MVA manuscript No.
(will be inserted by the editor)

3D Hough Transform for Sphere Recognition on Point Clouds

A Systematic Study and a New Method Proposal

Marco Camurri · Roberto Vezzani · Rita Cucchiara

Received: date / Accepted: date

Abstract Three-dimensional object recognition on range data and 3D point clouds is becoming more important nowadays. Since many real objects have a shape that could be approximated by simple primitives, robust pattern recognition can be used to search for primitive models. For example, the Hough Transform is a well known technique which is largely adopted in 2D image space. In this paper, we systematically analyse different probabilistic/randomized Hough Transform algorithms for spherical object detection in dense point clouds. In particular, we study and compare four variants which are characterized by the number of points drawn together for surface computation into the parametric space and we formally discuss their models. We also propose a new method that combines the advantages of both single-point and multi-point approaches for a faster and more accurate detection. The methods are tested on synthetic and real datasets.

Keywords 3D Hough Transform · Sphere detection · Randomized HT · Probabilistic HT

1 Introduction

The high availability of low-cost range sensors has attracted the research community's attention towards object recognition tasks on 3D data [1]. Since 3D point clouds bring more information than 2D images, they allow to overcome some intrinsic difficulties, like the precise detection of the edges (*e.g.,* due to albedo) and the ambiguities caused by the projection of a 3D scene into a 2D pixel matrix.

The main limitations of 3D point clouds — when acquired with low cost range sensors — are usually related to their sparse distribution and the presence of noise.

Many applications could take advantage by extracting three-dimensional information from the environment. For instance, recognizing 3D geometric regular shapes, such as spheres, could be a basic step for machine vision tasks, HCI tasks, head detection and medical image processing [2]. Among the other applications, a fast and precise detection and 3D localization of players and balls could be very useful for physical rehabilitation or monitoring of sport performance. Common libraries designed to support the analysis of data coming from RGB-D sensors (*e.g.,* , Microsoft Kinect) already guarantee very precise segmentation and description of the players, but they lack a corresponding support for the detection of objects which the players interact with or play through. As many real objects can be approximated by simple primitives [3] — such as planes, cylinders and spheres — 3D object recognition can be handled by parametric shape detection. One of the most popular and powerful methods for detecting parametric shapes in 2D images is the Hough Transform (HT), first introduced in [4] for line detection. During the last decades many HT-based methods have been widely proposed for 2D detection of circles, ellipses and also arbitrary shapes [5], but only recently has the HT been applied to a 3D scene, mainly for plane detection [6,7,8,9]. The popularity of this method stems from the simplicity and generality of the conversion of features extracted from the image space into a set of votes collected in a parametric space, which is easier to analyze. Furthermore, HT is very robust to outliers, occlusions and deformations of the input data and it is highly parallelizable.

A great limitation for this method is that both the computational cost and the memory requirements increase expo-

Roberto Vezzani · Rita Cucchiara
University of Modena and Reggio Emilia - Italy
Tel.: +39-059-2056136, Fax: +39-059-2056129
E-mail: roberto.vezzani@unimore.it, rita.cucchiara@unimore.it

Marco Camurri
Department of Advanced Robotics, Istituto Italiano di Tecnologia,
via Morego, 30, 16163 Genova
E-mail: marco.camurri@iit.it

nentially with respect to dimensionality of the input space. Several approaches have been proposed to reduce this burden, which could make 3D shape detection unfeasible. Most of them focus on reducing the dimensionality by exploiting gradient information to split a high dimensional input space into several lower dimensional subsets and process them separately.

This paper focuses on the potential exploitation of HT methods for detecting spheres in 3D point clouds. For a sphere detection task, a *Standard HT* (SHT) application works on a 4-dimensional parameter space consisting of three parameters x_0, y_0, z_0 for the sphere center and a fourth r for the sphere radius. Each point votes for an hyper-cone and we need at least four intersecting hyper-cones to find the target. A different approach could be evaluating more than one point at the same time, in order to reduce the dimensionality of each surface accumulated into the parameter space. An extreme case is represented by a reformulation of *Randomized HT* (RHT) for spheres, which evaluates four points together and produces a 0-dimensional manifold (*i.e.,* a single point) on the parameter space. This makes each voting operation faster, but it requires a large number of votes, since the likelihood of picking a good quadruple is biquadratic against the inlier ratio (*i.e.,* number of inlier points over total number of points); thus, the number of required votes to have satisfactory results could be too high.

In this paper we present two main contributions: first, we provide a theoretical analysis of HT applied to sphere detection; in particular we make a comparison between different solutions which draw from one to four point subsets for each surface calculation and discuss the link between the solutions and noise that affects input data. Secondly we propose a combined method to find spherical objects in dense point clouds. The method involves two steps: a first pass scans the cloud with a single-point HT application to reduce search area and the amount of outliers. The second pass applies a four-point HT to find an accurate parameter vector. The method does not require any gradient or color information, can detect multiple spheres and is very robust to outliers and partial data.

The rest of the paper is organized as follows. In Section 2 we review the literature relevant to our contribution. In Section 3 we provide a theoretical analysis of the HT applied to 3D sphere detection and compare different HT-based variants. In Section 4 we describe our novel approach. Section 5 illustrates the experimental framework used to validate the performance of our implementations. The final section presents conclusive observations about our work.

2 Related work

A brief literature review related to our work is presented below. The material is listed in chronological order and it is divided into two groups: for shape detection on 2D and 3D data, respectively. An old but detailed survey can be found in [10].

2.1 Early development in 2D image processing

HT was first introduced in 1962 by Paul Hough [11] for curve detection in bubble chamber photographs [12, 13] and reached the image processing community's attention in 1969, thanks to the work of Rosenfeld [14] which studied the simplest and most used HT application of line finding in binary images. The classical form of HT has been formalized some years later by Duda and Hart [15] with the so-called "normal parametrization".

Several other parametrizations were proposed to detect straight lines as well as more complex shapes, like circles [15, 16] and ellipses [17]. To overcome the increasing number of parameters for such shapes — which means a higher dimensional and therefore more expensive parameter space — the most adopted approach was to decompose the image space into a few easier to analyze subsets. Such a decomposition usually exploits some geometric properties of the parametric shape and gradient information to divide the initial set of parameters into smaller subsets that are processed independently (*e.g.,* [17] exploits the fact that parallel lines tangent to an ellipse, which is described by five parameters, are equally distant from the center; thus, they first look for the center of the ellipse by projecting couples of edges into a 2D accumulator and then they fix the center of the ellipse and look for the other three parameters in a similar way; see also examples on the 3D domain in Section 2.2). Thus, these approaches are not general, since they rely on the considered type of parametric shape.

A different approach was proposed by Illingworth and Kittler [16]. Their implementation, called "Adaptive Hough Transform" (AHT), adopts an iterative coarse-to-fine search strategy for detecting lines and circles [18] in 2D and 3D parameter spaces. It uses a small accumulator array that is restricted around significant peaks at the end of each iteration to produce a smaller but finer-grained parameter space for the next one, until a desired approximation level is reached. A coarse-to-fine approach is quite straightforward to use and it has also been adopted for our work.

In 1990 Kiryati *et al.* [19] introduced the "Probabilistic Hough Transform" (PHT), which aims to reduce computational and memory costs of the HT by finding the minimum subset of input data that produces useful results. They demonstrated that a successful identification can be performed by randomly selecting only 2% of the original input data. Several other works used the same underlying idea. One of the most important was the Randomized Hough Transform (RHT), proposed by Xu *et al.* [20, 21] in 1989. This method was used for line detection and randomly selects pairs of points instead

of single ones from the image space to vote into a mono-dimensional array, efficiently implemented by a concatenated list. Using more than one point as an input constraint has the advantage to decrease the number of free variables, or, in other words, the number of projected points corresponding to each input element. On the other hand, the number of input elements (couples, triplets, sets of points) exponentially grows, and a some kind of input selection is mandatory. Kiryati *et al.* [22] compared probabilistic and randomized approaches for line detection by varying the input properties. Inspired by this work, we analyze the performances of different HT variants for the more complex case of sphere detection in 3D spaces.

The integrated use of two different transform-based methods has been explored on 2D images, with the Iterative Randomized Hough Transform (IRHT) approach [23]. This method, proposed by Lu and Tan, iteratively applies a Randomized HT to a 2D image for ellipse detection, starting from the assumption that a RHT with a small poll size usually identifies a great ellipse that contains the target (greater ellipses have in fact more consensus points than small ones, so they should receive more votes). At each iteration a new search starts inside the area identified by the great ellipse, until convergence is reached.

Princen *et al.* [24] proposed a formal definition of HT for 2D shapes. Here we adapt a similar model for 3D data.

2.2 3D HT development

Very few attempts at HT for 3D data have been presented in the literature. In 1990 Hsu and Huang [25] proposed the so-called "Partitioned Hough Transform" (PaHT), a partitioned method to reduce both time and space of ellipsoid detection by using the independent properties of ellipsoid parameters. This task is performed assuming that the input image — consisting in a 64×64 matrix range image — could contain several ellipsoids with the same rotational axis, orthogonal to the image plane. Starting from a 7D parametric shape (X_0, Y_0, Z_0 for center coordinates, A, B, C deformation coefficients, θ rotation angle around z-axis), the algorithm divides the parameter vector into two independent groups. In the first group only ellipses obtained by slicing the ellipsoid in sequences of ellipses stacked on the y-axis, so that only four parameters (X_0, Z_0, A, C_y with $C_y = C_y(C, y)$), are used. Similarly all ellipses stacked on the x-axis are detected and combined by projection to obtain a cross that locates the ellipsoid center on the xy-plane. Once those coordinates are detected, further steps are quite straightforward. It is demonstrated that PaHT can detect multiple ellipsoids, but cannot detect an ellipsoid if the noise is too large ($> \mathcal{N}(0, 0.01)$). This method also considers quite small input spaces and makes use of gradient information to detect the orientation of the ellipse border at each step. For larger spaces it is not computational feasible.

Taylor [26] proposed a methodology to reconstruct parametric regular surfaces, again from 64×64 range images, by decomposition of the parameter set into several low-dimensional (with 1 or 2 degrees of freedom) subsets. This result is achieved by using a multi-window parameter estimation technique, a k-tree parameter space searching a voting scheme and a conflict resolution process, which eliminates ambiguities related to parameter guessing hypotheses. The implementation has been proved to be very accurate for multiple sphere and cylinder detection, and also robust to typical noise produced by range sensors available at that time. However, no high outlier density situations were considered as our method does.

Spherical object detection is considered an important task for clinical applications, especially for replacement surgery. Van der Glas [2] proposed a technique to automatically determine the sub-voxel position and size of a sphere in unsegmented 3D images, to facilitate the glenohumeral joint substitution with a prosthesis. The method involves two steps. First, the sphere center is detected by using the gradient vector direction (by definition, every normal vector to a spherical surface points to its center) as input, and a three-dimensional parameter space for voting. Second, the sphere radius is determined by mapping gradient voxel values on a histogram and finding maxima. Anisotropy in the z direction and partial data are also taken into account. Even though this method is very accurate and suitable for its application, it is not general. It assumes in fact that there is always a sphere in the 3D image, and it also needs gradient information. Strong restrictions to the radius range are also applied, according to human biometrics. Similar considerations can be taken for the hierarchical approach proposed by Cao *et al.* [27], developed to detect the spherical parameters of the femoral head in CT images. The approach treats every CT slice with a HT-based circle detector and votes for the best (x_0, y_0) projected sphere center, thus halving the number of parameters. Secondly, the last center coordinate z_0 and radius R are detected with a second HT voting stage by exploiting geometric relationship between the sphere parameters.

Sphere detection was also used to automatically calibrate 3D optical sensors, with a sort of three-dimensional version of the commonly used pattern for 2D camera calibration. Ogundana *et al.* [28] proposed a method for sphere detection in point clouds which employs an optimized accumulator design; it provides compact data storage and efficient data access. The method has proved to be fast and accurate, but, due to the restriction applied, it is not general enough. It is assumed in fact that the number of spheres is fixed and known. It is also assumed that the radius is known, thus the real problem dimensionality is reduced to three (center coordinates only).

Kharbat *et al.* [29] proposed a sphere detection and tracking method using an enhanced HT technique and a H_∞ filter. Although the goal for this work is sphere detection, the real use of a HT-based method is for 2D circle detection, assuming that a projected sphere on a pixel matrix always assumes a circular shape. However, since the method seems robust within its application domain, a possible adaptation to circles on slices extracted from a point cloud will be explored as future work.

Recently, Abuzaina *et al.* [30] proposed a basic Hough Transform method used to detect partially occluded spheres from point clouds captured with the Microsoft Kinect sensor. Even if the authors highlighted the computational issues related to the algorithm, they handled the problem only by sub-sampling the input point cloud (by a factor of 20) and adapting the quantization steps accordingly, but no algorithmic solutions for data selection or space partitioning have been proposed.

A comparison of presented methods is summarized in Table 1, where we take into account: the source type and size (small range images, stack of 2D images, static images…), whether the method uses gradient information, whether the number of spheres is variable or not, whether the sphere radii are variable or not, and whether the method is robust to outliers and noise.

Table 1 Comparison between different sphere detection methods. From left to right: method authors, source type and size (Range Images, MR and CT stacks, 2D images, Point Clouds), whether the method uses of gradient information (**G**), whether it supports multiple sphere detection (**M**), whether it supports variable radius sphere detection (**R**), robustness level to outlier (**O**) and robustness level to noise (**N**)

Method	Source	G	M	R	O	N
Hsu and Huang [25]	4096 px RI	no	yes	yes	weak	weak
Taylor [26]	4096 px RI	no	yes	yes	weak	robust
Van der Glas [2]	MR stack	yes	no	yes	weak	robust
Cao *et al.* [27]	CT stack	yes	no	yes	weak	robust
Ogundana *et al.* [28]	> 0.3M PC	no	no	no	weak	robust
Kharbat *et al.* [29]	2D	no	no	no	weak	robust
Abuzaina *et al.* [30]	> 0.3M PC	no	no	yes	robust	robust
Our method	> 0.3M PC	no	yes	yes	robust	robust

3 Hough transform for 3D Spheres

A sphere in a 3D space is unambiguously defined by four parameters, which are generally identified by its three center coordinates and the radius length. Thus, a Standard HT application should map each 3D point of a spherical surface into a manifold embedded in a four-dimensional parameter space.

In the following we define four models for HT, namely H_1, H_2, H_3, H_4, that take as input data a single point as in the Standard HT or a multi-point set of 2, 3 or 4 points, respectively.

While the first model is simply a redefinition of the Standard HT method for spheres in 3D space — or a Probabilistic HT if we take a random subsample of the original input — the other three solutions derive from the Randomized HT, revised for highly parametric shapes. We will formally discuss the model properties with respect to challenging noisy conditions and the probability to have a desired accuracy with a random selection of the input multi-point sets.

3.1 Definitions and symbols

Let us denote with $X = (x_1, \dots, x_M)^{\mathrm{T}}$ a generic point in a M-dimensional feature space and by $\Omega = (\omega_1, \dots, \omega_P)^{\mathrm{T}}$ a point in a P-dimensional parameter space $\boldsymbol{\Omega}$. In this specific case, the feature space contains a 3D point cloud (*i.e.,* $M = 3$), while spheres in 3D space are described by the above mentioned four parameters ($P = 4$). For notation convenience X and Ω components will be indicated with $X = (x, y, z)^{\mathrm{T}}$ and $\Omega = (x_0, y_0, z_0, r)^{\mathrm{T}}$, respectively.

Let $\mathbf{X}$ be a specific point cloud, *i.e.,* a collection of N feature points X_j:

$$\mathbf{X} = \left\{X_j\right\}_{j=1,\dots,N} \qquad X_j = \left(x_j, y_j, z_j\right). \tag{1}$$

Adapting the formalism proposed by Princen *et al.* [24] to 3D images, the Hough Transformation is based on a parametric constraint $f(X, \Omega) = 0$, which can be defined for 3D spheres as in the following equation:

$$f(X, \Omega) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0. \tag{2}$$

Given a source point X_j, the set of points in the parameter space Ω which satisfy the constraint of Eq. 2 lies on a hypersurface $\boldsymbol{\Omega}_j$ (more precisely, a right spherical hyper-cone):

$$\boldsymbol{\Omega}_j = \left\{ \Omega \in \boldsymbol{\Omega} \;\middle|\; f(X_j, \Omega) = 0 \right\}. \tag{3}$$

From a practical point of view, $\boldsymbol{\Omega}_j$ contains all parameter sets which generate spheres passing through the point X_j. Given a set of points $\mathbf{X}$ belonging to a sphere, the intersection of the corresponding hyper-surfaces will provide the parameter vector Ω_0 of the sphere itself:

$$\Omega_0 = \bigcap_{j=1}^{N} \boldsymbol{\Omega}_j \tag{4}$$

The estimation of the parameter vector Ω_0 from a real point cloud is not feasible in an analytic and exact way [15]. The HT methods, instead, adopt a bounding and quantization

of the parameter space, which is treated as a discrete accumulator array H. Each array element corresponds to a portion of Ω (cell) defined by the quantization step $\boldsymbol{\Delta}$.

Let $\hat{\boldsymbol{\Omega}}$ be the quantized parameter space. Let $C_{\hat{\Omega}} \subset \boldsymbol{\Omega}$ be the region of $\boldsymbol{\Omega}$ quantized to $\hat{\Omega}$. So we can define the function $p(X, \hat{\Omega})$ as follows:

$$p(X, \hat{\Omega}) = \begin{cases} 1, & \exists\, \Omega \in C_{\hat{\Omega}} \colon f(X, \Omega) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Equation 5 defines a relationship between feature points and the discretized parameter space (*i.e.,* the elements of the accumulator array H). It returns a non-zero value only if there is at least one point Ω in the continuous parameter space that exactly satisfies the shape constraint f (*i.e.,* it generates a sphere passing through X); Ω should also be contained within the discretized region selected by $C_{\hat{\Omega}}$. Thus, the value of the $\hat{\Omega}$-th cell of the accumulator array H_1 is obtained from a point cloud $\mathbf{X}$ as follows:

$$H_1^{\hat{\Omega}}(\mathbf{X}) = \sum_{j=1}^{N} p(X_j, \hat{\Omega}) \,, \tag{6}$$

where the subscript of H_1 indicates that only one feature point is drawn from the input data for each hyper-surface computation in parameter space.

Every non-zero term in Eq. 6 provides an additional hint to the parameter estimation, and it is also referred as a "voting" operation. When the voting stage is completed, a peak detection step analyzes the accumulator array and produces the vector of estimated parameters $\Omega_0 \in \hat{\boldsymbol{\Omega}}$ for the sought shape by calculating $\arg\max_{\hat{\Omega} \in \hat{\boldsymbol{\Omega}}} H_1^{\hat{\Omega}}$.

3.2 Increasing input constraint size

Given a subset of n input points $\Xi = \{X_i\}_{i=1,\dots,n}$, the constraint p expressed in Eq. 5 can be rewritten as follows:

$$p(\Xi, \hat{\Omega}) = \begin{cases} 1, & \exists\, \Omega \in C_{\hat{\Omega}} \colon f(X_i, \Omega) = 0 \;\; \forall X_i \in \Xi \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Function p in Eq. 7 returns a non-zero value only if there exists a parameter vector Ω inside the cell $C_{\hat{\Omega}}$ which satisfies the relation f for all the points X_i in the cloud Ξ.

By varying the subset cardinality $n = |\Xi|$, different variants of the Hough Transform can be defined instead of Eq. 6. In particular, for input sets of size $n = 2$ we have:

$$H_2^{\hat{\Omega}}(\mathbf{X}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} p(\{X_i, X_j\}, \hat{\Omega}) \tag{8}$$

In this case, given two points $X_1 = (x_1, y_1, z_1)$, $X_2 = (x_2, y_2, z_2)$ randomly drawn from all combinations expressed in Eq. 8, and their corresponding midpoint $X_m = (x_m, y_m, z_m)$, every sphere of radius $r < r_{\max}$ must have a center restricted to lie on a circle with center X_m, radius $d = \sqrt{r^2 - \|X_1 - X_m\|^2}$ and axis passing trough X_1 and X_2 (see Fig. 1a, where one example sphere with fixed radius and its center locus is depicted). As the center coordinates cannot vary freely, a more convenient parametrization could be:

$$\begin{cases} x = x_m + d \cdot u_1 \cos(\theta) + d \cdot v_1 \sin(\theta) \\ y = y_m + d \cdot u_2 \cos(\theta) + d \cdot v_2 \sin(\theta) \\ z = z_m + d \cdot u_3 \cos(\theta) + d \cdot v_3 \sin(\theta) \end{cases} \tag{9}$$

where $\theta \in [-\pi, +\pi]$, $d \in (0, d_{\max}]$ and $U = (u_1, u_2, u_3)$, $V = (v_1, v_2, v_3)$ are two generating vectors for the plane that contains the circle. With such a parametrization, the voting stage is reduced to evaluating every valid (θ, d) couple, instead of every triple (x_0, y_0, z_0). Using this multi-point model we start with $n = 2$ thus we use two nested loops instead of three.

For subsets Ξ composed by three points, the H accumulators are computed as:

$$H_3^{\hat{\Omega}}(\mathbf{X}) = \sum_{i=1}^{N-2} \sum_{j=i+1}^{N-1} \sum_{k=j+1}^{N} p(\{X_i, X_j, X_k\}, \hat{\Omega}) \tag{10}$$

In this case, for a three-points subset $\{X_1, X_2, X_3\}$, every sphere center must lie on a straight line, passing through the subset circumcenter $C = (x_c, y_c, z_c)$ and orthogonal to the plane π identified by the subset. A proper parametrization could be:

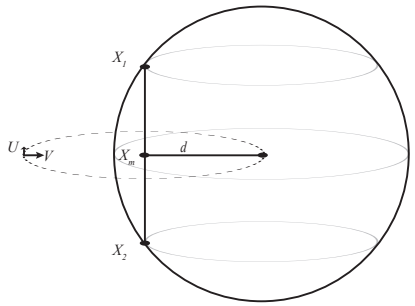
$$\begin{cases} x = x_c + n_1 \cdot t \\ y = y_c + n_2 \cdot t \\ z = z_c + n_3 \cdot t \end{cases} \tag{11}$$

where $\hat{\mathbf{n}} = (n_1, n_2, n_3)$ is the normal vector of plane π (see Fig. 1b). As the computation of votes is run across the only free parameter t, the complexity is further reduced by one dimension.

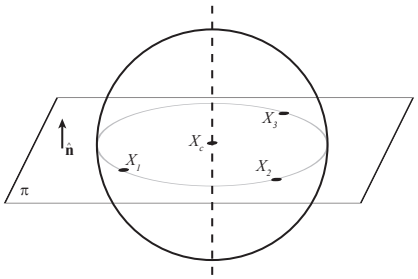
Finally, for 4-element sets, the accumulator is computed as:

$$H_4^{\hat{\Omega}}(\mathbf{X}) = \sum_{i=1}^{N-3} \sum_{j=i+1}^{N-2} \sum_{k=j+1}^{N-1} \sum_{l=k+1}^{N} p(\{X_i, X_j, X_k, X_l\}, \hat{\Omega}) \tag{12}$$

Equation 12 shows an extreme case, where the input subsets Ξ are transformed to a single point in the parameter space. HT corresponds to a RHT application to sphere detection.



(a) Sphere center locus with 2 point constraint



(b) Sphere center locus with 3 point constraint

Fig. 1 Sphere center loci with different point constraints

3.3 Implementation issues

The four HT algorithm variants described above — in the following called H_1, H_2, H_3, H_4 — present several implementation difficulties, both related to complexity and parametrization variables of the stored hyper-surfaces. The following paragraphs deal with these main issues.

Parametrization To correctly implement the HT variants expressed in Eq. 8 and 10 a different parametrization for computing and for storing hyper-surfaces must be used. This introduces non-trivial issues, since distortions, approximations and quantization value conversion could negatively affect the voting stage (*e.g.,* given a discrete (θ, d) couple, it is not assured that corresponding accumulator cell C_{Ω}, will receive at most one vote). However, a conversion from "voting space" to parameter space could introduce additional computational costs. For these reasons, H_2 and H_3 implementations will not be discussed in detail. However, we tested them with a simplified implementation, but they showed poor results if compared with the extreme cases of H_1 and H_4.

Computational complexity Most of the computational effort is made in the voting stage (*i.e.,* number of voting operations). Generalizing Eq. 6, 8, 10 and 12, the computational complexity of HT for a p-dimensional shape detection can be expressed as:

$$C_c = \binom{N}{n} \cdot \prod_{i=1}^{p-n} S_i = O\left(N^n \cdot \overline{S}^{p-n}\right) \tag{13}$$

where S_i represents the number of quantized values that run along the i-th parameter and $\overline{S}$ is the mean value of S_i. Equation 13 shows that high values of n are inapplicable when the cloud is dense (*i.e., N* is high), since savings brought by second factor are rapidly overtaken by the first one. To overcome this problem, the idea of using a random selection of the input as done in [20] could be applied; this would limit the number of subsets Xi used for the voting stage, but a criterion to approximately estimate a suitable size of such a sampling of the input has to be defined. Section 3.4 will introduce the problem of noise and outliers, which is related to a correct choice.

Memory consumption As the accumulator is generally implemented as a finite multidimensional array, the parameter space must be bounded along every dimension. Let us denote with M_i (resp. m_i) a generic upper (resp. lower) limit value for the i-th parameter. The most general choice for center coordinates values could be using the maximum (resp. minimum) coordinates of the point cloud $\boldsymbol{X}$ and using the minimum extended value of $\boldsymbol{X}$ bounding box as the radius (so it is assumed that at least a sphere octant is contained whitin the point cloud).

Let Δ_i be the quantization value used to divide the parameter space along the i-th dimension. Thus, each cell C of partitioned space $\boldsymbol{\Omega}$ has size $\prod_{i=1}^{4} \Delta_i$ and the entire accumulator contains a number Γ of cells equal to:

$$\Gamma = \prod_{i=1}^{4} \left\lceil \frac{|M_i - m_i|}{\Delta_i} \right\rceil \tag{14}$$

When sparse point clouds are analysed (*i.e.,* points are very distant from each other), the difference $|M_i - m_i|$ forces to use bigger Δ_i values to avoid memory overload and to maintain the value Γ within an affordable range.

This could lead to excessive approximations or, in the worst cases, to a wrong estimation.

3.4 Noise and outliers

The correct estimation of a parameter vector Ω_0 from a point cloud $\boldsymbol{X}$ requires that all the analysed elements belong to the sphere. In real cases this is not true — even if a real sphere is acquired — mainly due to the presence of noise and outliers. Intrinsic errors in the acquisition phase, deformations and numerical issues affecting feature points, introduce a sort of additive random noise $\Delta X = (\Delta x, \Delta y, \Delta z)$ so, instead of the real point location $X = (x_r, y_r, z_r)^{\mathrm{T}}$, the input cloud contains its noisy version $X + \Delta X = (x_r + \Delta x, y_r + \Delta y, z_r + \Delta z)^{\mathrm{T}} = (x, y, z)^{\mathrm{T}}$.

In addition to noise, the input cloud usually includes outlier features, which belong to non-target objects — even spherical — or to the background. Those points negatively

influence the Hough voting stage for a target sphere, and represent the most relevant issue for sphere detection in a real context.

For each target shape the entire image space can be arbitrarily partitioned into two subsets, one containing inlier points and the other outlier points, respectively. Such distinction is necessary to discuss the model, as inlier rate dramatically influences the performance of the HT variants proposed in Section 3.2.

3.5 Input selection by varying of inlier rate

The four HT variants introduced in Sections 3.1 and 3.2 are defined in a non-probabilistic fashion, but as mentioned above, for a real implementation a proper selection of subsets Ξ is needed. In this section we analyze how the size of drawn subsets influences the probability to have a successful detection if we use a subset of the cloud points as in the Probabilistic HT model. For simplicity we will continue to use the $H_n(\Omega)$ notation even with probabilistic versions of HT.

As stated in the last section, input sets can be viewed as a disjointed union of the inlier points set I and the outlier points set O, with $N = |I| + |O| = N_{\text{in}} + N_{\text{out}}$. Let us consider the probability $P(\{X_i\}_n \subset I)$ to find a correct n-subset $\{X_i\}_n$ *i.e.,* a subset whose points entirely belongs to the model and contribute to increase the correct peak into the accumulator space. For a sufficiently large value of N, P can be approximated as:

$$P(\{X_i\}_n \subset I) \approx \left(\frac{N_{\text{in}}}{N}\right)^n \tag{15}$$

Let us now consider a point cloud with a certain rate N_{in}/N. Let t denote the number of correct votes, v a desired number of correct votes and s the number of randomly selected points (resp. couples, triples, ...) actually taken. Here we consider v as the minimum number of votes necessary to achieve a correct detection. For $H_1(\Omega)$ at least four hypercones must intersect into parameter space, so that $v = 4$; for $H_2(\Omega)$ we have $v = 3$; for $H_3(\Omega)$ we have $v = 2$ and, finally, using $H_4(\Omega)$, we have $v = 1$.

We aim to estimate the minimum number of single points (for H_1) or sample subsets (for H_2, H_3, H_4) that must be taken to have a number of correct votes greater than v. In such a manner we estimate in which condition a detection can be feasible. The probability to have a certain success rate $t > v$ can be calculated as:

$$\begin{aligned} P(t > v) &= \\ &= 1 - P_f(t \le v) = 1 - \sum_{k=0}^{v} \binom{s}{k} \left(\frac{N_{\text{in}}}{N}\right)^{nk} \left(1 - \left(\frac{N_{\text{in}}}{N}\right)^n\right)^{s-k} \end{aligned} \tag{16}$$

We now want to compare single-point and four-point algorithms. By adapting the approach taken in [22] to our case, a proper value of s must be taken so that $s_{\text{H1}} N_{\text{in}}/N \gg 1$ for H_1 application and $s_{\text{H4}} \cdot (N_{\text{in}}/N)^4 \gg 1$ for H_4 application. Given a certain probability of failure P_f, we also have that $s_{\text{H1}} \approx s_{\text{H4}} \cdot (N_{\text{in}}/N)^3$.

Furthermore, below a certain N_{in}/N rate threshold, the use of $H_4(\Omega)$ becomes computationally disadvantageous compared with $H_1(\Omega)$. In normal operating conditions, the threshold T can be calculated as (for more details, see Appendix A):

$$T = \sqrt[3]{\frac{c_{\text{H4}}}{\overline{S}^3 \cdot c_{\text{H1}}}} \tag{17}$$

where c_{H4}, c_{H1} are the computational costs of a single vote operation of $H_4(\Omega)$ and $H_1(\Omega)$, respectively. As can be seen, T is severely influenced by quantization choice, which in turn drives accuracy performance.

4 Combined Multi-Point Hough Transform

Let us consider a typical point cloud with real 3D data where the inlier rate is low (*e.g.,* $N_{\text{in}}/N \approx 0.1$), so that a H_1 application should be preferred, according to the criterion expressed in Section 3.5.

Since H_4 execution time depends uniquely on $(N_{\text{in}}/N)^4$, even a slight increment of inlier rate can dramatically improve its performance. In contrast, H_1 time performance is less sensitive to inlier rate changes ($s_{\text{H1}} \propto N_{\text{in}}/N$) and also depends on the quantization level, which is arbitrary. These characteristics could be combined to obtain better time and accuracy performance.

We propose a novel method for sphere detection suitable in noisy point clouds affected by low inlier rate. Our method is called Combined Multi-Point Hough Transform (CMPHT) and involves two steps: first, a H_1 is applied to find an approximation of the sphere location. Secondly, the parameter detection refinement is achieved by H_4.

4.1 Algorithm description

Referring to Algorithm 1, a sufficiently small four dimensional accumulator is initialized, so that H_1 is faster than H_4.

Algorithm 1 CMPHT

Require: $\mathbf{X} = \left\{X_j\right\}_{j=1,\dots,N}$ ▷ Input point cloud

function DETECT($\mathbf{X}$)
  repeat
    $\tilde{\Omega}_0 \leftarrow computeH1\,(\mathbf{X})$ ▷ Coarse detection using H_1
    Compute the ROI of $\tilde{\Omega}_0$
    $\Omega_0 \leftarrow computeH4\,(\mathbf{X}, ROI)$ ▷ Refinement using H_4
    $d \leftarrow density\,(\mathbf{X}, \Omega_0)$ ▷ Density estimation
    if $d \geq d_{th}$ **then**
      $\mathbf{\Omega_0} += \Omega_0$ ▷ Add Ω_0 to the output set
      $\mathbf{X} -= \Omega_0$ ▷ Remove the sphere from the cloud
    end if
  until $d \leq d_{th}$
  return $\mathbf{\Omega_0}$ ▷ Set of sphere parameters
end function

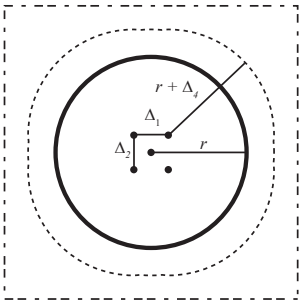


Fig. 2 Top view of search area box calculation (double dashed line).

Initially, H_1 is applied to the cloud $\mathbf{X}$ over the accumulator A and a set of approximated parameters $\tilde{\Omega}_0$ are produced. The H_1 computation is performed using a subset of $\mathbf{X}$, whose size depends on the estimated inlier amount of the cloud and is chosen according to the criterion expressed in Section 3.5.

Assuming that the quantization value is chosen properly (*i.e.,* the quantization step is adequately smaller than the target radius), those parameters identify a sector (*i.e.,* a Region of Interest) of the point cloud that should contain the target sphere. A proper sector size should be chosen by evaluating the amount of desired outlier removal, the accuracy of H_1 application, and point cloud density.

The Region of Interest can be calculated as follows: given an intermediate estimated parameter vector $\tilde{\Omega}_0 = (\tilde{x}_0, \tilde{y}_0, \tilde{z}_0, \tilde{r})$ (Fig. 2, solid line), the maximum error made on the i-th parameter should be $\Delta_i/2$, so the real sphere should be located inside a solid composed by morphologic dilatation of a cuboid with edges Δ_i using a sphere of radius $r + \Delta_4$ as structuring element (Fig. 2, dashed line). To simplify the model and compensate for possible errors, a box of center (x_0, y_0, z_0) and volume $(2(r + \Delta_4) + 2\max_i \Delta_i)^3$ is used.

Once the search area is restricted, a H_4 with a larger accumulator (*i.e.,* an accumulator with a more sensitive quantization interval) and a poll size proportional to outlier exclusion amount is applied; for this last step we opted for H_4 instead of a least square method because it is more robust to outliers (we expect that a typical ROI contains a non negletable set of points from objects close to the target, *e.g.,* the floor where a ball lies).

The new estimated parameters obtained during the previous step must be validated. For an almost spherical object with radius r, corresponding points should be located only between two concentric spheres with slightly different radii $r + \epsilon$, $r - \epsilon$. A volume difference between these two spheres can be used to compute a discretization of object surface area:

$$A_{\text{sphere}} = \frac{\frac{4}{3}\pi\,(r+\epsilon)^3 - \frac{4}{3}\pi\,(r-\epsilon)^3}{2\epsilon} \tag{18}$$

By counting the number of points belonging to the surface defined in Eq. 18, and by dividing it to the surface area, a measure of point surface density d is defined. This measure expresses the number of points per surface unit on a sphere of radius r. Considering the radius variability, we can define $\epsilon = c \cdot r\,,\ c \in [0, 1]$ and the density measure becomes:

$$d(c) = \frac{3 \cdot N_{\text{in}}}{4\,\pi\,r^2\,(c^2+3)} \tag{19}$$

The surface point density value is expected to be much higher for a correct identification than an erroneous one. Thus, it was used as a good measure to identify a new sphere. A correct threshold value for surface point density could be inferred using device information (*e.g.,* average number of points produced, maximum range covered), a ground truth or by an early input data analysis.

If the density of the detected sphere is above the threshold d_{th}, the new parameter vector found is stored and the corresponding points are excluded from the original point cloud. Otherwise no more spheres are detected and the algorithm stops.

4.2 Execution time and memory space usage

Given a cloud with a known N_{in}/N, H_4 requires that poll size must be proportional to $(N_{\text{in}}/N)^4$ to achieve a small failure rate. Denoting the proportionality constant with b, the execution time of a pure H_4 is simply $c_{\text{H4}} \cdot b/(N_{\text{in}}/N)^4$.

In contrast, a pure H_1 execution time is proportional to (N_{in}/N) and $\overline{S}^3$, where $\overline{S}$ is the mean accumulator size for a dimension and depends on Δ_i. Thus the execution time for H_1 is $c_{\text{H1}} \cdot \overline{S}^3 \cdot c/(N_{\text{in}}/N)$ (where c is the proportionality constant for H_1).

Our combined method joins H_1 and H_4, so its execution time performs better when:

$$c_{\text{H1}} \cdot \overline{S}_*^3 \cdot c/(N_{\text{in}}/N) + c_{\text{H4}} \cdot b/(N_{\text{in}}/N_{\text{ROI}})^4 < c_{\text{H4}} \cdot b/(N_{\text{in}}/N)^4$$

(20)

i.e., when the sum of execution time of a H_1 with a coarser accumulator dimension $\overline{S}_*$ and a H_4 with a reduced cloud N_{ROI} is smaller than a pure H_4 application with the whole original cloud. Since $\overline{S}$ is arbitrary and H_4 poll size depends on a fourth power of inlier rate, a time saving can be usually achieved when clouds contain many outliers.

Memory usage is the same for H_4, since the accumulator used during the H_1 step is small, while during the H_4 pass the same accumulator of a pure H_4 is used. However, since the ROI is smaller than the whole cloud, the quantization of CMPHT is finer, so with the same memory occupation a more accurate estimation is achieved.

More sophisticated methods to reduce memory usage could be used (*e.g.,* sparse n-dimensional arrays), but they were not investigated here since they are outside the scope of our work.

5 Experimental results

5.1 Datasets and metrics

We created both synthetic and real datasets for our tests. Seven synthetic datasets, named from "D1" to "D7" (Fig. 3), have been designed to simulate different types of error that can occur during the acquisition of a point cloud in a real context. Datasets from D1 to D6 consists of 100 point clouds each. Each point cloud is composed by 4096 points sampled from a basic perfect sphere which are modified to reflect a specific error type. Center coordinates and radius of the basic sphere of each sample vary randomly in the intervals $(-10, 10)$ and $(0, 10)$, respectively. The dataset D7, also composed of 100 point clouds, is designed to simulate a more realistic scenario and includes partial spheres and other non-spherical objects in the scene. Even for D7, the point clouds differ from each other for random position and dimension of the objects, with intervals that prevent them from interpenetrating each other or lying outside the squared floor (see Fig. 3g). The real dataset is populated by various game ball configurations acquired with a Kinect sensor (Fig. 4)[1].

Dataset D1: it consists of basic sphere sample points with Gaussian noise added to every coordinate; it is used to simulate the presence of noise caused by a bad acquisition of a smooth surface or by irregularities in the surface itself.

Dataset D2: it is created by adding Gaussian noise with $\sigma = 0.1$ to the y and z components of the basic sphere sample points, and by subtracting from every x component (with $x < x_0$) Gaussian noise with $\sigma = r/2$. It was used to simulate point scattering within a single direction. This is usually caused by erroneous reconstruction by structured light or laser range devices.

Dataset D3: it is created by modifying every point sample p_i of the perfect sphere with a power law so that the corresponding dataset point is:

$$d_i = \Re\left[(x_i - p_i)^{(1+c)} + x_i\right]$$

where x_i is the component of p_i on the x-axis, where distortion is applied, c is a uniform random variable taken from the interval $]0, 1[$ and $\Re[z]$ indicates the real part of a number z; it is used to simulate a non-linear distortion along a single axis (usually optical axis).

Dataset D4: it is obtained by applying a homothetic transformation to every sample point of a perfect sphere. Coefficients are chosen randomly from the interval $(1, 5/4)$. It is used to simulate a linear distortion among all axes, which usually occurs due to acquisitions with a non-perfectly square pixel sensor.

Dataset D5: it is obtained by adding Gaussian noise with $\sigma = r/2$ to every coordinate of a certain percentage of the basic perfect sphere sample points. It is used to evaluate system robustness to uncorrelated outlier points.

Dataset D6: it is obtained by substitution of a certain percentage of a perfect sphere sample points with points of another sphere. It is used to evaluate system robustness to correlated outliers. Two spheres appear in each instance of the dataset, but only one can be considered as a target sphere by keeping the point percentage of the other below 50%.

Dataset D7: it is obtained by composition of different shapes to reproduce a realistic scenario with multiple partial spheres and other non-spherical objects. In particular the dataset contains two half spheres that lie on a square plane together with a cube and a rectangular box. All the data were added with uniformly distributed noise (< 1% of the radius); the maximum inlier rate for each sphere is around 6 % (Fig. 3g)

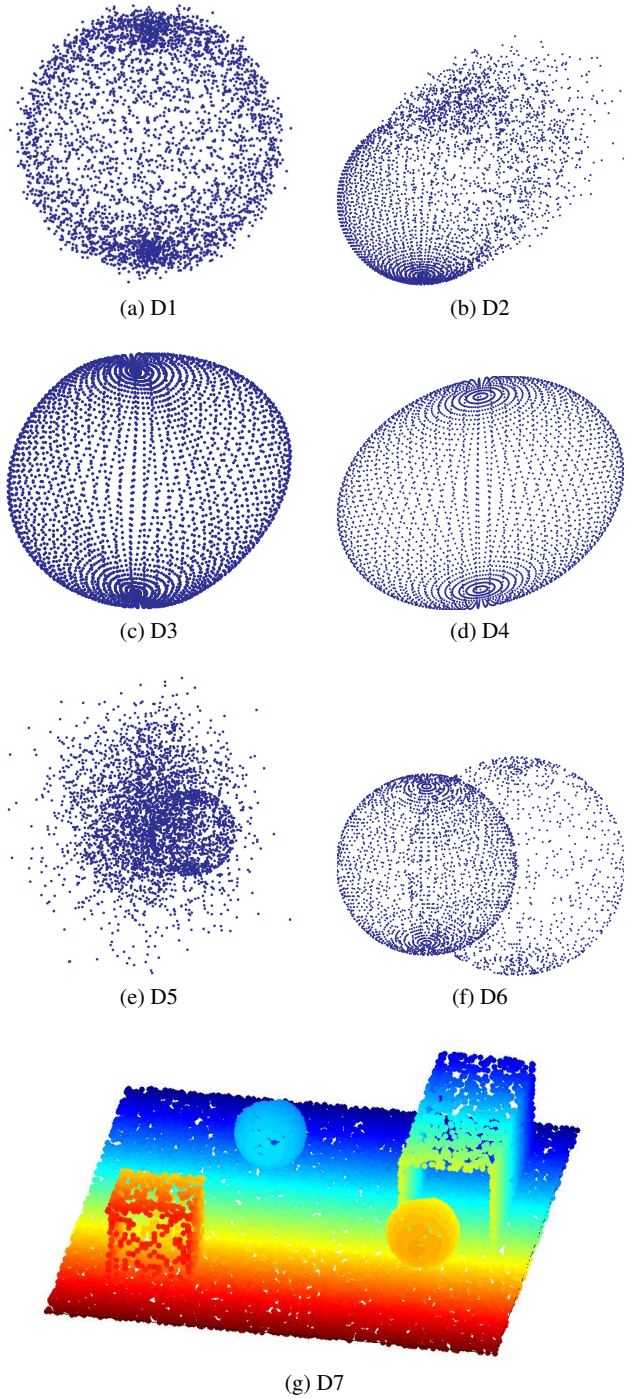
Real Dataset: a dataset for a real application was created by recording some spherical objects (game balls) using the Microsoft Kinect sensor (Fig. 4), which has been demonstrated to have a good accuracy for indoor 3D range data acquisition within 5 m [31,32]. The real dataset consists of 20 point clouds, both multi-target and single target.

5.2 Tests of HT variants on synthetic point clouds

In this section we compare the performance of the four HT variants discussed in Section 3.3 against different simulated conditions. The goal of these tests is to understand how each type of error alone affects the overall performance of the

[1] The datasets are available at the following url: `http://imagelab.ing.unimore.it/go/hough`

Fig. 4 Real dataset



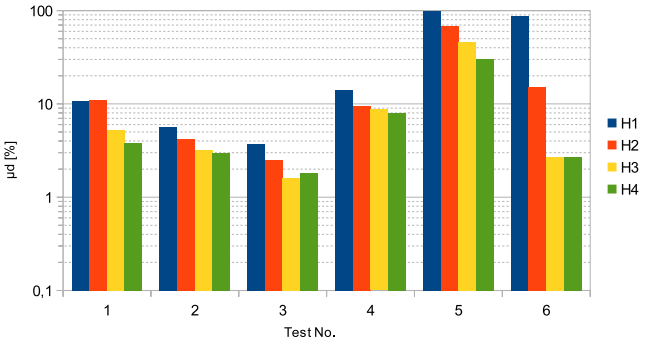
(a) D1 (b) D2

(c) D3 (d) D4

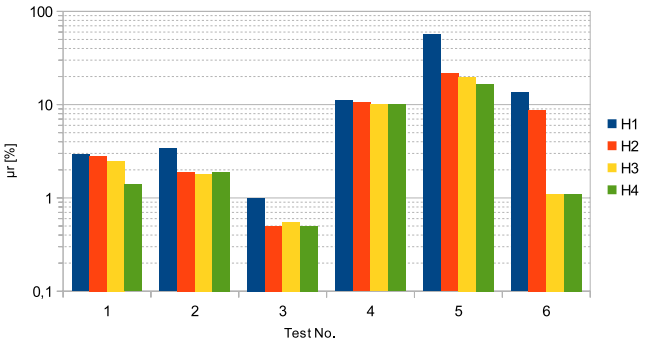
(e) D5 (f) D6

(g) D7

Fig. 3 Synthetic dataset examples. To make D7 more clear, the points are coloured according to depth and the cloud is downsampled.



(a) Mean distance error μ_d for D1 to D6 tests



(b) Mean radius error μ_r for D1 to D6 tests

Fig. 5 Mean percentage errors (logarithmic scale)

variants. Results are shown in Fig. 5, where, for each dataset and each variant, we show: the mean distance between the estimated center and the real center, normalized by the real radius length, μ_d (Fig. 5a); the mean estimated radius length normalized by the real radius length μ_r (Fig. 5b). These measures were obtained by running each variant inside three nested loops: in the first loop we run over each sample of the dataset, in the second we execute each variant 10 times on the same sample and in the third we progressively increase the input size by a constant, until a time limit is hit (5 s in this case). Time is reported just as an indication, since the code is not optimized.

For simplicity we did not change the parametrization of $H_2(\Omega)$ and $H_3(\Omega)$ between hyper-surface calculation and storage: we used (x_0, y_0) and x_0 as variables, instead of (θ, ρ) and t (see Section 3.2).

Noisy sphere test To analyze the behavior of the four variants with noisy data, we used several instances of D1 noisy

spheres with different input sizes. Results confirm that noise modestly influences accuracy performance of the algorithms with high values of n (Fig. 5, first column). The H_4 algorithm exhibits, as expected, a better accuracy. As can be seen, the accuracy of H_2 is slightly worse than H_1. This is explained by issues in finding a correct parametrization for voting and storing phases (see Section 3.3).

Scattered sphere test Point scattering can be considered a condition similar to the presence of outliers, since mismatched points are too distant from the model and cannot be considered as part of it. Empirical measurements showed that typical scattering values correspond to a 30% outlier rate, which is far below the threshold in Eq. 17. This observation is confirmed by our experiments, where spheres from dataset D2 have been tested with increasing input sizes. The results are summarized in Fig. 5, second column, where H_4 performs better than H_1.

Distorted sphere test Concerning axis distortions (both linear and non-linear), the H_4 algorithm remains higher performing and more stable than other variants, as confirmed by our tests summarized in columns 3 and 4 of Fig. 5. Tests were made with spheres taken from datasets D3 and D4, using increasing input sizes.

Outlier sphere test Tests executed on dataset D5 should exhibit an inverted trend with respect to the others, where distortions and errors can be modeled as a small amount of outliers. As can be seen in Fig. 5 (column 5), this is not actually true. The H_4 algorithm remains more accurate, although many more draws are necessary to achieve satisfactory detection (according to observations taken in Section 3.5). Nevertheless, H_1 remains interesting, since its execution time is adjustable by modifying the quantization level and is less sensitive to the estimation of the inlier rate. Furthermore, usually when H_1 fails the estimation it is because it tends to find the point that maximizes the number of features that are equally distant to it. This results in a bigger sphere that is tangent to the target and also collects other spurious points (notice the high peak on column 5 of Fig. 5b). The region around this sphere can be used in place of the whole cloud to reduce the huge number of quadruples that H_4 needs to find the target (see Fig. 6).

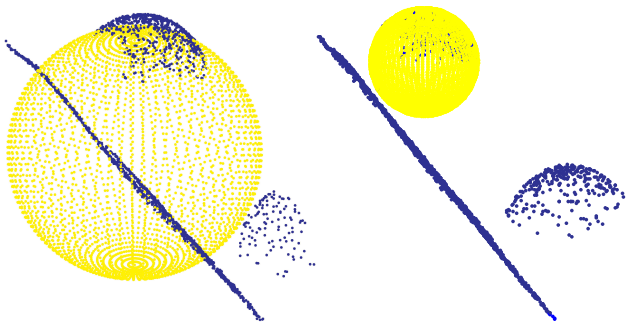
Results of tests on the D6 dataset (Fig. 5, column 6) have commensurate performance with the others, since low amounts of outliers ($< 50\%$), even if correlated, poorly affect algorithm accuracy performance.

5.3 Test of CMPHT on synthetic datasets

Since H_4 outperformed the other variants on almost all previous tests, we decided to test it only, against our method, on a more realistic scenario. For the test we selected 10 instances from D7 and computed the average error of 100 iterations on the same instance. In this case the stopping criterion was a maximum number of points (quadruples) extracted from Eq. 13 and the result shown is the best obtained within that threshold. In Fig.7 we show that our method (red diamonds for distance error, green triangle for radius error) is faster and more accurate than H_4 (blue squares for distance error, yellow triangle for radius error), since a smaller search volume with few outliers allows to compute fewer votes and to perform a finer search with the same accumulator size.

5.4 Test of CMPHT on real datasets

To evaluate the performance of our method on a real scenario, we tested it against the Real Dataset and compared it with a multi-target version of H_4. The stopping criterion and selection of the results are the same as the tests with D7. The ground truth was obtained with a least squares method applied to manually segmented targets.



(a) (b)

Fig. 6 Estimation produced by H_1 with coarse quantization (left) and the result after CMPHT with the same input and quantization step (right). An RGB picture of the whole cloud is depicted in Fig. 4 (top left image).

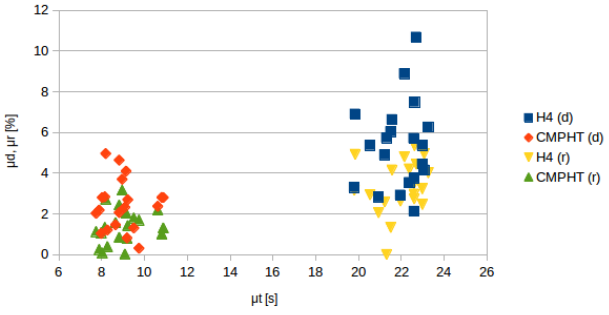


Fig. 7 Mean distance error μ_d and Mean radius error μ_r against mean time μ_t for tests on D7

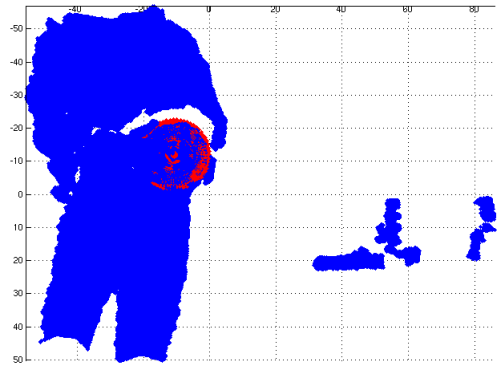


Fig. 8 Result of the detection of CMPHT after background removal

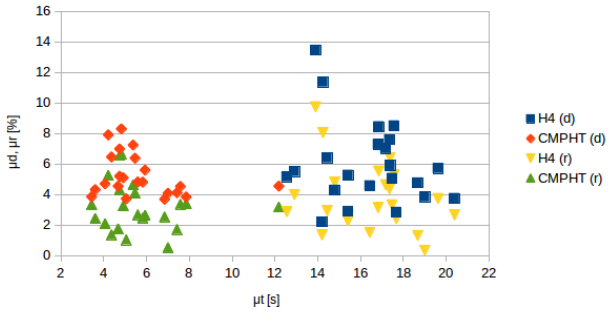


Fig. 9 Mean distance error μ_d and Mean radius error μ_r against mean time μ_t for tests on the Real Dataset

The results are illustrated in Fig. 9: for both distance and radius error our method outperforms H_4 and achieves more accurate results in less time.

Both algorithms show similar result with respect to tests with synthetic datasets but some spheres were not detected, in particular those in very large clouds (Fig. 4, first two clouds from the left, bottom row, where the door and the table on the background were captured) or in clouds where the target is far from the camera (Fig. 4, seventh cloud from left, top row). This happens for three reasons: first, real clouds can have a volume so extended that, even with the largest allowed quantization step, the H_1 step of CMPHT is very slow. On the other hand, since the cloud is dense, H_4 requires too many samples and thus too much time to be successfully executed. The second reason is device-related and concerns point density; point density decreases with distance [32], thus targets that are far from the sensor can generate more false negatives because the density threshold is not reached (on the other hand, a low density threshold would increase the number of false positives, thus a trade-off is needed). The third reason is related to distortions, which are non-neglectable over 3 m [32].

These observations are confirmed by additional tests with some clouds from the Real Dataset that were limited to 1.5 m (Fig. 8). By eliminating the background, the results with the CMPHT were similar to ones presented in Fig. 9, while H_4 still failed the detection in 70 % of cases.

The background removal step has been introduced since one of the main applications of the proposed method is the detection and 3D localization of balls moved by players (see Section 1). Depending on the final application or the scene topology, the background removal step can be integrated or replaced with a plane removal to filter out points belonging to the floor or even the walls.

In summary, the suitability of the method is related to the type of the cloud: since the first stage of the CMPHT is based on H_1, our method is more suitable for dense, noisy and shallow clouds, because in these cases the quantization is sufficiently coarse to allow a fast computation, but it still remains appropriately below the radius length. In contrast, deep and sparse clouds call for straight multi-point approaches.

Although the assumption of a shallow cloud could be acceptable for many applications (*e.g.,* gesture recognition for entertainment or rehabilitation, where a person is supposed to move more laterally than frontally), further investigations about our future work will include optimization techniques to increase the speed on large clouds and the accuracy of objects far from the camera.

Another cause of detection failure is strong distractors, (*i.e.,* dense groups of points that satisfy the constraint of Eq. 2). For instance, points sampled from intersecting planes are mostly equally distant to one single point. In this case, the HT fails and detects a false sphere tangent to planes, as depicted in Fig. 10.

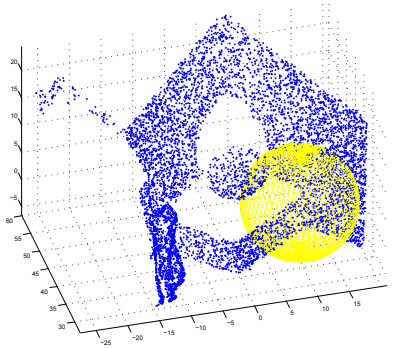


Fig. 10 Estimation produced by H_1 with coarse quantization. An RGB image of the whole cloud is depicted in Fig. 4 (seventh image from the left, top row).

6 Conclusion and Future Work

In this paper we provided a systematic study of different HT algorithms applied to sphere detection in dense and

noisy point clouds. In particular we studied several probabilistic/randomized HT-based methods (*i.e.,* methods that process a random selection of subsets of input points). The various algorithms differ on the fact that one (H_1) uses a single point as input of the transform while the others use multi-point tuples as input. Then we inferred a criterion to determine which is more suitable for a specific condition. Algorithm variants were also compared with intensive tests in different simulated conditions.

We also proposed a novel combined HT-based method (CMPHT) which scans a point cloud in two steps: firstly with a single-point algorithm to identify a region of interest, and secondly a multi-point algorithm refines the final detection. The method has been tested both with a synthetic and a real dataset, the latter being acquired with a Kinect sensor, and showed faster execution times and a more accurate estimation against a straight four-point algorithm, although it is less stable with real datasets, due to strong distractors, especially corner walls.

We believe that the proposed algorithm will be very useful in future 3D object recognition applications.

Future work will focus on two directions: improving the speed of the algorithm against vast clouds, and robustness against strong distractors. Possible improvements are related to further reduction of the quantization step and in particular it will include: planar region and/or background removal, accumulator design optimized for sparse data, similarly to what is used in [30], and data partitioning followed by region growing approaches.

The second direction is related to comparing our algorithm with other approaches. In particular we will consider adapting the method proposed by [29] to slices extracted from the point cloud, similarly to what was achieved by [27] for the specific domain of medical imaging.

A Computing inlier threshold T to compare 1-Point and 4-Point HT methods

Given a point cloud with a certain inlier rate, we want to compute the inlier rate threshold T to identify whether it is more convenient to use a single-point approach or a four-point for sphere detection.

According to [22], the error function for H_1 is:

$$\varepsilon_{H1} = \frac{1}{24} \cdot c^4 \mathrm{e}^{-c} \tag{21}$$

where $c = s_{H1} \cdot N_{\mathrm{in}}/N$. Similarly we have:

$$\varepsilon_{H4} = b \cdot \mathrm{e}^{-b} \tag{22}$$

where $b = s_{H4} \cdot (N_{\mathrm{in}}/N)^4$. From 21 and 22 we have:

$$s_{H1} = s_{H4} \cdot (N_{\mathrm{in}}/N)^3 \cdot \frac{c}{b} \approx s_{H4} \cdot (N_{\mathrm{in}}/N)^3 \tag{23}$$

Total costs of H_1 and H_4 are respectively: $c_{H1} \cdot \overline{S}^3 \cdot s_{H4} \cdot (N_{\mathrm{in}}/N)^3$ and $c_{H4} \cdot s_{H4}$ thus, if we make those costs equal we can compute the threshold $T = N_{\mathrm{in}}/N$ as:

$$T = \sqrt[3]{\frac{c_{H4}}{c_{H1} \cdot \overline{S}^3}} \tag{24}$$

Eq. 24 introduces a criterion for choosing the best between H_1 and H_4 which is based on the estimated inlier rate N_{in}/N of the input cloud. It also shows that the threshold T depends on the medium size of the cells $\overline{S}$, a parameter that can be chosen *a priori*. For small values of $\overline{S}$ and very noisy clouds (*e.g.,* with $N_{\mathrm{in}}/N < 0.05$) the criterion would indicate H_4 as best choice ($N_{\mathrm{in}}/N > T$), but with a high number of quadruples as input and thus with a long execution time. In contrast, for relatively big values of $\overline{S}$ one should prefer H_1, but there is a risk of voting for big spheres that are tangent to the target. CMPHT leverages this property by reducing the search area using H_1 with coarse quantization, and then refining the results with H_4.

References

1. R. Schnabel, R. Wessel, R. Wahl, R. Klein, Shape recognition in 3d point-clouds, in: V. Skala (Ed.), The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2008, UNION Agency-Science Press, 2008.
2. M. van der Glas, F. M. Vos, C. P. Botha, A. M. Vossepoel, Determination of position and radius of ball joints, in: Medical Imaging 2002: Image Processing, Vol. 4684, San Diego, CA, 2002, pp. 1571–1577.
3. T. Rabbani, F. Van Den Heuvel, Efficient hough transform for automatic detection of cylinders in point clouds, ISPRS WG III/3, III/4 3 (2005) 60–65.
4. P. V. C. Hough, Method and means for recognizing complex patterns (12 1962).
5. D. Ballard, Generalizing the hough transform to detect arbitrary shapes, Pattern Recognition 13 (2) (1981) 111–122.
6. D. Borrmann, J. Elseberg, K. Lingemann, A. Nchter, The 3d hough transform for plane detection in point clouds: A review and a new accumulator design, 3D Research 2 (2011) 1–13, 10.1007/3DRes.02(2011)3.
7. F. Tarsha-Kurdi, T. Landes, P. Grussenmeyer, Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data, in: P. Rönnholm, H. Hyyppä, J. Hyyppä (Eds.), ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007, Espoo, Finland, 2007, pp. 407–412.
8. D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, A data structure for the 3d hough transform for plane detection, in: Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV '10), Lecce, Italy, 2010.
9. M. Bernal-Marin, E. Bayro-Corrochano, Integration of hough transform of lines and planes in the framework of conformal geometric algebra for 2d and 3d robot vision, Pattern Recognition Letters 32 (16) (2011) 2213–2223.
10. J. Illingworth, J. Kittler, A survey of the hough transform, Computer Vision, Graphics, and Image Processing 44 (1) (1988) 87–116.
11. P. Hart, How the hough transform was invented [dsp history], Signal Processing Magazine, IEEE 26 (6) (2009) 18–22.
12. P. Bastien, L. Dunn, Global transformations in pattern recognition of bubble chamber photographs, Computers, IEEE Transactions on C-20 (9) (1971) 995–1001.
13. M. J. Bazin, J. W. Benoit, Off-line global approach to pattern recognition for bubble chamber pictures, Nuclear Science, IEEE Transactions on 12 (4) (1965) 291 – 293.
14. A. Rosenfeld, Picture processing by computer, ACM Comput. Surv. 1 (3) (1969) 147–176.
15. 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.

16. J. Illingworth, J. Kittler, The adaptive hough transform, Pattern Analysis and Machine Intelligence, IEEE Transactions on PAMI-9 (5) (1987) 690 –698.
17. S. Tsuji, F. Matsumoto, Detection of ellipses by a modified hough transformation, Computers, IEEE Transactions on C-27 (8) (1978) 777–781.
18. H. K. Yuen, J. Princen, J. Illingworth, J. Kittler, Comparative study of hough transform methods for circle finding, Image Vision Computing 8 (1) (1990) 71–77.
19. N. Kiryati, Y. Eldar, A. Bruckstein, A probabilistic hough transform, Pattern Recognition 24 (4) (1991) 303 – 316.
20. L. Xu, E. Oja, P. Kultanen, A new curve detection method: Randomized hough transform (rht), Pattern Recognition Letters 11 (5) (1990) 331–338.
21. P. Kultanen, L. Xu, E. Oja, Randomized hough transform (rht), in: Pattern Recognition, 1990. Proceedings., 10th International Conference on, Vol. i, 1990, pp. 631–635 vol.1.
22. N. Kiryati, H. Kälviäinen, S. Alaoutinen, Randomized or probabilistic hough transform: unified performance evaluation, Pattern Recognition Letters 21 (13-14) (2000) 1157–1164, selected Papers from The 11th Scandinavian Conference on Image.
23. W. Lu, J. Tan, Detection of incomplete ellipse in images with strong noise by iterative randomized hough transform (irht), Pattern Recognition 41 (4) (2008) 1268 – 1279.
24. J. Princen, J. Illingworth, J. Kittler, A formal definition of the hough transform: Properties and relationships, Journal of Mathematical Imaging and Vision 1 (1992) 153–168.
25. C.-C. Hsu, J. S. Huang, Partitioned hough transform for ellipsoid detection, Pattern Recogn. 23 (3-4) (1990) 275–282.
26. R. Taylor, An efficient implementation of decomposable parameter spaces, in: Pattern Recognition, 1990. Proceedings., 10th International Conference on, Vol. 1, 1990, pp. 613–619.
27. M. Cao, C. Ye, O. Doessel, C. Liu, Spherical parameter detection based on hierarchical hough transform, Pattern Recognition Letters 27 (9) (2006) 980 – 986.
28. O. O. Ogundana, C. R. Coggrave, R. L. Burguete, J. M. Huntley, Fast hough transform for automated detection of spheres in three-dimensional point clouds, Optical Engineering 46 (5) (2007) 051002–1–051002–11.
29. M. Kharbat, N. Aouf, A. Tsourdos, B. White, Sphere detection and tracking for a space capturing operation, in: Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on, 2007, pp. 182–187.
30. A. Abuzaina, M. Nixon, J. Carter, Sphere detection in kinect point clouds via the 3d hough transform, in: R. Wilson, E. Hancock, A. Bors, W. Smith (Eds.), Computer Analysis of Images and Patterns, Vol. 8048 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 290–297.
31. K. Khoshelham, Accuracy analysis of kinect depth data, ISPRS Workshop Laser Scanning 38 (2011) 1.
32. J. Smisek, M. Jancosek, T. Pajdla, 3d with kinect, in: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, 2011, pp. 1154–1160.