Pattern Recognition Letters 103 (2018) 8–15

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Hough Transform for real-time plane detection in depth images

Eduardo Vera[a,*], Djalma Lucio[b], Leandro A.F. Fernandes[a], Luiz Velho[b]

[a] Instituto de Computação, Universidade Federal Fluminense (UFF), Niterói, RJ 24210-240, Brazil
[b] Instituto Nacional de Matemática Pura e Aplicada (IMPA), Rio de Janeiro, RJ 22460-320, Brazil

ARTICLE INFO

Article history:
Received 26 April 2017
Available online 2 January 2018

MSC:
68T45
68U10

Keywords:
Hough Transform
Plane detection
Depth image
Real time

ABSTRACT

The automatic detection of planes in depth images plays an important role in computer vision. Plane detection from unorganized point clouds usually requires complex data structures to pre-organize the points. On the other hand, existing detection approaches tailored to depth images use the structure of the image and the 2.5-D projection of the scene to simplify the task. However, they are sensitive to noise and to discontinuities caused by occlusion. We present a real-time deterministic technique for plane detection in depth images that uses an implicit quadtree to identify clusters of approximately coplanar points in the 2.5-D space. The detection is performed by an efficient Hough-transform voting scheme that models the uncertainty associated with the best-fitting plane with respect to each cluster as a Gaussian distribution. Experiments shows that our approach is fast, scalable, and robust even in the presence of noise, partial occlusion, and discontinuities.

1. Introduction

The growing popularity of depth cameras like Structure Sensor and Project Tango makes the 3-D scanning technology cheaper and ubiquitous. Consequently, it motivates the development of real-time solutions with applications in image-based reconstruction, autonomous vehicle navigation, and augmented reality. The ability to detect planar structures from depth images in real-time is an important step towards the development of these solutions.

Existing techniques for detecting planar structures from unorganized point clouds are not suitable for handling depth data in real time because they often require special data structures to organize points in 3-D space, or exploit computationally intensive non-deterministic strategies. In contrast, surface growing techniques that detect planar patches in depth images are tailored to the 2.5-D structure of depth data. However, the quality of detections relies on the amount of noise and on the absence of occlusions and missing portions of depth information, which may lead to multiple detections of the same plane.

We present a real-time deterministic solution for plane detection in depth images. Our approach extends the kernel-based voting scheme for plane detection in unorganized 3-D point clouds developed by Limberger and Oliveira [1] by exploiting the 2.5-D nature of depth data to organize input points into simple and efficient 2-D data structures. By taking advantage of the regular lattice of the discrete image plane, our solution avoids re-computations and performs plane detection in real-time on a personal computer, with linear cost over the number of input pixels. In addition, it is robust to noise typically found in this kind of image and to multiple detections of the same plane, even in the presence of occlusion and missing data.

The *main contribution* of this paper is a real-time plane detection scheme for depth images: the Depth Kernel-Based Hough Transform (D-KHT), which has asymptotic time complexity $O(n)$ on the number of pixels, and whose implementation runs 3 to 14 times faster than state-of-the-art techniques.

Fig. 1 shows a result obtained by the D-KHT. Each color in Fig. 1(c) represents a different plane detected in the depth image in Fig. 1(b). Black regions correspond to non-planar surfaces.

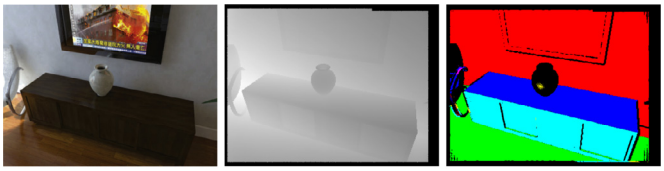
2. Related work

The Standard Hough Transform (SHT) [2] is one of the most popular techniques for detecting geometric entities in low-dimensional spaces. The SHT for straight-line detection discretizes the space defined by the line parameters as an accumulator map. Given a binary edge image, the accumulator's bins related to the lines passing through each edge pixel are incremented by one unit per voting pixel. The local maxima created by this voting process correspond to the parameters of the most likely lines in the image.

Fernandes and Oliveira [3] proposed the Kernel-Based Hough Transform (KHT), a real-time algorithm for straight line detection

* Corresponding author.
E-mail addresses: eduardovera@ic.uff.br (E. Vera), dlucio@impa.br (D. Lucio).
URL: http://www.ic.uff.br/~laffernandes (L.A.F. Fernandes), http://www.lvelho.impa.br (L. Velho)

https://doi.org/10.1016/j.patrec.2017.12.027
0167-8655/



(a) Color image (b) Input depth image (c) Output

Fig. 1. Result produced by the proposed approach. Detected planes are identified in (c) by colors, while black pixels correspond to non-planar object. For this example, the D-KHT runs in ∼21 ms on a 2.3 GHz PC and behaves well, even in the presence of non-planar surfaces.

which first builds clusters of approximately collinear edge pixels and then uses these clusters to vote in the accumulator map, considering the Gaussian uncertainty of the line that better explains each cluster as the voting kernel. Limberger and Oliveira [1] extended the KHT to the detection of planes in an unorganized point cloud in 3-D space (3-D KHT). They use an octree to organize the point cloud in such a way that each leaf node of the tree includes a cluster of approximately coplanar points, or a set of outliers. The latter leaves are neglected in the voting process. Experiments show that the 3-D KHT outperforms previous HT-based techniques for plane detection.

The Probabilistic Hough Transform (PHT) [4] performs the HT procedure for plane detection on a random subset of m points from the input point cloud. Estimating the optimal value for m is a challenging task because one has to deal with the trade-off between noise resilience and processing cost. The Adaptive Probabilistic Hough Transform (APHT) [5] replaces the selection of m by an adaptive stopping rule that terminates voting as soon as a given number of objects seem to be reliably detected. Unfortunately, the APHT is sensitive to noise. In the Progressive Probabilistic Hough Transform (PPHT) [6], the voting procedure is also performed for a randomly selected subset of input entries. However, the number of points that have already voted define a threshold for the peaks of votes considered as valid detections. When a plane is detected, the points that lie on it are deleted from the input set. The adaptive thresholding and the point deletion strategies make the PPHT less sensitive to noise than the APHT. The Randomized Hough Transform (RHT) [7] also removes points related to detected planes from the point set. The voting strategy of the RHT relies on voting for the single cell corresponding to the plane spanned by random point triples. Refer to [8] for a detailed comparison of the probabilistic variations of the HT for plane detection.

Surface Growing (SG) [9,10] is a region-growing-based technique to detect planar patches in depth images by using a smoothness constraint. It selects seed pixels in the image and expands the area around those pixels if the underlying surface can be correctly grouped into a flat surface. SG works well in the presence of a small amount of noise, but has a high computational cost and the quality of detection is tied to the selection of the seed points. Furthermore, SG requires connectivity between pixels depicting the same surface in order to avoid multiple detections of the same plane. Such connectivity may not be observed in depth images since occlusion and depth shadowing may cause damaged and missing depth information.

The Random Sample Consensus (RANSAC) [11] is an iterative non-deterministic method to estimate parameters of a mathematical model. At each iteration, the technique randomly selects the minimal amount of input entries required to adjust the model of some intended type of structure (e.g., three points define a plane). Then, it counts how many entries from the input dataset can be explained by the fitted model (inliers). The iterative process stops when the probability of finding a model with more inliers than the current best model is below a given threshold. Its main advantage is noise resilience. Unfortunately, most of the computational cost is devoted to manage models computed from unrelated input entries. Schnabel et al. [12] introduced an optimization to RANSAC using an octree to establish spatial proximity among samples, improving the detection of planes, spheres, cylinders, cones, and tori from points in 3-D spaces.

Erdogan et al. [13] proposed the use of Markov Chain Monte Carlo for the plane detection problem. This technique provided good results. However, it is not able to reach real-time frame rates due to its computational cost.

Hemmat et al. [14] proposed a plane-detection technique based on finding 3-D edges in depth images and searching lines between those edges. Hemmat et al.'s approach prevents the computation of normal vectors, reducing the overall computational cost of the solution. However, it only reaches real time performance by multithreaded implementation.

For a detailed review on object detection and recognition in depth images and point clouds, please refer to [15].

3. Depth Kernel-Based Hough Transform

Fig. 2 illustrates the flowchart of the proposed plane detection pipeline: (i) Starting from a depth image, our approach uses Summed-Area Tables (SATs) [16] and a quadtree [17] to efficiently subdivide the set of pixels into clusters of approximately coplanar points, in 3-D; (ii) for each cluster, it uses first-order error propagation to estimate the uncertain plane that fits clustered points; then, each cluster votes in an accumulator map using the trivariate-Gaussian kernel computed from the plane's associated uncertainties; and (iii) after the voting step, the location of local maxima in the accumulator map correspond to the most likely planes.

3.1. Clustering depth data

The clustering step builds the implicit quadtree on the depth image using the whole image as the root node (Fig. 3). For each node having at least s_{ms} samples (*i.e.,* points in 3-D computed from valid depth pixels), we use Principal Component Analysis (PCA) to decide whether the current set of samples define a cluster of coplanar samples or must be subdivided according to the quadrants of the current rectangular image region represented by the node. If a node does not have enough samples then it is a leaf node. In this case, its samples are considered outliers and do not participate to the voting process. Fig. 3(b) depicts outliers as black nodes. For a node to be considered as having approximately coplanar samples, we must calculate its mean sample $\mu_{(x,y,z)}$ and covariance matrix $\Sigma_{(x,y,z)}$ as:

$$\mu_{(x,y,z)} = \begin{pmatrix} \mu_x \\ \mu_y \\ \mu_z \end{pmatrix} \quad \text{and} \quad \Sigma_{(x,y,z)} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix}, \tag{1}$$

moreover, check whether $2\sqrt{\lambda_1} < s_t$, where λ_1 is the least eigenvalue of $\Sigma_{(x,y,z)}$ and s_t is the maximum spread a distribution of samples may have in its thinnest dimension. In (1),

$$\sigma_{ab} = \frac{1}{m-1}\left(\sum_{k=1}^{m} a_k\, b_k - \mu_b \sum_{k=1}^{m} a_k - \mu_a \sum_{k=1}^{m} b_k + m\,\mu_a\,\mu_b\right) \tag{2}$$

and

$$\mu_a = \frac{1}{m}\sum_{k=1}^{m} a_k, \quad \mu_b = \frac{1}{m}\sum_{k=1}^{m} b_k, \tag{3}$$

where m is the number of samples in a given node, and $\{a_k\}_{k=1}^{m}$ and $\{b_k\}_{k=1}^{m}$ are the coordinates x, y, or z of samples in the unit of

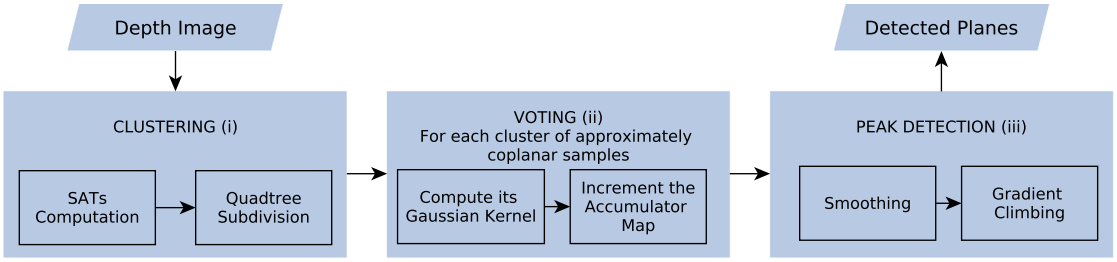
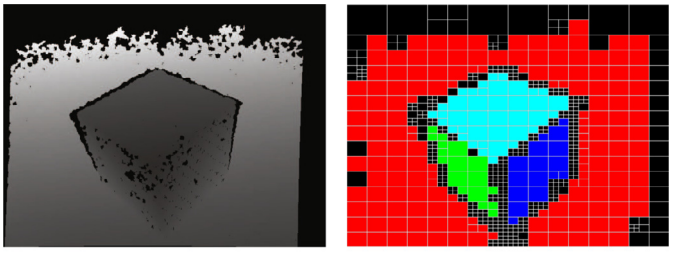


Fig. 2. D-KHT workflow: (i) Given a depth image as input, its pixels are clustered into approximately coplanar regions; (ii) For each region identified in step (i), we vote on the spherical accumulator map considering the Gaussian distribution that describes the uncertainty of the best-fitting plane for the region; (iii) Finally, we use a hill climbing approach to find peaks of votes in the accumulator map, whose coordinates correspond to the most likely planes in the image.



(a) Input depth image (b) Clustered input

Fig. 3. The *Cube* dataset consists in a depth image of a cube on a planar surface (a). Black pixels in (a) depict invalid depth information and do not correspond to 3-D point samples. Image (b) shows the leaf nodes of the implicit quadtree built over the depth image. Nodes having the same color indicate clusters of point samples that voted to the same detected plane. Black nodes in (b) correspond to outliers. They did not participate to the voting process.

measurement used by the device:

$$\begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = \begin{pmatrix} \frac{i_k - c_x}{\alpha_x} z_k \\ \frac{j_k - c_y}{\alpha_y} z_k \\ z_k \end{pmatrix}. \tag{4}$$

In (4), we assume a pinhole camera having skew equal to zero. $(i_k, j_k)^T$ are the pixel coordinates of the k-th sample point. $(c_x, c_y)^T$ is the camera's principal point, in pixels, and α_x and α_y represent focal length in terms of pixels. In our experiments, the camera parameters were retrieved from the datasets or from the calibration procedure of the device.

In contrast to the 3-D KHT, we consider only a single parameter (s_t) to guide the clustering step. According to our experience, the single-layer spread of feature points in 3-D is influenced by the relative angle between the principal axis of the camera and the normal to the plane, making the definition of relative isotropy used by the 3-D KHT a challenging task. Besides that, handling a single parameter is more intuitive to the user. Also, depth images allow the implicit construction of a quadtree instead of the explicit construction of the octree adopted by the 3-D KHT, because the depth image is a height map from the camera's perspective, leading to the organization of 3-D points considering only the 2-D coordinate system of the image.

When using PCA on depth images, one must be aware that the depth value, differently of pixel coordinates, is usually given in the standard unit of measurement assumed by the capturing device. Thus, in order to compute $\Sigma_{(x,y,z)}$ one has to map valid pixels to the actual 3-D space of the scene (see (4)).

By pre-computing SATs (nine, altogether), we are able to compute each $\Sigma_{(x,y,z)}$ in constant time since each summation in the covariance (2) and mean (3) formulas can be evaluated by using only four SAT references and three arithmetic operations.

3.2. Computing Gaussian kernels

Let $\mathcal{C}$ be a cluster containing approximately coplanar point samples in a leaf node of the quadtree. The plane that better fits the samples in $\mathcal{C}$ passes through the mean point $\mu_{(x,y,z)}$ and has unit normal vector $\vec{n} = (n_x, n_y, n_z)^T$, where $\vec{n}$ is the eigenvector associated to the least eigenvalue of $\Sigma_{(x,y,z)}$. The Gaussian kernel that weights votes from $\mathcal{C}$ in the parameter space of the normal equation of the plane is centered at:

$$\mu_{(\rho,\phi,\theta)} = \begin{pmatrix} \mu_\rho \\ \mu_\phi \\ \mu_\theta \end{pmatrix} = \begin{pmatrix} n_x \mu_x + n_y \mu_y + n_z \mu_z \\ \cos^{-1}(n_z) \\ \tan^{-1}\left(\frac{n_y}{n_x}\right) \end{pmatrix}. \tag{5}$$

The covariance matrix of the Gaussian kernel in the parameter space can be computed using first-order error propagation as:

$$\Sigma_{(\rho,\phi,\theta)} = J\, \Sigma_{(x,y,z)} J^T, \tag{6}$$

where J is the Jacobian of (5), and $\Sigma_{(x,y,z)}$ is defined by (1).

3.3. Spherical accumulator map

We have adopted Borrmann et al.'s [8] accumulator map in our approach. However, we assume the camera's center and its principal axis as, respectively, the origin and the z-axis of the 3-D Cartesian space. In their work, [8] demonstrated that an unbiased spherical accumulator map $\mathcal{A}$ is ideal for HTs tailored to detect arbitrary planes in 3-D spaces. The axes of $\mathcal{A}$ are $\theta \in [-\pi, +\pi)$, $\phi \in [0, \pi)$, and $\rho \in [0, \rho_{high}]$, whose upper limit ρ_{high} defines the radius of $\mathcal{A}$. We choose ρ_{high} as the distance between the camera and the farthest point sample. The discrete values assumed by ρ in $\mathcal{A}$ are defined by linear interpolation of the interval $[0, \rho_{high}]$. The amount of linear samples in ρ and ϕ directions (N_ρ and N_ϕ, respectively) are defined by the user based on his/her expectations on the granularity of detection results. The linear discretization of θ is given as a function of ϕ (see for details [8]).

3.4. Kernel-based voting

We use the Gaussian kernels computed according to Section 3.2 to update the spherical accumulator map $\mathcal{A}$ described in Section 3.3. For a given kernel defined by a mean parameter vector $\mu_{(\rho,\phi,\theta)}$ (5) and a covariance matrix $\Sigma_{(\rho,\phi,\theta)}$ (6), we increment the bins of $\mathcal{A}$ that falls within two standard deviations of the mean. Therefore, the contribution of a kernel is considered significant only at the bins whose parameter vector $q = (\rho, \phi, \theta)^T$ satisfies:

$$f(q) \geq f(\mu_{(\rho,\phi,\theta)} + 2\sqrt{\kappa}\,\vec{u}), \tag{7}$$

where $\{\vec{u}, \kappa\}$ is any eigenpair of $\Sigma_{(\rho,\phi,\theta)}$, and f denotes the probability density function (PDF) of the Gaussian distribution:

$$f(q) = \frac{1}{\sqrt{(2\pi)^3 |\Sigma_{(\rho,\phi,\theta)}|}} \exp\left(-\frac{1}{2}\delta^T \Sigma^{-1}_{(\rho,\phi,\theta)} \delta\right), \tag{8}$$

for $\delta = q - \mu_{(\rho,\phi,\theta)}$. $|\Sigma_{(\rho,\phi,\theta)}|$ and $\Sigma^{-1}_{(\rho,\phi,\theta)}$ denote, respectively, the determinant and the inverse of matrix $\Sigma_{(\rho,\phi,\theta)}$. Notice that the right side of (7) is a different constant value for each kernel.

Since $\mathcal{A}$ is a discrete domain, we start voting at the bin that includes $\mu_{(\rho,\phi,\theta)}$ and neighbor bins are reached in a flood-fill fashion, having the condition (7) to stop the flooding.

It is important to note that the density of points in the clusters varies with camera distance, while the integral of (8) is one due to the normalization axiom of probability. Therefore, kernel votes in each bin is composed by multiplying the evaluation of (8) to the $w_{\mathcal{C}}$ factor (9) computed for a given cluster $\mathcal{C}$ regarding the spatial coverage of its tree node and the number of samples. Similar to [1], but with appropriate adjustments for our clustering model, we define

$$w_{\mathcal{C}} = w_a \frac{\mathcal{R}_{\text{area}}}{\mathcal{I}_{\text{area}}} + w_d \frac{m}{n}, \tag{9}$$

where $\mathcal{I}_{\text{area}}$ and $\mathcal{R}_{\text{area}}$ are the areas (in pixels) of the image and of the rectangular region of the node, respectively. n is the number of samples in the whole image, and m is the number of samples in the cluster. The values of w_a and w_d are restricted to $w_a + w_d = 1$. In our experiments, we confirm the findings of Limberger and Oliveira [1] that better results are reached by using $w_a = 0.75$ and $w_d = 0.25$. Thus, we favor number of pixels (area) against number of samples.

3.5. Detecting peaks of votes

After the voting process, the peaks of votes in the accumulator map correspond to the detected planes. However, it is essential to apply a low-pass filter to consolidate local maxima [3] before searching for peaks of votes. In our experiments, we computed the convolution of the accumulator map and a six-connected filter with central weight of 0.2002 and neighbor weights of 0.1333. This filtering operation smooths the voting map, helping to consolidate adjacent peaks as single detected planes.

We apply a hill climbing strategy to detect peaks of votes in the smoothed accumulator map. For each kernel used in the voting procedure, we take the accumulator bin addressed by the parameters $\mu_{(\rho,\phi,\theta)}$ of the mean plane and check whether this bin corresponds to a local maximum. If the condition is true, the bin is marked as a detected plane. Otherwise, we take the neighbor bin having more votes than the current bin and repeat the process until we find a local maximum.

Once we have the parameters of each plane detected by hill climbing, the next step is to readjust the relevance of the planes with respect to the image. The relevance is given as function of the weighting factor $w_{\mathcal{C}}$ (9) of each cluster that has climbed to a given local maximum. The relevance of a given peak $\mathcal{P}$ is given by the summation of the weights associated to the clusters that contributed to the peak. We have observed that this simple strategy lead to better ordering of detected planes than the use of the number of votes in datasets of real scenes.

Given the parameter vector $(\rho, \phi, \theta)^T$ of a detected plane, its normal vector $\vec{n}$ is computed as:

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} \sin\phi\cos\theta \\ \sin\phi\sin\theta \\ \cos\phi \end{pmatrix}.$$

The normal vector $\vec{n}$ and the distance ρ define the normal equation of the plane supporting a subset of 3-D point entries.

3.6. Time complexity analysis

Our plane detection approach is comprised of three steps having linear asymptotic time complexity each. Therefore, the time complexity of the D-KHT is also $O(n)$. Without loss of generality, let's assume an input depth image with $d \times d$ pixels and $n = d^2$ point samples for $d \geq 2$ being a power of two. The D-KHT first computes a set of SATs and build an implicit quadtree to subdivide the image into clusters of approximately coplanar samples. The cost of computing each SAT is $O(n)$ (see [16]). By assuming the worst-case scenario, the quadtree would subdivide the samples into $n/2$ clusters (tree leafs) defined by 2×2 image regions. In that case, the height of the tree would be $\log_4 n$, leading to $\sum_{l=1}^{\log_4 n} 4^{l-1} = (n-1)/3$ nodes. At each node, we have to compute a 3×3 covariance matrix $\Sigma_{(x,y,z)}$ and its eigendecomposition. The SATs allow the computation of $\Sigma_{(x,y,z)}$ in $O(1)$ time. Decomposing $\Sigma_{(x,y,z)}$ is accomplished in $O(1)$ time, too. Therefore, the operations in the whole clustering procedure are performed in $O(n)$.

The voting step consists of applying first-order error propagation to compute the Gaussian distribution of the uncertain plane of each cluster and update the accumulator map using those distributions. The first operation is performed in time $O(1)$, per cluster. The number of accumulator bins updated by each cluster depends on the distribution of samples in the cluster, and on the resolution of the accumulator. Nevertheless, it is typically much smaller than n. Therefore, we can safely handle the number of updated bins as a constant value, leading to time $O(1)$, per cluster. In the worst case we have $n/4$ clusters. Hence, the asymptotic time for the voting step is $O(n)$.

Finally, filtering any given accumulator bin is performed in $O(1)$ time, for a total cost of $O(n)$ by having only to filter bins that received votes. Hill climbing is accomplished for each cluster, leading to a small amount of accesses to accumulator bins per climbing process (much smaller than n). Thus, the time complexity of the peak detection step is $O(n)$, and the overall time complexity of the D-KHT is $O(n + n + n) = O(n)$.

3.7. Discussion

Despite sharing the KHT pipeline [3] and using the same model of accumulator map [8], the D-KHT and the 3-D KHT have fundamental differences. First, they assume different types of input data and clustering strategies. The 3-D KHT expects unorganized point clouds as input and uses an octree to organize and to cluster the points. Thus, the clustering step of the 3-D KHT has an asymptotic cost of $O(n\log_8 n)$. The D-KHT uses an implicit quadtree and SATs to perform clustering in depth images with cost $O(n)$. In addition, in order to find local maxima in the accumulator map, the 3-D KHT sorts the accumulator bins by the number of votes in descending order and checks whether a given bin has more votes than its neighbors, leading to $O(n\log n)$ cost. Our approach detects peaks by an efficient gradient climbing strategy with cost $O(n)$. As will be seen in Section 4.1, the differences between the D-KHT and the 3-D KHT are key for the performance of our approach.

4. Experiments and results

We have compared the performance of the D-KHT against state-of-the-art plane detection techniques by using the reference implementation of the 3-D KHT provided by Limberger and Oliveira, the `OpenCV` implementation of SG, which is based on [9] and [10], and the reference implementation of the RANSAC proposed by Schnabel et al. [12] for shape detection. Both D-KHT and 3-D KHT implementations use `dlib` for computing eigendecomposition. However, while the 3-D KHT uses `OpenMP` to parallelize the clustering procedure, our implementation is strictly sequential. The `C++` codes were compiled with Visual Studio 2013 and the experiments were performed on an Intel Core i5-6200U 64-bits CPU with 2.3 GHz and 8Gb of RAM. We have converted

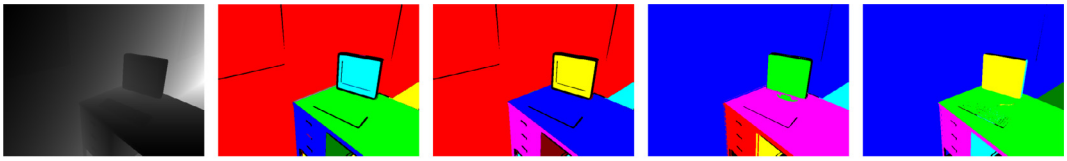


Fig. 4. Frame 2453 from the *Office 1* dataset compares the detection capability of several techniques in a noiseless scenario (best viewed in color): (a) Input depth image; and (b)–(e) planes detected in (a) by, respectively, D-KHT, 3-D KHT, SG, and RANSAC. In this example, all the techniques are able of retrieving the main planes of the scene but the D-KHT is ∼7 times faster than the 3-D KHT, and ∼7 and ∼65 times faster than SG and RANSAC, respectively.

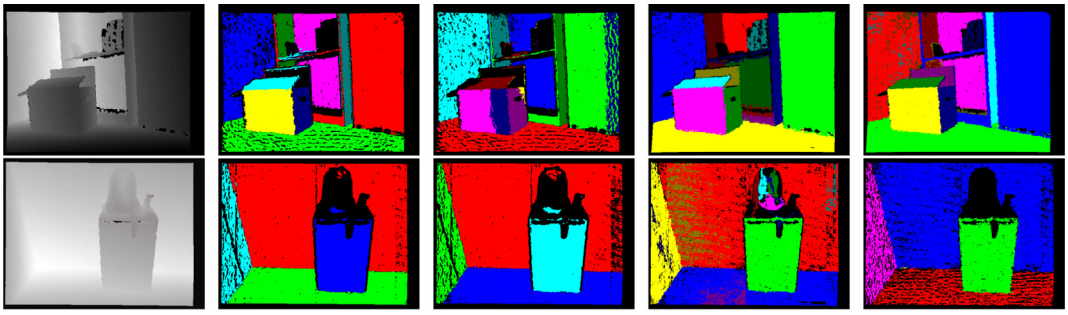


Fig. 5. The *Kinect Scene 3* dataset (top) and Frame 4161 from the *Copy Room* dataset (bottom) compare the detection capability of several techniques in real scenarios: (a) Input depth images; and (b)–(e) planes detected in (a) by, respectively, D-KHT, 3-D KHT, SG, and RANSAC. As expected, SG is more sensitive to noise and discontinuities than HT-based approaches and RANSAC. Notice the double-detection of supporting planes for the wall and the floor.

depth images into point clouds to use them as input for the 3-D KHT and RANSAC. All input depth images have 640×480 pixels.

We have used depth images from thirty six different datasets in our experiments. Figs. 1, 4, and 5 present the results produced for some of them. Supplementary Material A includes more results due to space restrictions. The detected planes were rendered by painting all entry points attending two conditions: they must be sufficiently close to the given plane, and the normal vectors estimated from the depth image must be sufficiently aligned to the normal to the plane. Black regions correspond to non-planar surfaces and noise. The synthetic datasets *Living Room 1* and the *Office 1*, with 2,870 and 2,690 frames each, were provided by Choi et al. [18]. The synthetic dataset *Occlusion Room* was built by us and includes one frame. We have used thirty three real datasets. The *Kinect Scenes* have one frame each. They were obtained with a Microsoft Kinect by Oehler et al. [19], and include ground truth information. Results of all the thirty scenes provided by Oehler et al. [19] can be found in Supplementary Materials A–D. Due to space restrictions, this section discusses only six *Kinect Scenes*. The *Copy Room* dataset was captured by Zhou and Koltun [20] with an Asus Xtion PRO LIVE camera. It is comprised of 5490 frames. We used a Structure Sensor to capture the *Cube* and the *Distance* datasets. We use the latter to analyze the stability of the algorithms in function of distance.

4.1. Computational performance

The processing times of the techniques are affected by the selected parameters. We choose parameter values that optimize the execution times of each individual algorithm, while trying to keep equivalent detection quality. We have set the same parameter values for all frames of a given dataset in order to verify the parameter dependency of the techniques upon different viewpoints of the same scene. Refer to Supplementary Material B for details on the parameters assumed in our experiments.

Table 1 summarizes the mean processing times (in milliseconds) involved in each step of the techniques (groups of columns) considering all frames of each dataset (rows). The *Total* columns present the mean total times (*i.e.,* summation of the previous columns). Supplementary Material C presents the mean running times per frame considering 50 executions. The *Distance* dataset was not included in this analysis.

The complexity of the scenes is related to the number and distribution of planar and non-planar surface patches in the environment, and the number of planes including one or more planar patches. For instance, the *Cube* dataset is much less complex than the *Copy Room* dataset considering all its frames. Table 1 shows that there is a relation between the mean frame rate and the complexity of the scene in the D-KHT. Notice that the proposed approach can achieve mean frame rate ranging from ∼55.2 (*Copy Room*) to ∼588.2 fps (*Cube*) for all real datasets. The 3-D KHT, on the other hand, was capable of achieve real-time performance (*i.e.,* more than 30 fps) only for the *Kinect Scenes 1, 5, 7*, and *11*, and for the *Cube* datasets. It achieved from ∼7.8 to ∼8.6 fps for synthetic datasets. SG could not achieve real-time performance, unless the depth and the normal maps have been provided as input. RANSAC was not capable of detect planes in real-time.

The use of SATs and an implicit quadtree makes the clustering step of the D-KHT from ∼2.4 to ∼36.3 times faster than the octree-based clustering strategy of the 3-D KHT. For synthetic scenes like the *Living Room 1* and the *Office 1, i.e.,* where all pixels have valid depth information, the voting procedure dominates the processing time of both techniques. For real datasets, the voting procedure barely dominates the time of the D-KHT, while the peak detection is the slower step of the 3-D KHT. Finally, there is a noticeable difference between the performances of the local maxima detection procedures of both techniques. The hill climbing strategy of the D-KHT is up to ∼93.2 times faster than the 3-D KHT in this step (*e.g., Kinect Scene 3* dataset). The normal vectors estimation had the longest execution time in SG and RANSAC for all the datasets, while the SG detection assuming the normal map as input achieved from ∼15.2 to ∼51.1 fps for synthetic scenes and ∼11.9–∼46.7 fps for real scenes. The model fitting step of RANSAC alone achieve from ∼1.3 to ∼4.9 fps. Devices such as Structure Sensor and Project Tango provide the normal map with

Table 1
Comparison of the performance of D-KHT, 3-D KHT, SG, and RANSAC. Mean times expressed in milliseconds.

Dataset	D-KHT				3-D KHT				SG			RANSAC		
	Clust.	Voting	Peak	Total	Clust.	Voting	Peak	Total	Normal	Growing	Total	Normal	Fitting	Total
S1 - *Living Room 1*	0.6	15.3	14.8	**30.7**	21.2	46.1	48.9	**116.2**	120.6	66.0	**186.6**	955.1	315.4	**1270.5**
S2 - *Office 1*	1.4	15.1	8.8	**25.3**	25.3	56.1	46.6	**128.0**	116.5	36.5	**153.0**	965.9	444.4	**1410.3**
S3 - *Occlusion Room*	1.9	4.5	5.5	**12.0**	20.0	51.6	49.4	**120.9**	105.9	19.6	**125.5**	1017.3	309.8	**1327.2**
R1 - *Kinect Scene 1*	1.3	1.2	1.1	**3.6**	14.3	2.0	8.4	**24.7**	110.2	83.7	**193.9**	966.0	414.8	**1380.8**
R2 - *Kinect Scene 3*	0.7	2.3	2.5	**5.5**	12.5	15.2	68.7	**96.5**	112.5	25.7	**138.2**	939.0	319.9	**1258.9**
R3 - *Kinect Scene 5*	0.7	4.7	5.5	**10.9**	13.8	3.4	14.8	**32.0**	114.3	24.1	**138.4**	934.9	373.6	**1308.5**
R4 - *Kinect Scene 7*	1.0	1.8	1.6	**4.4**	19.0	1.8	6.7	**27.5**	113.8	22.0	**135.8**	927.5	402.4	**1330.0**
R5 - *Kinect Scene 10*	1.1	4.2	2.9	**8.2**	13.9	2.8	18.4	**35.2**	115.2	69.2	**184.4**	924.5	438.8	**1363.3**
R6 - *Kinect Scene 11*	4.4	1.8	3.4	**9.6**	10.4	2.2	10.0	**22.6**	110.4	53.4	**163.8**	930.2	768.7	**1698.9**
R7 - *Copy Room*	3.1	10.5	4.6	**18.1**	12.2	23.3	40.7	**76.2**	93.8	25.7	**119.4**	753.3	332.9	**1086.2**
R8 - *Cube*	0.4	0.5	0.8	**1.7**	13.0	2.8	7.5	**23.3**	117.9	21.4	**139.3**	792.0	203.4	**995.4**

the depth images. Even tough, experiments show that the D-KHT is faster than the detection steps of these algorithms.

4.2. Quality of detections

We have compared the quality of detections by using the ground truth information manually produced by us for the *Living Room 1* (frame 294), the *Office 1* (frame 2453), and the *Occlusion Room* datasets, and the ground truth information accompanying all the *Kinect Scenes*. In contrast to synthetic datasets, real datasets are comprised of depth images with optical distortions, noise, as well as damaged and missing portions of depth data. So, we have produced noisy versions of the synthetic datasets by using the distortion model introduced by Teichman et al. [21]. Such a model incorporates disparity-based quantization, realistic high-frequency noise, and low-frequency distortion estimated on real depth sensors. We have applied the distortion model of a Microsoft Kinect in this paper.

Table 2 shows the amount of planes detected by each technique, and precision and recall measurements on points that are correctly assigned to their respective planes. The *Plane* columns present the number of existing planes detected (*DP*), the number of existing planes that were detected multiple times (*MD*), the number of missing planes (*MP*), and the number of spurious planes detected (*SP*). Thus, each scene has *DP+MP* planes. A careful inspection on those numbers reveals a known problem of SG and RANSAC: High *SP* values. For instance, *Kinect Scenes 1, 7, 10*, and *11* have many spurious planes. This problem is caused by the random nature of these approaches, which may prevent the selection of appropriate seed points or candidate models. On the other hand, HT-based approaches seems to miss more planes than other techniques. Nevertheless, the real-time performance of the D-KHT allows the detection of those planes in subsequent video frames.

According to Table 2, precision and recall favor D-KHT over 3-D KHT in virtually all cases. Also, adding noise to synthetic images ($S^{\dagger}$ datasets in Table 2) affects the *DP, MD, MP*, and *SP* values of D-KHT, 3-D KHT, and SG techniques in the same proportion, with RANSAC being slightly more noise-resilient.

Supplementary Material E presents additional experiments.

4.3. Distance analysis

We have used the *Distance* dataset to observe the quality of detection of three planar surfaces as the depth sensor gradually moves away from them, with 50 frames captured every half meter. The distance to the nearest surface (*N*) ranges from 0.5 (first position) to 3.5 m (last position), while the average (*A*) and farthest (*F*) surfaces are, respectively, 1.7 and 4.0 m apart from the sensor's first location. Supplementary Material F includes images and statistics regarding this experiment.

Table 2
Quality of detections. *S#* and *R#* denote synthetic and real datasets (see Table 1). The † symbol indicates noisy versions of synthetic datasets.

Dataset/Technique		Plane				Point	
		DP	MD	MP	SP	Precision	Recall
S1	D-KHT	4	0	4	2	0.98	0.96
	3-D KHT	4	0	4	0	0.72	0.97
	SG	4	1	4	2	0.74	0.94
	RANSAC	5	0	3	1	0.70	1.00
S1†	D-KHT	4	0	4	2	0.71	0.86
	3-D KHT	3	1	5	2	0.66	0.70
	SG	4	1	4	3	0.72	0.87
	RANSAC	4	1	4	4	0.71	0.80
S2	D-KHT	6	0	3	0	0.77	0.96
	3-D KHT	6	0	3	0	0.77	0.96
	SG	6	0	3	0	0.77	0.96
	RANSAC	6	0	3	2	0.76	0.98
S2†	D-KHT	5	0	4	0	0.73	0.83
	3-D KHT	3	0	6	0	0.78	0.86
	SG	5	0	4	0	0.75	0.84
	RANSAC	6	0	3	0	0.75	0.78
S3	D-KHT	13	0	4	0	0.85	0.95
	3-D KHT	13	0	4	0	0.96	0.88
	SG	14	4	3	3	0.92	0.85
	RANSAC	14	0	3	9	0.96	0.97
S3†	D-KHT	7	0	10	1	0.83	0.80
	3-D KHT	3	0	14	1	0.93	0.93
	SG	9	0	8	4	0.90	0.72
	RANSAC	7	0	10	6	0.90	0.80
R1	D-KHT	11	0	4	1	0.79	0.85
	3-D KHT	10	0	5	6	0.72	0.82
	SG	14	2	1	9	0.75	0.79
	RANSAC	12	2	3	7	0.72	0.62
R2	D-KHT	9	7	4	0	0.95	0.91
	3-D KHT	9	1	4	1	0.93	0.87
	SG	11	1	2	3	0.91	0.84
	RANSAC	11	5	2	0	0.95	0.86
R3	D-KHT	9	0	3	3	0.82	0.71
	3-D KHT	6	0	6	4	0.76	0.55
	SG	10	0	2	3	0.84	0.89
	RANSAC	11	0	1	3	0.84	0.77
R4	D-KHT	6	2	7	3	0.86	0.77
	3-D KHT	7	1	6	2	0.79	0.83
	SG	12	0	1	6	0.83	0.74
	RANSAC	10	0	3	12	0.79	0.76
R5	D-KHT	8	0	1	6	0.88	0.73
	3-D KHT	8	0	1	6	0.68	0.68
	SG	9	2	0	11	0.91	0.83
	RANSAC	9	3	0	12	0.87	0.87
R6	D-KHT	8	0	6	8	0.79	0.67
	3-D KHT	8	0	6	6	0.70	0.77
	SG	14	2	0	9	0.78	0.65
	RANSAC	11	3	3	7	0.80	0.77

Table 3
Quality of detections at different distances having *N* as reference.

Plane/Tech.		0.5 meter			1.0 meter			1.5 meters		
		Cos.	Pre.	Rec.	Cos.	Pre.	Rec.	Cos.	Pre.	Rec.
N	D-KHT	1.00	0.93	0.96	1.00	0.89	0.93	1.00	0.90	0.91
	3-D KHT	0.93	0.92	0.90	0.81	0.76	0.73	0.54	0.49	0.49
	SG	1.00	0.92	0.97	1.00	0.91	0.89	0.99	0.88	0.85
	RANSAC	1.00	0.94	0.99	1.00	0.89	0.98	0.69	0.54	0.67
A	D-KHT	1.00	1.00	0.84	1.00	1.00	0.73	0.96	0.96	0.58
	3-D KHT	0.99	1.00	0.70	0.98	0.98	0.55	1.00	0.99	0.60
	SG	1.00	0.99	0.50	0.94	0.93	0.52	0.96	0.93	0.60
	RANSAC	1.00	0.98	0.59	1.00	0.97	0.62	0.98	0.94	0.92
F	D-KHT	0.98	1.00	0.47	0.99	1.00	0.25	0.82	0.48	0.11
	3-D KHT	1.00	1.00	0.46	0.86	0.88	0.20	0.63	0.63	0.24
	SG	0.65	0.65	0.35	0.70	0.73	0.17	0.88	0.87	0.29
	RANSAC	0.98	0.96	0.81	1.00	0.96	0.32	0.98	0.95	0.83

Ground truth information such as pixel coverage and plane's orientation was produced by manual segmentation and calibration procedures. We have used three metrics to measure the stability of detections produced by D-KHT, 3-D KHT, SG, and RANSAC in each of the seven sensor's positions: The cosine similarity between the normal to the planes, precision, and recall on points that are correctly assigned to their surfaces.

This experiment shows that the quality of detections decreases as distances increases. This was expected because the depth sensor precision falls off with distance. Due to space restrictions, Table 3 only presents mean metric values considering frames captured at 0.5, 1.0 and 1.5 m from *N*. The metrics show that the D-KHT outperforms other approaches at short distance (first three positions). From 2.5 m, the D-KHT and the 3-D KHT stop detecting the surface *N* (the smallest one), and loose surface *F* in most frames when it is at least 7.0 m away. The lack of precision of HT-based techniques regarding distant noisy structures can be explained by the discrete nature of their parameter space. As expected, SG suffers from multiple detections of the same planes due to discontinuities. RANSAC was more resilient to noise at medium and long distances, being capable of detecting all the three planes of interest even in the most distant position. Unfortunately, RANSAC can only process a couple of frames per second.

The results presented in Tables 1–3 depict D-KHT's potential for use in short-range real-time applications.

4.4. Limitations

As any HT-based approach, the performance and robustness of the D-KHT are constrained to the discretization of the parameter space. In addition, parallel close planes may be retrieved as a single instance in noisy datasets. Furthermore, the presence of non-planar surfaces may lead to the detection of spurious planes. Fortunately, those planes often have lower importance, being rejected with the choice of a suppression threshold.

Not being applicable to point clouds may be a disadvantage of detection techniques tailored to depth images. Many robotic applications deal with registered point clouds obtained from different views. In those cases, the point cloud registration followed by the detection step considering all the data from different sources must be replaced by high-level feature detection in each source (*e.g.,* patch detection) followed by registration.

5. Conclusion and future work

We have presented an $O(n)$ HT-based approach for real-time plane detection in depth images, where n is the number of pixels in the input image. We use an implicit quadtree to identify clusters of approximately coplanar points in the 2.5-D projection of the scene. PCA and the desired minimum samples (valid depth pixels) in tree nodes guide the subdivision criteria of the quadtree. Thus, the maximum height of the tree is known *a priori*, making the computational cost of the detection procedure predictable. For each cluster, our approach casts votes for a reduced set of planes in the accumulator representing the parameter space of possible planes. Casted votes are weighted by trivariate-Gaussian distributions that models the uncertainty on the best-fitting plane of the node samples in each cluster. Peaks of votes are retrieved by an efficient hill climbing procedure. Our approach is deterministic, fast, and robust to the detection of spurious planes, even in the presence discontinuities.

We are currently investigating how to reduce the memory footprint of our technique by considering only the subset of planes that cross the camera's frustum. We also believe that there is a relation between N_ρ and s_t. Thus, we are conducting experiments in an attempt to set these two parameters as functions of a single value. As future work, we will investigate automatic ways to learn ideal parameter values from examples. We are exploring the proposed algorithm as part of a real-time solution for geometric reconstruction of indoor environments by stitching planar regions. In this application, the input depth images are provided on-the-fly by off-the-shelf depth sensors.

Acknowledgments

We thank Limberger et al. and Schnabel et al. for kindly providing the reference implementations of their techniques, and Souza for modeling the *Occlusion Room*. This work was sponsored by CNPq-Brazil grants 456.016/2014-7, 308.316/2014-2, and FAPERJ grants E-26/110.092/2014, E-26/202.832/2015. We thank the reviewers for their insightful suggestions.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.patrec.2017.12.027.

References

[1] F.A. Limberger, M.M. Oliveira, Real-time detection of planar regions in unorganized point clouds, Pattern Recognit. 48 (6) (2015) 2043–2053, doi:10.1016/j.patcog.2014.12.020.
[2] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15, doi:10.1145/361237.361242.
[3] L.A.F. Fernandes, M.M. Oliveira, Real-time line detection through an improved Hough transform voting scheme, Pattern Recognit. 41 (1) (2008) 299–314, doi:10.1016/j.patcog.2007.04.003.
[4] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic Hough transform, Pattern Recognit. 24 (4) (1991) 303–316, doi:10.1016/0031-3203(91)90073-E.
[5] A. Yla-Jaaski, N. Kiryati, Adaptive termination of voting in the probabilistic circular Hough transform, IEEE Trans. Pattern Anal. Mach. Intell. 16 (9) (1994) 911–915, doi:10.1109/34.310688.
[6] J. Matas, C. Galambos, J. Kittler, Progressive probabilistic Hough transform, in: Proc. of BMVC, 1998, pp. 26.1–26.10, doi:10.5244/C.12.26.
[7] L. Xu, E. Oja, P. Kultanen, A new curve detection method: randomized Hough transform (RHT), Pattern Recogn. Lett. 11 (5) (1990) 331–338, doi:10.1016/0167-8655(90)90042-Z.
[8] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, The 3D Hough transform for plane detection in point clouds: a review and a new accumulator design, 3D Research 2 (2) (2011) 1–13, doi:10.1007/3DRes.02(2011)3.
[9] J. Poppinga, N. Vaskevicius, A. Birk, K. Pathak, Fast plane detection and polygonalization in noisy 3D range images, in: Proc. of IROS, 2008, pp. 3378–3383, doi:10.1109/IROS.2008.4650729.
[10] J. Xiao, J. Zhang, J. Zhang, H. Zhang, H.P. Hildre, Fast plane detection for SLAM from noisy range images in both structured and unstructured environments, in: Proc. of ICMA, 2011, pp. 1768–1773, doi:10.1109/ICMA.2011.5986247.
[11] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395, doi:10.1145/358669.358692.
[12] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, Comput. Graph. Forum 26 (2) (2007) 214–226, doi:10.1111/j.1467-8659.2007.01016.x.

[13] C. Erdogan, M. Paluri, F. Dellaert, Planar segmentation of RGBD images using fast linear fitting and Markov chain Monte Carlo, in: Proc. of CRV, 2012, pp. 32–39, doi:10.1109/CRV.2012.12.
[14] H.J. Hemmat, A. Pourtaherian, E. Bondarev, et al., Fast planar segmentation of depth images, in: Proc. of SPIE/IS&T Electronic Imaging, 2015, pp. 93990I–193990I, doi:10.1117/12.2083340.
[15] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, 3D object recognition in cluttered scenes with local surface features: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 36 (11) (2014) 2270–2287, doi:10.1109/TPAMI.2014.2316828.
[16] F.C. Crow, Summed-area tables for texture mapping, SIGGRAPH Comput. Graph. 18 (3) (1984) 207–212, doi:10.1145/964965.808600.
[17] H. Samet, The quadtree and related hierarchical data structures, ACM Comput. Surv. 16 (2) (1984) 187–260, doi:10.1145/356924.356930.
[18] S. Choi, Q.-Y. Zhou, V. Koltun, Robust reconstruction of indoor scenes, in: Proc. of CVPR, 2015, pp. 5556–5565, doi:10.1109/CVPR.2015.7299195.
[19] B. Oehler, J. Stueckler, J. Welle, D. Schulz, S. Behnke, Efficient multi-resolution plane segmentation of 3D point clouds, in: Proc. of ICIRA, 2011, pp. 145–156, doi:10.1007/978-3-642-25489-5_15.
[20] Q.-Y. Zhou, V. Koltun, Dense scene reconstruction with points of interest, ACM Trans. Graph. 32 (4) (2013) 112:1–8, doi:10.1145/2461912.2461919.
[21] A. Teichman, S. Miller, S. Thrun, Unsupervised intrinsic calibration of depth sensors via SLAM, in: Proc. of Robotics: Science and Systems, 248, 2013, pp. p27.1–p27.8, doi:10.15607/RSS.2013.IX.027.