Highly Parallelizable Algorithm for Keypoint Detection in 3-D Point Clouds

Jens Garstka and Gabriele Peters

Abstract In computer vision a reliable recognition and classification of objects is an essential milestone on the way to autonomous scene understanding. In particular, keypoint detection is an essential prerequisite towards its successful implementation. The aim of keypoint algorithms is the identification of such areas within 2-D or 3-D representations of objects which have a particularly high saliency and which are as unambiguous as possible. While keypoints are widely used in the 2-D domain, their 3-D counterparts are more rare in practice. One of the reasons often consists in their long computation time. We present a highly parallelizable algorithm for 3-D keypoint detection which can be implemented on modern GPUs for fast execution. In addition to its speed, the algorithm is characterized by a high robustness against rotations and translations of the objects and a moderate robustness against noise. We evaluate our approach in a direct comparison with state-of-the-art keypoint detection algorithms in terms of repeatability and computation time.

Keywords 3-D keypoint detection · 3-D object recognition · 3-D computer vision

1 Introduction

3-D object recognition and classification is a fundamental part of computer vision research. Many of the existing recognition systems use local feature based methods as they are more robust to occlusion and clutter. In those systems keypoint detection should be the first major step to get distinctive local areas for discriminative local feature descriptions. But a recent survey on feature based 3-D object recognition

J. Garstka (✉) · G. Peters
Faculty of Mathematics and Computer Science, Human-Computer Interaction,
FernUniversität in Hagen, University of Hagen, 58084 Hagen, Germany
e-mail: jens.garstka@fernuni-hagen.de
URL: http://www.fernuni-hagen.de/mci

G. Peters
e-mail: gabriele.peters@fernuni-hagen.de

J. Filipe et al. (eds.), *Informatics in Control, Automation and Robotics 12th International Conference, ICINCO 2015 Colmar, France, July 21-23, 2015 Revised Selected Papers*, Lecture Notes in Electrical Engineering 383, DOI 10.1007/978-3-319-31898-1_15

systems by Guo et al. [7] shows, that many major systems for local feature based 3-D object recognition use sparse sampling or mesh decimation methods to create a set of points on which the feature descriptors will be computed. In terms of repeatability and informativeness these methods do not result in qualified 3-D keypoints.

There are a variety of reasons why existing 3-D keypoint detection algorithms are not used: Some of them are sensitive to noise and some are time consuming. Furthermore, there is only a handful of methods, that work with unstructured 3-D point clouds without a time consuming approximation of local surface patches or normal vectors [5, 12–14, 20, 21].

This paper addresses the problems described above. The proposed method is a fast and robust algorithm for automatic identification of 3-D keypoints in unstructured 3-D point clouds. We create a filled and watertight voxel representation (in terms of a dense voxel connectivity) of a point cloud. This voxel representation is convolved with a spherical convolution kernel. The sphere works as an integral operator on all voxels containing points of the point cloud. The convolution gives the proportion of voxels of the sphere which are inside the point cloud. The proportion values are used to identify regions of interest, and from these robust keypoints are extracted. All parts of the algorithm are highly parallelized and thus will be computed very quickly. The size of the convolution kernel can be adopted to the size of the area which is used by the local feature descriptor. Furthermore, we can easily simulate a lower resolution point cloud by increasing the voxel size. Therefore, we can create keypoints for multiple resolutions. Finally we will show, that our approach provides robust keypoints, even if we add noise to the point cloud.

2 Related Work

There are many 3-D keypoint detection algorithms that work on meshes or use surface reconstruction methods. A brief overview is given in a recent survey paper by Guo et al. [7]. But there are only a few of them that work directly on unstructured 3-D point cloud data. They have been compared multiple times, e.g., by Salti et al. [15], Dutagaci et al. [3], and Filipe and Alexandre [4] and therefore, we will give just a short overview of algorithms, which are designed to work with point clouds only.

Pauly et al. [14] use a principal component analysis to compute a covariance matrix C for the local neighborhood of each point $\mathbf{p}$. With the eigenvalues λ_1, λ_2 and λ_3 they introduce the surface variation $\sigma_n(\mathbf{p}) = \lambda_1/(\lambda_1 + \lambda_2 + \lambda_3)$, for a neighborhood of size n, i.e., the n nearest neighbors to $\mathbf{p}$. Within a smoothed map of surface variations Pauly et al. do a local maxima search to find the keypoints. A major drawback of this method is, that the surface variation is sensitive to noise (Guo et al. [7]).

Matei et al. [12] use a similar approach as Pauly et al., but they use only the smallest eigenvalue λ_3 of the covariance matrix C for a local neighborhood of a point $\mathbf{p}$ to determine the surface variation. But in contrast to Pauly et al., the method from Matei et al. provides only a fixed-scale keypoint detection.

The algorithm presented by Flint et al. [5] is a 3-D extension of 2-D algorithms like SIFT [11] and SURF [2] called THRIFT. They divide the spatial space by a uniform voxel grid and calculate a normalized quantity D for each voxel. To construct a density scale-space Flint et al. convolve D with a series of 3-D Gaussian kernels $g(\sigma)$. This gives rise to a scale-space $S(\mathbf{p}, \sigma) = (D \otimes g(\sigma))(\mathbf{p})$ for each 3-D point $\mathbf{p}$. Finally, they compute the determinant of Hessian matrix at each point of the scale space. Within the resulting $3 \times 3 \times 3 \times 3$ matrix, a non maximal suppression reduces the entries to local maxima, which become interest points.

Unnikrishnan and Hebert [20] introduce a 3-D keypoint detection algorithm based on an integral operator for point clouds, which captures surface variations. The surface variations are determined by an exponential damped displacement of the points along the normal vectors weighted by the mean curvature. The difference between the original points and the displaced points are the surface variations which will be used to extract the 3-D keypoints, i.e., if a displacement is an extremum within the geodesic neighborhood the corresponding 3-D point is used as keypoint.

Zhong [21] propose another surface variation method. In their work they use the ratio of two successive eigenvalues to discard keypoint candidates. This is done, because two of the eigenvalues can become equal and thus ambiguous, when the corresponding local part of the point cloud is symmetric. Apart from this, they use the smallest eigenvalue to extract 3-D keypoints, as proposed by Matei et al.

Finally, also Mian et al. [13] propose a surface variation method. For each point $\mathbf{p}$ they rotate the local point cloud neighborhood in order to align its normal vector $n_{\mathbf{p}}$ to the z-axis. To calculate the surface variation they apply a principal component analysis to the oriented point cloud and use the ratio between the first two principal axes of the local surface as measure to extract the 3-D keypoints.

3 Our Algorithm

The basic concept of our algorithms is adopted from a keypoint detection algorithm for 3-D surfaces introduced by Gelfand et al. [6]. To be able to use an integral volume to calculate the inner part of a sphere without structural information of the point cloud, we designed a volumetric convolution of a watertight voxel representation of the point cloud and a spherical convolution kernel. This convolution calculates the ratio between inner and outer voxels for all voxels that contain at least one point of the 3-D point cloud. The convolution values of the point cloud get filled into a histogram. Keypoint candidates are 3-D points with rare values, i.e., points corresponding to histogram bins with a low level of filling. We cluster these candidates, find the nearest neighbor of the centroid for each cluster, and use these points as 3-D keypoints.

Thus, our method for getting stable keypoints in an unstructured 3-D point cloud primarily consists of the following steps:

1. Estimate the point cloud resolution to get an appropriate size for the voxel grid.
2. Transfer the point cloud to a watertight voxel representation and fill all voxels inside of this watertight voxel model with values of 1.
3. Calculate a convolution with a voxel representation of a spherical convolution kernel.
4. For each 3-D point fill the convolution results of its corresponding voxel into a histogram.
5. Cluster 3-D points with rare values, i.e., 3-D points of less filled histogram bins, and use the centroid of each cluster to get the nearest 3-D point as stable keypoint.

The details of these steps are provided in the sections below.

3.1 Point Cloud Resolution

A common way to calculate a point cloud resolution is to calculate the mean distance between each point and its nearest neighbor. Since we use the point cloud resolution to get an appropriate size for the voxel grid (which is to be made watertight in the following steps), we are looking for a voxel size which leads to a voxel representation of the point cloud, where only a few voxels corresponding to the surface of the object remain empty, while as much as possible of the structural information is preserved.

To get appropriate approximations of point cloud resolutions, we carried out an experiment on datasets obtained from 'The Stanford 3-D Scanning Repository' [19]. We examined the mean Euclidean distances between n nearest neighbors of m randomly selected 3-D points, with $n \in [2, 10]$ and $m \in [1, 100]$. The relative difference between the number of voxels based on the 3-D points to the number of voxels based on the triangle mesh were filled into a separate histogram for each value of n. The histogram of the 'Stanford Bunny' shown in Fig. 1 is illustrative for all results.

With $n = 7$ the absolute mean of the relative difference is at a minimum. Thus, the experiments using different 3-D objects show that an approximated point cloud resolution with the use of 7 nearest neighbors and with a sample size of 50 randomly selected points is a good choice to get a densely filled initial voxel grid within a small computation time.

3.2 Fast Creation of a Watertight 3-D Model

Initially we create a voxel grid of cubic voxels with an edge length of the point cloud resolution as described above. Each voxel containing a 3-D point is initialized with a value of 1.0. This creates an approximated voxel representation of the surface. The voxels representing the point cloud are defined as watertight, if the voxels result in a densely connected structure without gaps.

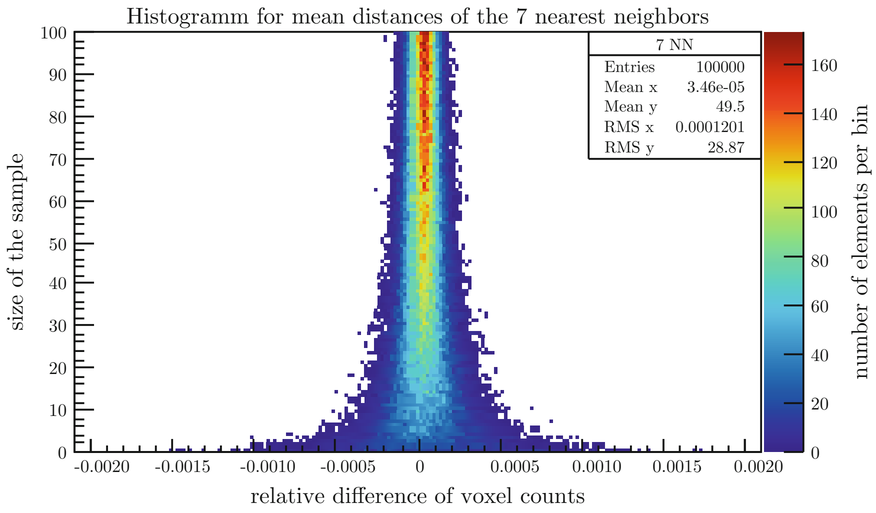


Fig. 1 The quality of the voxel grid based on the point cloud resolution calculated by the mean of $n = 7$ nearest neighbors. The x-axis shows the relative difference between the number of voxels based on the 3-D points to the number of voxels based on the triangle mesh of 'Stanford Bunny'. The y-axis shows the number of randomly selected points which have been used to calculate the values

To be able to fill inner values of objects, we assume that it is known whether the point cloud represents a closed model or a depth scan, which has its depth direction along the z-axis.

In case of a depth scan, which is very common in robotics, we take the maximum depth value z_{max} and the radius of the convolution kernel r_{conv}, and set the value of all voxels along the z-axis beginning with the first voxel with a value of 1 (a surface voxel) and ending with depth of at least $z_{max} + r_{conv}$ to a value of 1, too. An illustration of this can be seen in Fig. 2.

In case we assume a closed model we have to fill all inner voxels. Due to variations in density of the point cloud and the desire to keep the voxel size as small as possible, it often occurs that the voxel surface is not watertight. For first tests we implemented two different approaches to close holes in the voxel grid. The first implementation was based on the method from Adamson and Alexa [1]. Their approach is in fact intended to enable a ray tracing of a point cloud. They use spheres around all points of the point cloud to dilate the points to a closed surface. Shooting rays through this surface can be used to close holes. The second implementation was based on a method by Hornung and Kobbelt [8]. Their method creates a watertight model with a combination of adaptive voxel scales and local triangulations. Both methods create watertight models without normal estimation. But the major drawback of both methods consists in their long computation times.

Since the method we propose should be fast, these concepts were discarded for our approach. Instead we use a straightforward solution, which appears to be sufficient

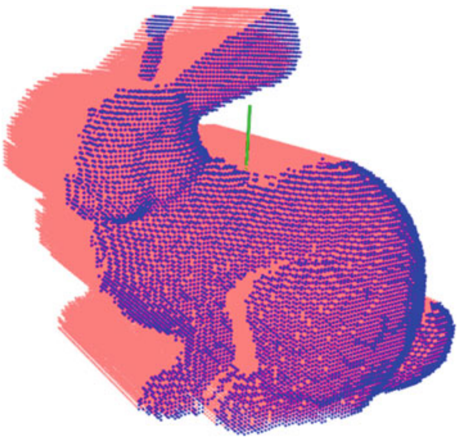


Fig. 2 This is a voxel representation of a view dependent patch (depth scan) from the 'Bunny'. *Blue dots* represent voxels corresponding to 3-D points. *Red dots* represent the filled voxels below the surface

for good but fast results. The filling of the voxel grid is described exemplary for the one direction. The calculation of the other directions is performed analogously.

Let u, v and w the indexes of the 3-D voxel grid in each dimension. For each pair (u, v) we iterate along all voxels in w-direction. Beginning with the first occurrence of a surface voxel (with a value of 1) we mark all inner voxels by adding -1 to the subsequent voxels until we reach the next surface voxel. These steps are repeated until we reach the w boundary of the voxel grid. If we added a value of -1 to the last voxel, we must have passed a surface through a hole. In this case we need to reset all values for (u, v) back to the previous values.

After we did this for each dimension, all voxels with a value ≤ -2, i.e., all voxel which have been marked as inner voxels by passes for at least two dimensions, will be treated as inner voxels and their value will be set to a value of 1. All other voxels will get a value of 0.

We already mentioned, that, if we pass the surface through a hole, we set back all voxel values to previous values. This might result in tubes with a width of one voxel (see Fig. 3a). Because of that, we fill these tubes iteratively in a post-processing step, with the following rule.

If a voxel at position (u, v, w) has a value of 0 and at least each of the 26 neighbor voxels except of one of the six direct neighbors ($u \pm 1$ or $v \pm 1$ or $w \pm 1$) has a value of 0, the voxel at (u, v, w) gets a value of 1, too. The result is shown in Fig. 3b.

3.3 Convolution

The convolution is done with a voxelized sphere of radius r_{conv}. For a fast GPU based implementation we use the NVIDIA FFT-implementation cuFFT. Figure 4a, b visualize the results showing those voxels which contain 3-D points from the initial point cloud. The convolution values are in [0, 1]. While values near 0 are depicted

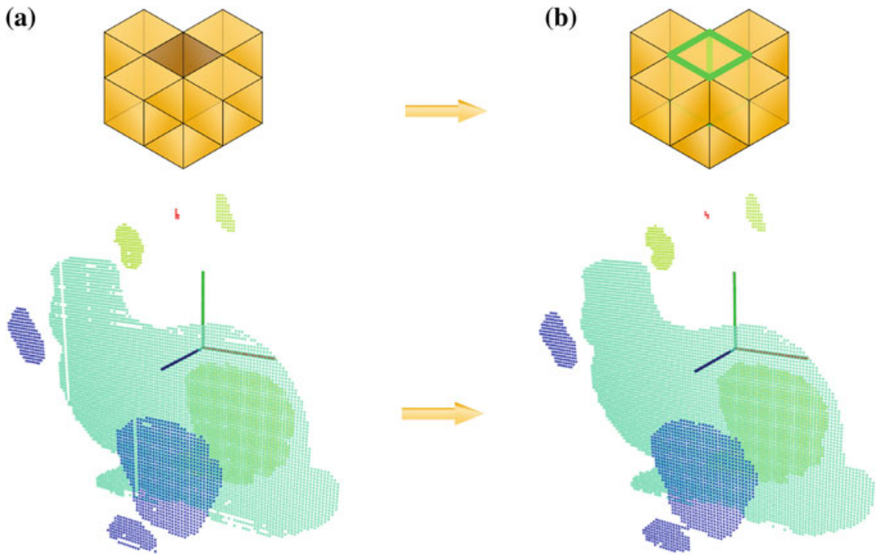


Fig. 3 The *lower part* of this figure shows sliced versions of the filled voxel grid of the 'Bunny'. All inner voxels get filled after the surface is passed. This is done for each direction u, v and w. If the surface gets missed through a hole, it might lead to tubes with a width of one voxel as shown in (**a**). If we post-process this voxel grid, we can identify and fill those tubes. This is done as shown in the *upper part* of this figure. The *upper part* of (**a**) shows a configuration, where a non-filled voxel (tinted in *brown*) is bounded by at least 5 filled direct neighbor voxels (tinted in *beige*). In such a case the voxel framed in green is filled as illustrated in the *upper part* of (**b**). This leads to a nearly complete filled voxel grid as shown in the *lower part* of (**b**)

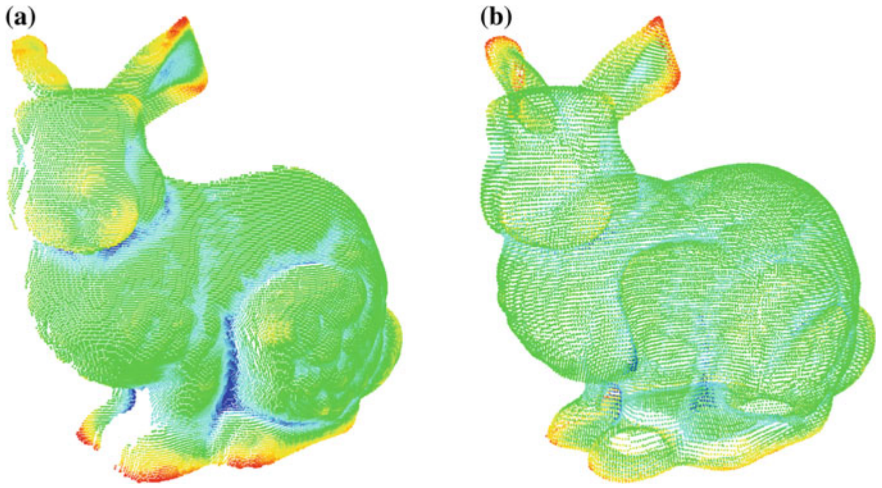


Fig. 4 The two figures show colored convolution results of the 'Bunny' in full resolution using a convolution kernel of radius $r_{conv} = 10pcr$ (pcr = point cloud resolution). **a** Shows a depth scan ($pcr = 0.00167$) with convolution values between 0.23 (*red*) and 0.79 (*blue*) while **b** shows the closed point cloud ($pcr = 0.00150$) with values between 0.08 (*red*) and 0.83 (*blue*)

red, values near 1 are depicted blue to purple, and values of about 0.5 are depicted green.

3.4 *Histograms*

Following the computation of the convolution, we have to identify all convolution values which are interesting, i.e., which are less frequent. To find those regions of values which are less frequent, we fill the convolution values into a histogram. To get an appropriate amount of bins we use Scott's rule [16] to get a bin width b:

$$b = \frac{3.49\sigma}{\sqrt[3]{N}}, \tag{1}$$

where σ is the standard deviation of N values. In case of the Stanford bunny the value of b is:

$$b = \frac{3.49 \cdot 0.096}{\sqrt[3]{35947}} \approx 0.01015 \tag{2}$$

The corresponding histogram is shown in Fig. 5.

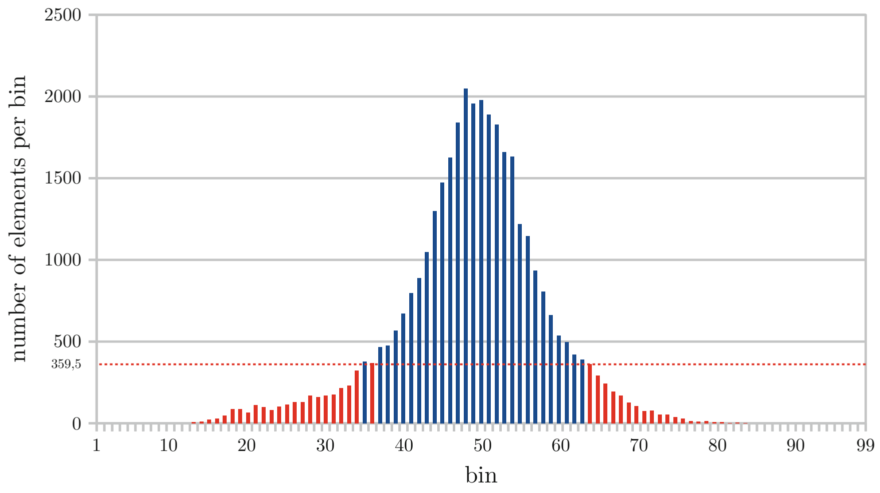


Fig. 5 This histogram illustrates the distribution of convolution values for the Stanford Bunny. The smallest convolution value is 0.12, while the largest convolution value is 0.85. With $\sigma = 0.096$ the interval [0, 1] is divided into 99 bins with a bin size of 0.0101. The *red dotted line* indicates the 1 % limit. All 3-D points with a convolution value of bins colored in *red* will be used as keypoint candidates

In case of a depth scan we need to take into account that the convolution values at outer margins do not show the correct values. Thus, we ignore all values of points within an outer margin r_{conv}.

3.5 Clustering

From the histogram we select all bins filled with values of at most 1 % of all points (see Fig. 5). It turned out that this is a good choice for an upper limit since higher values lead to large clusters. On the other hand, a significant smaller limit leads to fragmented and unstable clusters.

All points corresponding to the values in the selected bins will be used as keypoint candidates for clustering. Figure 6a shows all keypoint candidates for the 'Bunny'. We cluster these points using the Euclidean distance with a range limit of $3pcr$. This enables us to handle small primarily longish clusters, e.g., the region above the bunny's hind legs, as a single connected cluster. Figure 6b illustrates the different clusters with separate colors.

For each cluster we calculate the centroid. Each centroid is used to find its nearest neighbor among the 3-D points of the corresponding cluster. This nearest neighbor is used as 3-D keypoint. Figure 6c, d show the results for the 'Bunny'.

Additional examples of further objects from the Stanford 3-D Scanning Repository are given in the Appendix.

4 Results

We evaluated our results with respect to the two main quality features repeatability and computation time. To be comparable to other approaches we used the same method of comparing different keypoint detection algorithms and the same dataset as described by Filipe and Alexandre [4], i.e., we used the large-scale hierarchical multi-view RGB-D object dataset from Lai et al. [9]. The dataset contains over 200000 point clouds from 300 distinct objects. The point clouds have been collected using a turntable and an RGB-D camera. More details can be found in another article by Lai et al. [10].

4.1 Repeatability Under Rotation

Filipe and Alexandre [4] use two different measures to compare the repeatability: the relative and absolute repeatability. The relative repeatability is the proportion of keypoints determined from the rotated point cloud, that fall into a neighborhood of

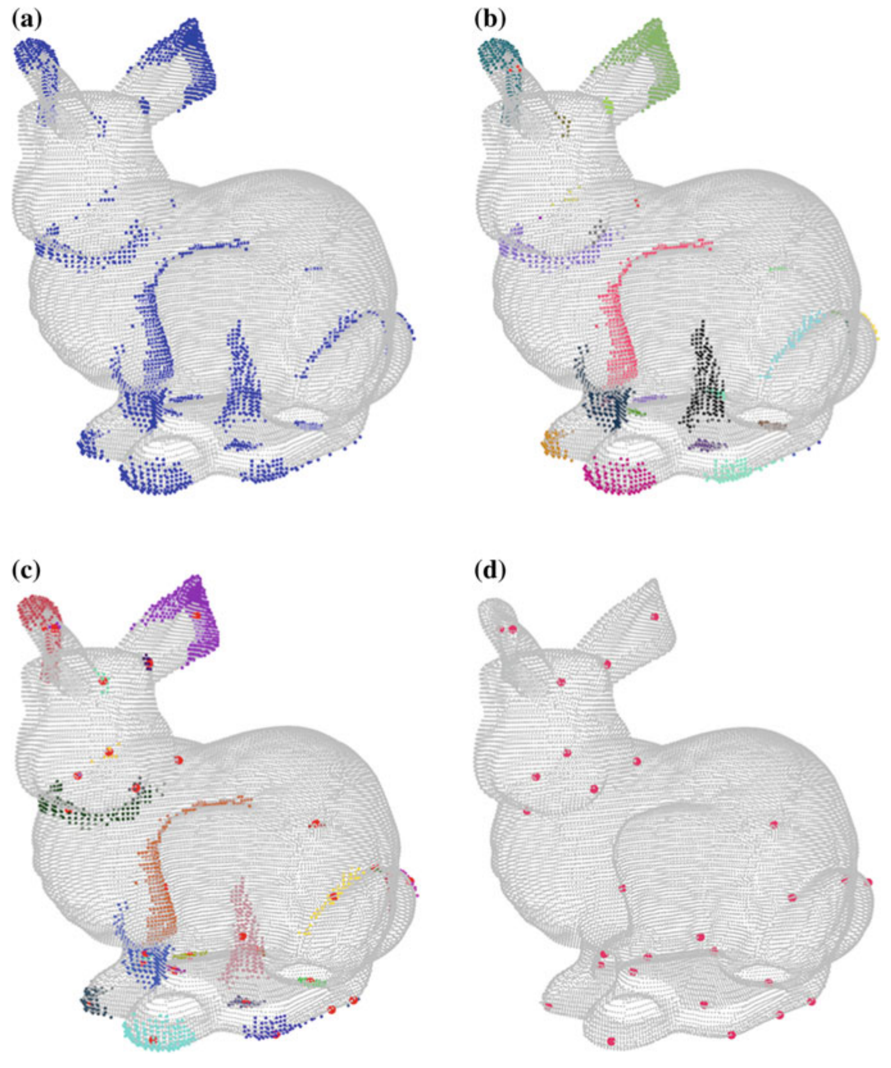


Fig. 6 These figures illustrate the process of obtaining clusters and keypoints from the set of possible candidates. **a** Shows all 3-D keypoint candidates (*blue*) for the 'Bunny'. **b** Shows all clusters found for an Euclidean distance with a range limit of $3pcr$. **c** Shows a combination of colorized clusters and the corresponding 3-D keypoints. Finally, **d** shows the isolated keypoints

a keypoint of the reference point cloud. The absolute repeatability is the absolute number of keypoints determined in the same manner as the relative repeatability.

To compute the relative and absolute repeatability, we randomly selected five object classes ('cap', 'greens', 'kleenex', 'scissor', and 'stapler') and picked 10

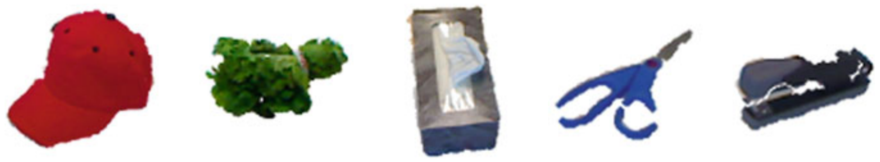


Fig. 7 The five object classes obtained from 'The Stanford 3-D Scanning Repository' [19] are *cap, greens, kleenex, scissor*, and *stapler*

point clouds within each object class randomly as well. A color image of one pose and one point cloud from each of the used object classes is shown in Fig. 7.

Additionally, these 50 base point clouds were rotated around random axes at angles of 5°, 15°, 25°, and 35°. Afterwards, we applied our algorithm on each of these 250 point clouds. For neighborhood sizes n from 0.00 to 0.02 in steps of 0.001 the keypoints of the rotated point clouds were compared with the keypoints of the base point clouds. If a keypoint of a rotated point cloud fell within a neighborhood n of a keypoint of the base point cloud, the keypoint was counted. Finally, the absolute repeatability was determined based on these counts for each n as an average of the 50 corresponding point clouds for each angle. The relative repeatability rates are the relations between the absolute repeatabilities of the rotated point clouds and the number of keypoints of the base point clouds.

Figure 8 opposes relative repeatability rates of our approach (left hand side) to the corresponding results of four state-of-the-art keypoint detection algorithms (right hand side). Figure 9 does the same for absolute repeatability rates. The approaches evaluated by Filipe and Alexandre [4] are Harris3D [17], SIFT3D [5], ISS3D [21], and SUSAN [18] which they extended to 3-D.

In more detail, the graphs on the left of both Figs. 8 and 9 show the average relative, resp. absolute, repeatability of keypoints computed with our algorithm for 5 randomly selected objects over 10 iterations, i.e., with 10 different rotation axes. The graphs on the right of both figures are taken from the evaluation done by Filipe and Alexandre [4].

It is striking, that the relative repeatability rates of our approach are considerably higher for large rotation angles than those of all other state-of-the-art approaches that have been compared by Filipe and Alexandre. Only for an rotation angle of 5° one of the other algorithms (ISS3D) is able to outperform our approach significantly in the range of small neighborhood radii. On the other hand, the results of our algorithm in terms of absolute repeatability of keypoints are in general less convincing, although for all of the considered rotation angles it outperforms one of the other approaches (Harris3D). For a rotation angle of 35° our algorithm outperformes even three of the other approaches considered (Harris3D, SIFT3D, and ISS3D).

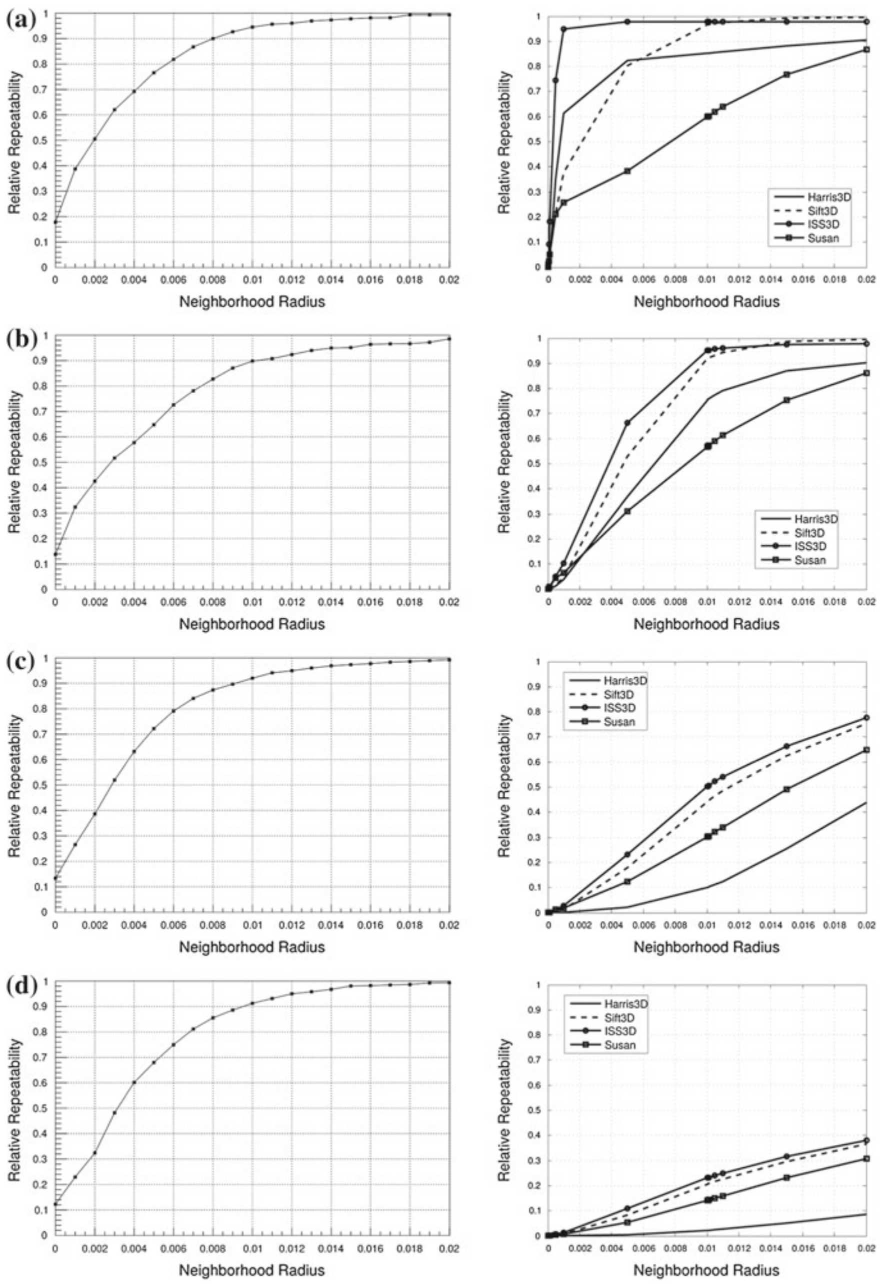


Fig. 8 Relative repeatability. *Left* our approach. *Right* four state-of-the-art approaches, diagrams taken from the evaluation done by Filipe and Alexandre [4]. **a** Relative repeatability of keypoints at a rotation angle of 5°, **b** Relative repeatability of keypoints at a rotation angle of 15°. **c** Relative repeatability of keypoints at a rotation angle of 25°. **d** Relative repeatability of keypoints at a rotation angle of 35°

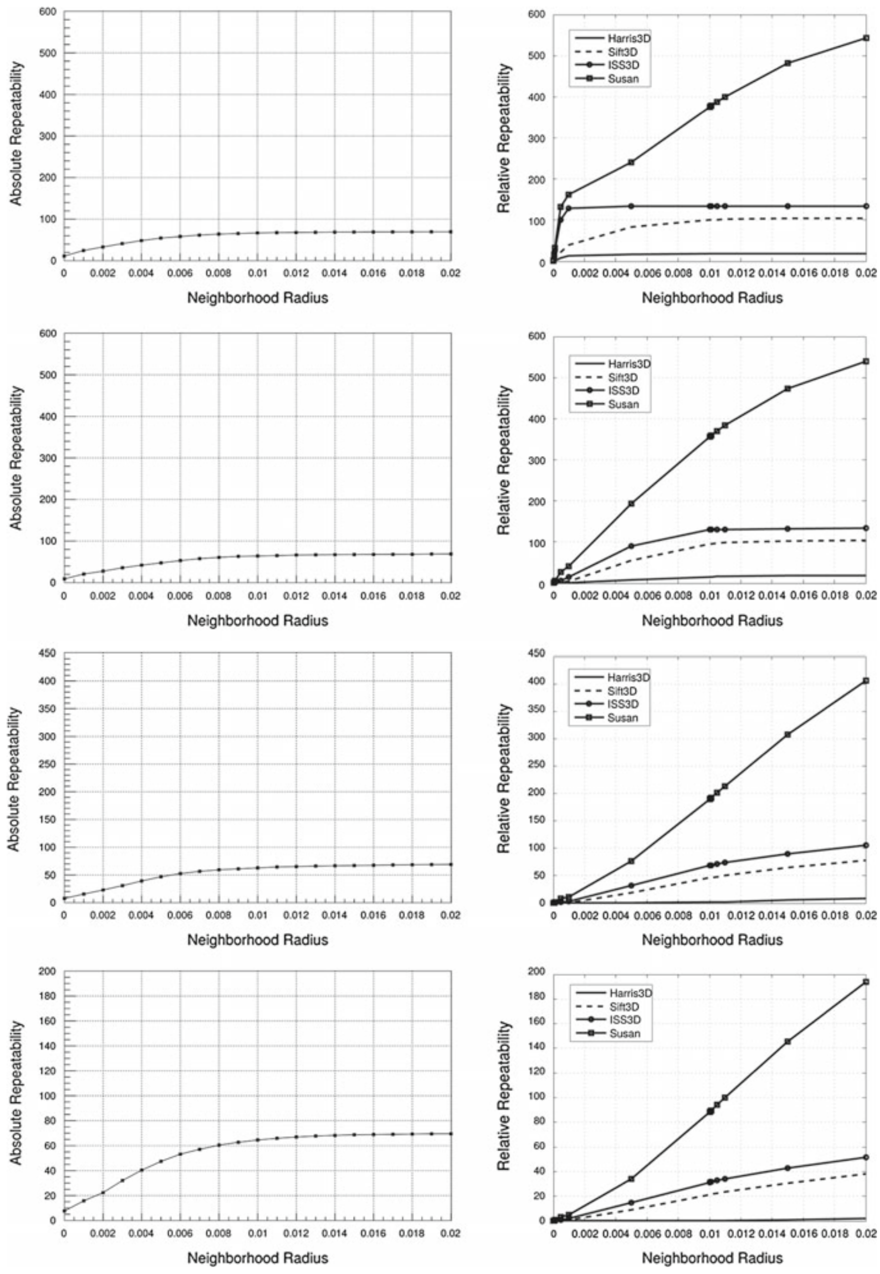


Fig. 9 Absolute repeatability. *Left* our approach. *Right* four state-of-the-art approaches, diagrams taken from the evaluation done by Filipe and Alexandre [4]. *Note* the diagrams taken from Filipe and Alexandre have wrong axis labels. However, they contain the absolute repeatability of keypoints. **a** Absolute repeatability of keypoints at a rotation angle of 5°. **b** Absolute repeatability of keypoints at a rotation angle of 15°. **c** Absolute repeatability of keypoints at a rotation angle of 25°. **d** Absolute repeatability of keypoints at a rotation angle of 35°

4.2 Repeatability Under Noise

Furthermore, we have repeated the described simulations for point clouds with additional random noise at a level of 0.5 times of the point cloud resolution. The graphs of Fig. 10 display the average repeatability rates (relative and absolute) for point clouds with added noise. The differences between these curves and their corresponding curves from the simulations without noise are negligible, which shows that our approach is able to cope with a fair amount of noise, as well.

4.3 Computation Time

To calculate average computation times we computed the 3-D keypoints 10 times for each point cloud. The system configuration we used for all experiments is given in Table 1.

The computation time of our algorithm correlates with the number of voxels, i.e., with the dimensions of the voxel grids, which must be powers of two. Measured average computation times in dependence of voxel grid dimensions are given in Table 2.

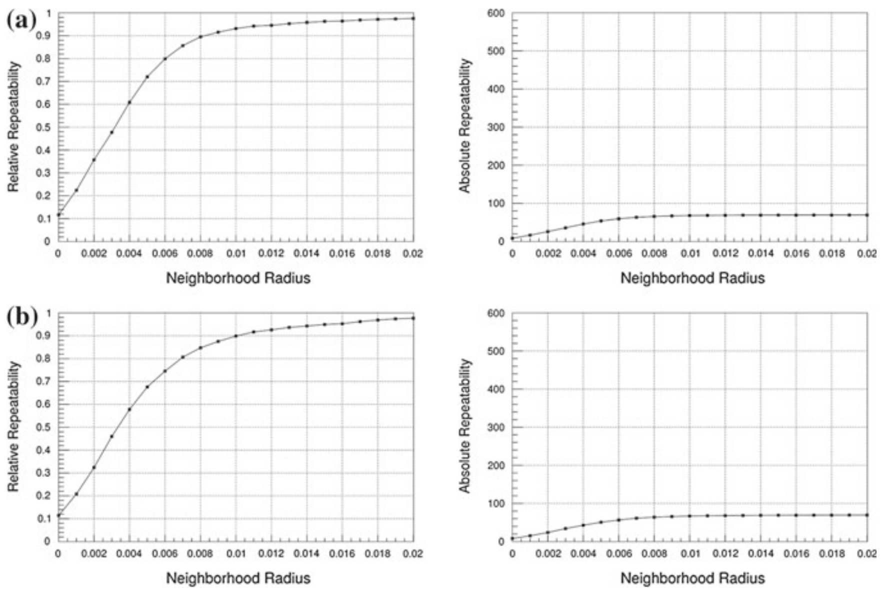


Fig. 10 Our approach with additional noise. *Left* relative repeatability. *Right* absolute repeatability. **a** Repeatability for a point cloud with $0.5pcr$ additional random noise at a rotation angle of 5°. **b** Repeatability for a point cloud with $0.5pcr$ additional random noise at a rotation angle of 15°

Table 1 System configuration for experiments

System configuration	
Processor	Intel Xeon E5630 @2.53 GHz
Memory	12 GB
GPU	NVIDIA GeForce GTX 670
GPU memory	2 GB GDDR5
OS	Debian GNU/Linux
NVIDIA driver	340.32

Table 2 Average computation times

Voxel grid	Time (ms)
$512 \times 256 \times 128$	≈ 1500
$256 \times 256 \times 128$	≈ 700
$256 \times 256 \times 64$	≈ 300
$256 \times 128 \times 128$	≈ 350
$256 \times 128 \times 64$	≈ 80
$128 \times 128 \times 128$	≈ 200
$128 \times 128 \times 64$	≈ 110
$128 \times 64 \times 64$	≈ 60
$64 \times 64 \times 64$	≈ 35

The computation time for most of the point clouds was below 1 s. For many of the point clouds the computation time fell within a range below 300 ms. The average computation time for all 250 point clouds which were used to compare the repeatability rates was 457 ms. This is considerably fast, especially in comparison to the average computation times of Harris3D (1010 ms) and ISS3D (1197 ms), which have been determined on the same system.

5 Conclusion

In this paper we have presented a fast and robust approach for extracting keypoints from an unstructured 3-D point cloud. The algorithm is highly parallelizable and can be implemented on modern GPUs.

We have analyzed the performance of our approach in comparison to four state-of-the-art 3-D keypoint detection algorithms by comparing their results on a number of 3-D objects from a large-scale hierarchical multi-view RGB-D object dataset.

Our approach has been proven to outperform other 3-D keypoint detection algorithms in terms of relative repeatability of keypoints. Results in terms of absolute repeatability rates are less significant. An important advantage of our approach is its speed. We are able to compute the 3-D keypoints within a time of 300 ms for most of the tested objects.

Furthermore, the results show a stable behavior of the keypoint detection algorithm even on point clouds with added noise. Thus, our algorithm might be a fast and more robust alternative for systems that use sparse sampling or mesh decimation methods to create a set of 3-D keypoints. Additional examples can be found in the appendix.

Appendix—Additional Examples

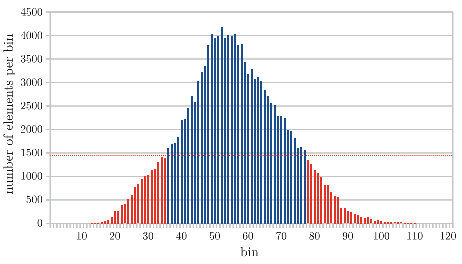
Happy Buddha

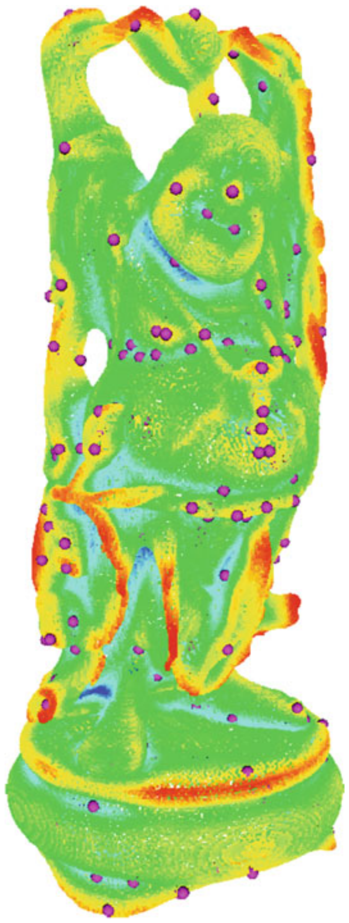
This object is obtained from 'The Stanford 3-D Scanning Repository' [19] and is characterized by the following properties:

- Points: 144647
- $pcr = 0.00071$
- Voxel grid: $135 \times 299 \times 135$
- $r_{conv} = 10 \cdot pcr$
- $\sigma = 0.124884$
- Bins: 121
- Keypoints: 210

The histogram below illustrates the distribution of convolution values for the 'Happy Buddha'. To save space the labels are not included in the histogram. They correspond to those shown in Fig. 5, i.e., the abscissa shows the bin number, while the ordinate shows the number of elements per bin.

The 3-D point cloud of the 'Happy Buddha' shown right is a combination of two types of figures which have already been used to illustrate the results of the 'Stanford Bunny'. The color gradient used to tint the point of the point cloud illustrates the convolution values from the smallest value (red) to the largest value (blue). This was already used in Fig. 4. The purple markers illustrate the final keypoints. This was already used in Fig. 6d.



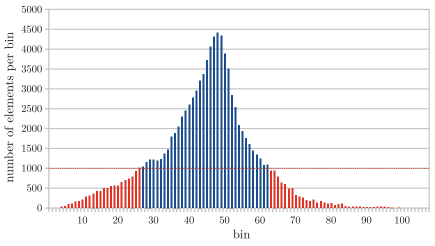


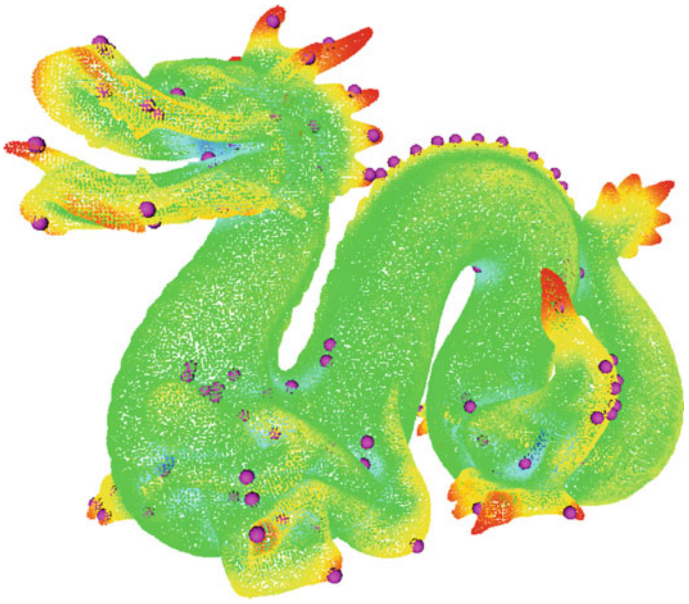
Dragon

This object is obtained from 'The Stanford 3-D Scanning Repository' [19] and is characterized by the following properties:

- Points: 100250
- $pcr = 0.00097$
- Voxel grid: $236 \times 174 \times 120$
- $r_{conv} = 10 \cdot pcr$
- $\sigma = 0.124507$
- Bins: 107
- Keypoints: 92

The histogram below illustrates the distribution of convolution values for the 'Dragon'. To save space the labels are not included in the histogram. They correspond to those shown in Fig. 5, i.e., the abscissa shows the bin number, while the ordinate shows the number of elements per bin.





The 3-D point cloud of the 'Dragon' shown above is a combination of two types of figures which have already been used to illustrate the results of the 'Stanford Bunny'. The color gradient used to tint the point of the point cloud illustrates the convolution values from the smallest value (red) to the largest value (blue). This was already used in Fig. 4. The purple markers illustrate the final keypoints. This was already used in Fig. 6d.

References

1. Adamson, A., Alexa, M.: Ray tracing point set surfaces. In: Shape Modeling International, pp. 272–279. IEEE (2003)
2. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: speeded up robust features. In: Computer Vision–ECCV 2006, pp. 404–417. Springer (2006)
3. Dutagaci, H., Cheung, C.P., Godil, A.: Evaluation of 3d interest point detection techniques via human-generated ground truth. Vis. Comput. **28**(9), 901–917 (2012)
4. Filipe, S., Alexandre, L.A.: A comparative evaluation of 3d keypoint detectors. In: 9th Conference on Telecommunications. Conftele 2013, Castelo Branco, Portugal, pp. 145–148 (2013)
5. Flint, A., Dick, A., Hengel, A.V.D.: Thrift: local 3d structure recognition. In: 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications, pp. 182–188. IEEE (2007)
6. Gelfand, N., Mitra, N.J., Guibas, L.J., Pottmann, H.: Robust global registration. In: Symposium on Geometry Processing, vol. 2, p. 5 (2005)
7. Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J.: 3d object recognition in cluttered scenes with local surface features: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **99**(PrePrints), 1 (2014)
8. Hornung, A., Kobbelt, L.: Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, pp. 41–50. SGP'06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006)
9. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1817–1824. IEEE (2011)
10. Lai, K., Bo, L., Ren, X., Fox, D.: A scalable tree-based approach for joint object and pose recognition. In: AAAI (2011)
11. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004)
12. Matei, B., Shan, Y., Sawhney, H.S., Tan, Y., Kumar, R., Huber, D., Hebert, M.: Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation. IEEE Trans. Pattern Anal. Mach. Intell. **28**(7), 1111–1126 (2006)
13. Mian, A., Bennamoun, M., Owens, R.: On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. Int. J. Comput. Vis. **89**(2–3), 348–361 (2010)
14. Pauly, M., Keiser, R., Gross, M.: Multi-scale feature extraction on point-sampled surfaces. In: Computer Graphics Forum, vol. 22, pp. 281–289. Wiley Online Library (2003)
15. Salti, S., Tombari, F., Stefano, L.D.: A performance evaluation of 3d keypoint detectors. In: International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), pp. 236–243. IEEE (2011)
16. Scott, D.W.: On optimal and data-based histograms. Biometrika **66**(3), 605–610 (1979)
17. Sipiran, I., Bustos, B.: Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. Vis. Comput. **27**(11), 963–976 (2011)
18. Smith, S.M., Brady, J.M.: Susana new approach to low level image processing. Int. J. Comput. Vis. **23**(1), 45–78 (1997)
19. The stanford 3d scanning repository (2014). http://graphics.stanford.edu/data/3Dscanrep
20. Unnikrishnan, R., Hebert, M.: Multi-scale interest regions from unorganized point clouds. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08, pp. 1–8. IEEE (2008)
21. Zhong, Y.: Intrinsic shape signatures: a shape descriptor for 3d object recognition. In: IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 689–696. IEEE (2009)