, to save computation time.

The only step that may introduce approximations in the result in both algorithms is the stop criterion, the results being identical to the linear search one in the case of exploration of the whole range image. Therefore, the parameter $\tilde{\delta}$ trades off search accuracy for efficiency: since an unnecessary high accuracy negatively affects run-time performance, it is important to choose the right value of such parameter. In the Experimental results section we will analyze the sensitivity of the RSM algorithm to this parameter and provide guidelines on how to choose it appropriately.

3 Experimental Results

In this section, we provide an experimental evaluation of the RSM method. The method has been implemented in C++, and it is here compared with the randomized kd-tree forest algorithm available in FLANN [12], as a representative of the state of the art for approximated NNS on point clouds, as well as with the *organized* NNS algorithm available in PCL [1], as a representative of approximated NNS algorithms for range images. The comparison has been done on a PC equipped with an Intel Xeon E312xx 2.00 GHz (4 cores) processor with 8 Gb of RAM. We have compiled our framework under Visual Studio 2013 with optimization O2, and inline function expansion level set to Ob2. No evaluated algorithm includes any kind of parallelization, so the tests are always run on a single core.

The experiments were performed on four datasets composed of RGB-D images acquired with a Kinect dataset, recently proposed in literature and publicly available[1]. These datasets were originally proposed for the task of point cloud registration, and each of them includes different views of an object without the background: they are denoted here as *Frog*, *Mario*, *Squirrel* and *Duck*. A sample view for each dataset is shown in Fig. 6. The measured average distance of each point from its nearest neighbor on this data is approximately 1 mm. Each dataset includes at least 13 range images. On each range image, 1000 query points have been randomly extracted from the available valid 3D point set, and the results averaged over this set.

We evaluate both the execution times and the accuracy achieved by the tested algorithms. To measure the accuracy, the NNS for each query point has been also carried out by a brute force algorithm performing an exhaustive investigation, and used as ground truth in order to count the number of correctly retrieved neighbors by each approximated NNS algorithm.

3.1 Parameter Sensitivity Analysis

The first experimental analysis we carried out is a sensitivity analysis with the goal of choosing a good value for parameter $\tilde{\delta}$, which is the main parameter the RSM algorithm relies on, in both kNN and Radius versions. In particular, as anticipated, such parameter trades-off accuracy for efficiency: the higher it is, the more precise the outcome of the search will be compared to that of an exhaustive search, but also the longer the whole process will take.

Figure 7 reports the charts relatively to the results, in terms of accuracy and efficiency, on the evaluated datasets where each curve is associated to different values of parameter $\tilde{\delta}$. The two top charts report results in the kNN Search case, while the two bottom charts are relative to the Radius Search case. In each case, the left chart measures the relative search accuracy with respect to the exhaustive search (number of correct neighbors found), while the right chart reports the average time to process 1000 query points in a range image. In the kNN Search case, the x axis reports increasing values of k, while in the Radius Search case, it reports increasing values of the radius (in meters), with values typically used in most applications of such 3D NNS algorithms. In particular, the tests were performed using, for the k parameter, a range of values between 2 and 150, while for the Radius parameter we have chosen a range from 0.005 to 0.030 m.

From the charts related to the accuracy, RSM shows to be equivalent to the exhaustive search if the value of $\tilde{\delta}$ is greater than or equal to 1. Yet, the drop in performance is limited even if $\tilde{\delta}$ is set to low values: the worst result we get is to retrive 92 % of the real neighbors when using $\tilde{\delta} = 0.5$ in the radius search. This result confirms the intution that a radial search can be a good exploration pattern for NNS and shows that the statistics used to define the stop condition are able to limit the explored neighbors to the most interesting

[1] http://www.vision.deis.unibo.it/lrf.

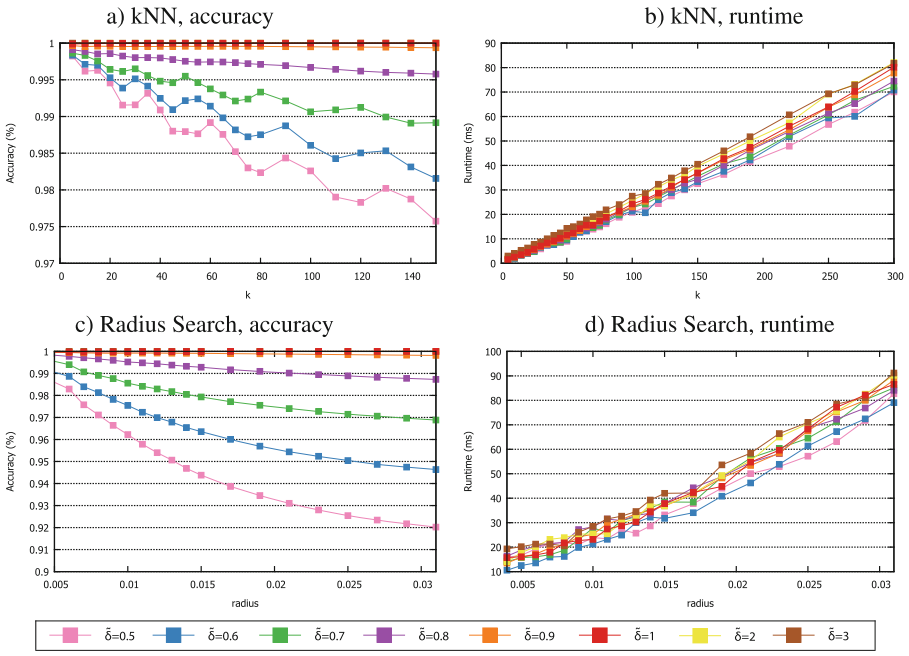

Fig. 7. Sensitivity analysis for the parameter $\tilde{\delta}$. The accuracy is reported on the left charts, the runtime on the right ones. Top charts: kNN Search; bottom charts: Radius Search. The scale of the y axis in (a) and (c) has been expanded for better visualization of the results.

ones. At the same time, looking at the reported runtime, as expected the lower the value of the parameter, the faster the overall efficiency. The gap between different approximation levels increases the larger k or radius get. As the gap in runtime is limited between the different choices of $\tilde{\delta}$, in the remainder of the experimental Section, we will employ the value of $\tilde{\delta} = 1$, given that it yields the highest efficiency among those reporting perfect accuracy.

3.2 Comparison with the State of the Art

Figure 8 shows the results in term of accuracy and runtime reported by the evaluated methods (RSM, FLANN and *Organized*) on the test dataset. As before, the top charts report the kNN Search case, while the bottom charts are relative to the Radius Search case, each chart showing the results at increasing values of the k and the radius parameter. In each case, the left chart measures the relative search accuracy with respect to the exhaustive search (number of correct neighbors found), while the right chart reports the average runtime over all the images of the datasets when processing 1000 query points on each range image.

Interestingly, in terms of accuracy all methods report, in both Search cases, a negligible loss of accuracy with respect to the exhaustive investigation. As far

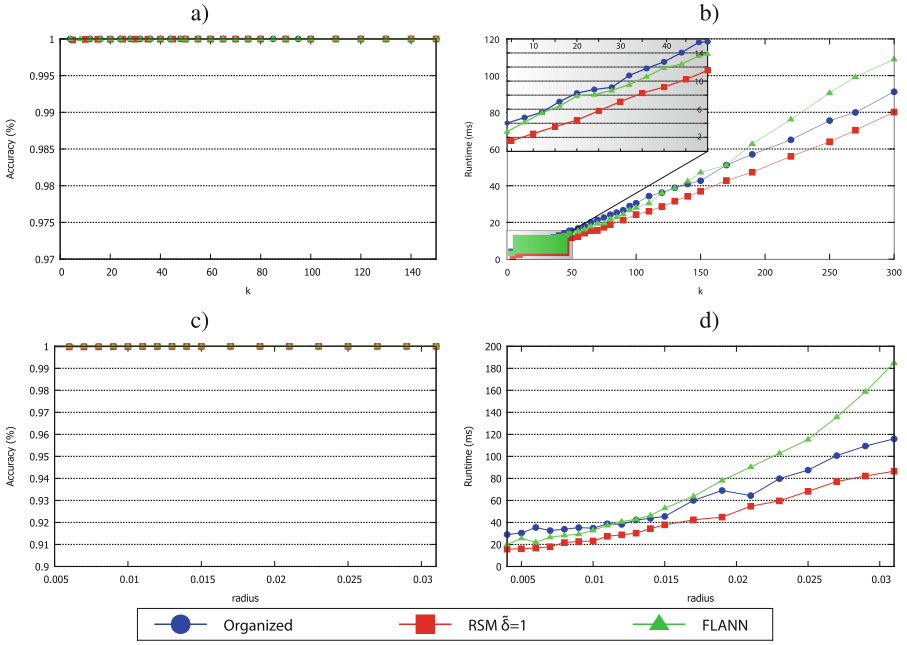

Fig. 8. Accuracy (a, c) and runtime (b, d) reported by RSM, FLANN and *Organized* methods on the evaluated dataset. The scale of the y axis in (a) and (c) has been expanded for better visualization of the results.

as the runtime is concerned, FLANN shows to scale worse than the other methods when k or the radius increases. Nevertheless, when just a few neighbors are sought, it turns out faster than *Organized*, which is surprising given that *Organized* has been specifically conceived for range images. RSM is consistently the fastest for all k and radii. This confirms that taking advantage of the structure inherent to range images can improve NNS efficiency with respect to using a general purpose solution like FLANN, and that a more natural exploration pattern like the radial one deployed by RSM can explore more promising areas of the image first and terminate the search earlier than the raster-scan search deployed by *Organized*.

3.3 Relevance of NNS in Keypoint Detection

To complement previous results, we have compared FLANN, *Organized* and RSM when used within a real and widely deployed application of the NNS problem such as 3D keypoint detection. As for the datasets, we have used the same data used in the previous experiments. In this case, results have been measured in terms of overall runtime of the whole detection process, so to measure out how much the computational advantage brought in by RSM impact in terms of the whole application. In addition, we have also measured the accuracy of the NNS

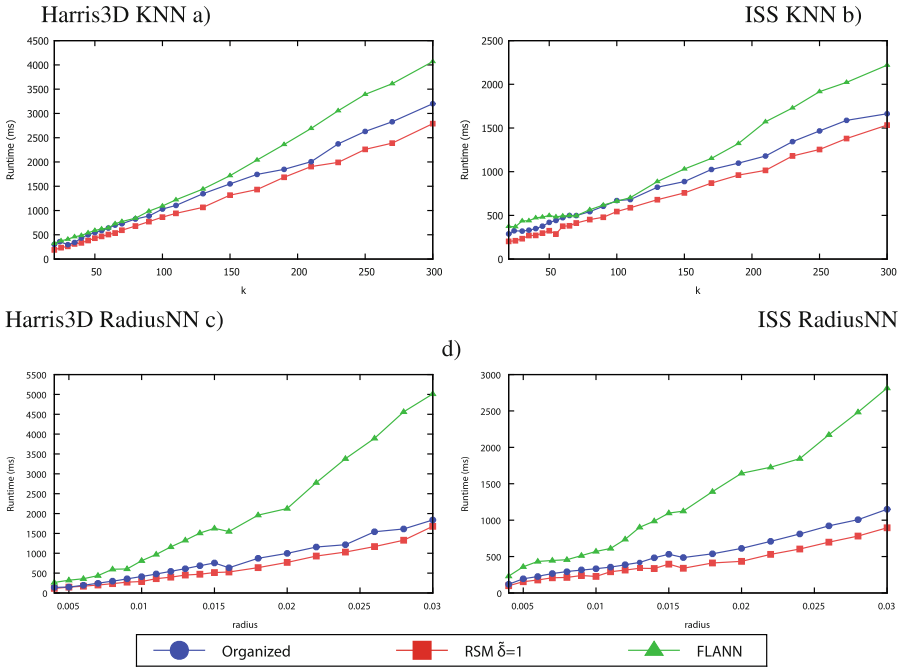

Fig. 9. Runtime reported, respectively, by the Harris3D detector (a, c) and the ISS detector (b, d) when employing, respectively, RSM, FLANN and *Organized* methods for the NNS.

in terms of the final application, i.e. by computing the relative repeatability [17] between the extracted 3D keypoints on pairs of overlapping views. To perform the repeatability evaluation, we have exploited the registration ground truth available with the dataset, that provides the 3D translation and 3D rotation registering each view into all other overlapping ones.

As for the choice of the 3D detectors, based on the analysis in [17] we have selected the Intrinsic Shape Signatures (ISS) detector [18], which provides a good trade-off between repeatability, distinctiveness and computational efficiency. In addition, we have also included in the comparison the Harris3D detector [1], which is an extension of the Harris corner detector [9] to the 3D case. For both detectors, we have used the available implementation in PCL [1]. As typically done by most 3D keypoint detectors [17], both methods rely on a NNS at each point of the range image to compute a local saliency: the extrema of such saliency are then used to localize distinctive keypoints. We have appropriately modified the code so to use, for the NNS, one method among RSM, *Organized* and FLANN, which have been tested both in the kNN Search as well as in the Radius Search version.

Figure 9 shows the results in terms of overall detection time for Harris3D (left charts) and ISS (right charts), both in the kNN Search (top charts) and in the

Radius Search (bottom charts) case. These charts show that by deploying RSM, we can clearly reduce the overall time required to perform keypoints detection, especially when analyzing structures at larger scales, i.e. those defined by larger k or radii, and confirm the practical importance of designing efficient methods to solve the NNS in 3D data.

3.4 Relevance of NNS in Descriptor Computation

In Fig. 10 are shown the results relative to the descriptors tests. On the vertical axis are shown the perfomances relative to the mean time for a descriptor computation. As in the previous tests on the detectors the runtime is evaluated on the Kinect datasets (Fig. 6), the keypoints used are obtained using ISS detectors. Since PCL implements only radius searchability for each descriptor algorithm only the RadiusNN search is compared in the tests. In the figures is possible to see how much the performances are influenced by the neighborhood search. First of all using Fast Point Feature Histograms (FPFH) [14] both RSM and Organized are faster than FLANN (Fig. 10a), but since the main computation is spent around the non-search part (the runtime axis is expressed in thousands of milliseconds) they are very close in the diagram, more explicative results were obtained using Signature of Histograms of OrienTations (SHOT) [16] and Spin Images [11] (respectively Fig. 10b, c), looking closer is possible to see the improvement given by RSM over both Organized and FLANN methods. Furthermore, comparing descriptors results is impossible to notice differences between a description obtained using FLANN algorithm compared to one obtained using RSM or Organized.

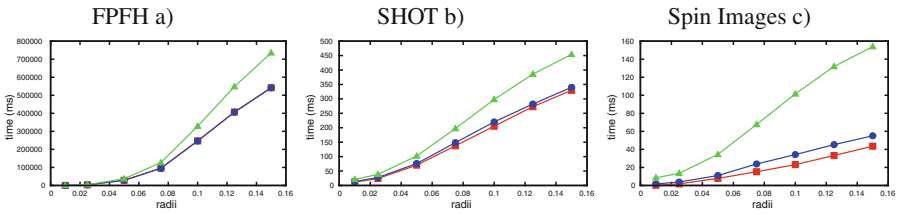

Fig. 10. Mean runtime reported for each keypoint descriptor evaluation, respectively we have FPFH (a), SHOT (b) and Spin Images (c), every descriptor was evaluated with RadiusNN search on a ISS type keypoint.

4 Concluding Remarks

In this paper, we have proposed a new method for NNS on range images, dubbed RSM, which proves to be faster than available algorithms for NNS on this kind of organized data, while preserving the same level of accuracy as the currently employed NNS methods for points clouds and range images. In particular, RSM is

able to leverage on the organized structure of range images and on effective stop conditions applied while exploring the neighborhood radially from the query point to terminate the search process as soon as the currently probed locations do not seem to contain additional nearest neighbors. RSM proved to provide computational savings both in the Radius Search as well as in the kNN Search case. Furthermore, the method proved to be particularly effective for speeding up 3D keypoint detection and description without reducing the quality of the results in terms of repeatability and distinctiveness.

References

1. Point cloud library. http://pointclouds.org/
2. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS, pp. 459–468 (2006)
3. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J. ACM **45**(212), 891–923 (1998)
4. Beis, J., Lowe, D.: Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1997)
5. Besl, P., McKay, N.D.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)
6. Burkhard, W.A., Keller, R.M.: Some approaches to best-match file searching. Commun. ACM **16**(4), 230–236 (1973)
7. Freidman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. **3**(3), 209–226 (1977)
8. Fukunaga, K., Narendra, P.: A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans. Comput. 750–753 (1975). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1672890
9. Harris, C., Stephens, M.: A combined corner and edge detector. In: Procedings of the Alvey Vision Conference 1988, pp. 147–151 (1988). http://www.bmva.org/bmvc/1988/avc-88-023.html
10. Holzer, S., Rusu, R.B., Dixon, M., Gedikli, S., Navab, N.: Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In: IEEE International Conference on Intelligent Robots and Systems, pp. 2684–2689 (2012)
11. Johnson, A., Hebert, M.: Surface matching for object recognition in complex 3D scenes. Image Vis. Comput. **16**, 635–651 (1998)
12. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: International Conference on Computer Vision Theory and Applications (VISAPP 2009), pp. 1–10 (2009). papers2://publication/uuid/3C5A483A-ADCA-4121-A768-8E31BB293A4D
13. Puglia, L., Vigliar, M., Raiconi, G.: SASCr3: a real time hardware coprocessor for stereo correspondence. In: Campilho, A., Kamel, M. (eds.) ICIAR 2014. LNCS, vol. 8815, pp. 383–391. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11755-3_43
14. Rusu, R., Blodow, N., Beetz, M.: Fast point feature histograms (FPFH) for 3D registration. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan (2009)

15. Silpa-Anan, C., Hartley, R.: Optimised KD-trees for fast image descriptor matching. In: 26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2008)
16. Tombari, F., Salti, S., Stefano, L.: Unique signatures of histograms for local surface description. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6316, pp. 356–369. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15558-1_26
17. Tombari, F., Salti, S., Di Stefano, L.: Performance evaluation of 3D keypoint detectors. Int. J. Comput. Vis. **102**, 198–220 (2013)
18. Yu, Z.: Intrinsic shape signatures: a shape descriptor for 3D object recognition. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009, pp. 689–696 (2009)