SPREAD: On Spherical Part Recognition by Axial Discretization in 4D Hough Space

Radhika Mittal and Partha Bhowmick

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, India
{radhikamittal.iitkgp@gmail.com, bhowmick@gmail.com}

Abstract. A novel algorithm is proposed to locate the sets of adjacent co-spherical triangles for a given object, which enables us to detect spheres and spherical parts constituting the object. An extension of the idea of Hough transform has been used, aided by axial discretization and restricted searching, along with the geometric data structure of doubly connected edge list. The algorithm has been analyzed and shown to achieve significant efficiency in space and run-time. On testing the algorithm with various 3D objects, it is found to produce the desired result. Effects of different input parameters have been explained and the robustness of the algorithm has been shown for rough/noisy surfaces.

1 Introduction

Recognition and analysis of 3D shapes in general, and spherical components in particular, have gained a significant research perspective for several important applications, e.g., detection of pulmonary nodules [19], medical analysis of brains [15], etc. Among different geometric approaches for 3D shape analysis [9,11], the most commonly used is Generalized Hough Transform (GHT) [1, 6], as it can easily be adapted based on specific needs of the application, e.g., [3, 7, 16–18].

We propose here an extension of Hough transform to detect spherical components, using a more space- and time-efficient technique of restricted discretization and localized searching instead of processing a 4D array containing the 3D coordinates of the center and the radius. To work with a significantly large range with a high accuracy, we use the novel idea of *axial discretization*. We take discrete points along the *circum-normal* (axis) of a face (triangle) and use restricted Hough transform to locate the parameters of the sphere it potentially belongs to, with respect to the given object. The restricted HT is based on the observation that if a face belongs to a particular sphere in the given object, then with a very high probability, its adjacent faces would also belong to the same sphere. The geometric data structure of DCEL has been used extensively to achieve an attractive run-time, as shown in this paper.

2 Proposed Algorithm

One of the novel principles while designing the algorithm SPREAD lies in the concept of *axial discretization*. Such a discretization has been resorted to in

order to reduce the computational burden of $O(n^4)$ for the conventional Hough transform (HT) [10,14]. This technique has not been used so far for recognizing spherical parts in 3D objects, although proposed to detect circular arcs in 2D images for several applications [4,5,8,12].

2.1 Principle of SPREAD

We use the following observation: If a (triangular) face f belongs to a particular spherical part in the given 3D object, then with a very high probability, the three faces adjacent to f would also be co-spheric with f. Based on this observation, we estimate the parameters, i.e., center and radius of the sphere, which all the adjacent faces potentially belong to. The algorithm SPREAD considers the discrete points along the *circum-normals* of these faces as input, and finds the approximate center. (The circum-normal of a face f is the normal to f that passes through its circum-center.) The first face f is selected at random. Although there exists a unique circle passing through the three vertices of any triangle, there exists an infinitude of spheres passing through them. The centers of all these spheres are collinear in 3D space. Hence, for a given face f with a center point taken on its circum-normal, the sphere parameters are first obtained, and then we build the initial spherical part on it by considering its three adjacent triangles to obtain the common solution space of these four triangles. The larger spherical part is built from this by doing a breadth-first search iteratively on the set of faces adjacent to each other whose corresponding *ε-balls* in the parameter space have all pair-wise intersection, as explained next.

The parameter space has four dimensions, the first three being corresponding to the three coordinates of the object space and the fourth to the radius. The distance metric used by us in this parameter space is based on L_2-norm. If p_i be the point in the 4D parameter space corresponding to a sphere S_i passing through the vertices of a face f, then the ε-ball corresponding to S_i is denoted by $\phi(p_i, \varepsilon)$ and defined as the 4D hyper-sphere with center at p_i and radius ε. Note that, if S_i has center $c_i = (x_i, y_i, z_i)$ and radius r_i, then it is uniquely represented in the parameter space by the point $p_i = (c_i, r_i) = (x_i, y_i, z_i, r_i)$. Hence, two faces f_i (with circum-normal $\mathbf{N}_i$) and f_j (with circum-normal $\mathbf{N}_j$) are said to be *co-spheric* with each other if and only if there exist two points $p_i(a_i, r_i)$ and $p_j(a_j, r_j)$ in the parameter space such that $a_i \in \mathbf{N}_i$, $a_j \in \mathbf{N}_j$, and there exists a non-empty intersection between $\phi(p_i, \varepsilon)$ and $\phi(p_j, \varepsilon)$. Evidently, two ε-balls are said to intersect each other if and only if the distance between their centers does not exceed 2ε. In general, a set of faces are mutually co-spheric if the two ε-balls corresponding to some axial point of one and some axial point of the other in each pair from the set have a non-empty intersection in the 4D parameter space.

The proposed algorithm SPREAD is made flexible to suit the needs of different objects by providing the hyper-ball radius, ε. The value of ε is specified w.r.t. the normalized scale, and varies from 0.01 to 0.04, as shown in Sec. 3. Smaller the value of ε, tighter is the detection of spherical parts, and larger the value of ε, looser is the detection. The speed and precision of the algorithm is

decided by another parameter, namely N, which indicates the number of discrete points to be considered along the circum-normal of each triangle while detecting a spherical part. And lastly, the minimum number of constituent faces of a co-spheric part, namely F, is considered to decide whether the concerned part can be reported as a practically spherical part. The exact usage of these parameters is described in detail later in this section and also in Sec. 3.

To maintain uniformity across different sizes of objects, we apply an isotropic scaling on the object by normalizing the coordinates of all the vertices of the triangulated object surface such that the maximum over the distances for all pairs of vertices is a constant (taken as 10 in our experimentation). We maintain a *Doubly Connected Edge List* (DCEL) [2] which helps in finding the adjacent faces. In this list, we store, for each edge, its source vertex, destination vertex, incident face, and twin edge. The twin $twin[e]$ of an edge e is the edge whose incident face is adjacent to the face incident on e. Note that, for each face, each of its incident (directed) edges is defined in a manner so that the face lies left of the edge. Thus, the source vertex of e coincides with the destination vertex of $twin[e]$, and vice versa (Fig. 1).

For each triangle, we find its circum-center coordinates (x_c, y_c, z_c) as follows.

$$x_c = \frac{\sum_{i=0}^{2} x_i \left(l_i^2 \left(l_{(i+1) \bmod 3}^2 + l_{(i+2) \bmod 3}^2 - l_i^2 \right) \right)}{2 \sum_{i=0}^{2} \left(l_i^2 l_{(i+1) \bmod 3}^2 \right) - \sum_{i=0}^{2} l_i^4}$$

where $l_i (i = 0, 1, 2)$ denotes the length of the side opposite to the ith vertex having coordinates (x_i, y_i, z_i). The corresponding formulas for finding y_c or z_c are similar, and obtained simply by replacing x_i with y_i or z_i, as applicable. The unit circum-normal of each triangle is found by normalizing the cross product of two of its edges, and then by translating it to the circum-center.

We start the process of detecting co-spherical sets of faces with all faces as initially 'unmarked'. Then a face f is picked at random. We find the three faces adjacent to f using the DCEL in constant time (Fig. 1). For each such adjacent face f_i, we uniformly discretize a finite segment of the circum-normal $\mathbf{N}$ passing through its circum-center. The discretization procedure samples $2N + 1$ points on each circum-normal: N points on either side of the plane of the face and one lying on it. They are sampled in a manner such that the distance between the farthest pair of sample points along the concerned circum-normal is a constant (taken to be 100 times the size of the normalized object, in our experimentation). If there exists a sample point $a \in \mathbf{N}$ for f and a sample point $a_i \in \mathbf{N}_i$ for a face f_i with $p = (a, r)$ and $p_i = (a_i, r_i)$, where r is the distance of the point a from a vertex of f (all vertices of f being equidistant from a) and r_i is that of a_i from a vertex of f_i, such that $\phi(p, \varepsilon) \cap \phi(p_i, \varepsilon) \neq \emptyset$, then there exists a solution for the common sphere of f and f_i (see Fig. 1). The parameters corresponding to this solution are obtained from $\phi(p, \varepsilon) \cap \phi(p_i, \varepsilon)$.

We maintain a queue Q to store all the faces that are co-spheric with f. Thus, all the adjacent faces which have been taken into consideration are enqueued in

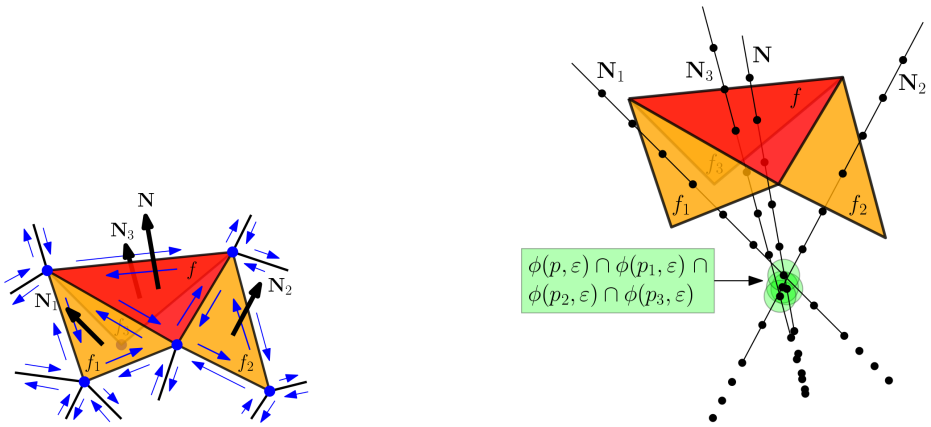


Fig. 1. Left: Information in DCEL. The red triangle denotes a face f having three (directed) edges, shown in blue. Each of of three edges is defined by a pair of vertices (blue), and has a twin edge in its adjacent face (orange). The unit normals are computed from the edges corresponding to the faces. **Right:** Finding the center of the sphere built by a face f (shown in red) and its three adjacent faces $\{f_1, f_2, f_3\}$ (orange), using *axial discretization* of their circum-normals $(\mathbf{N}, \mathbf{N}_1, \mathbf{N}_2, \mathbf{N}_3)$.

Q along with the face f; these faces are marked ON before being enqueued. We calculate the mean of all satisfying points on the discretized circum-normals to estimate the effective center and radius of the sphere S built by the face f and its surrounding faces. If the estimated radius is larger than a sufficiently large constant (taken to be twice the normalized size of the object in our experimentation), then Q is emptied, indicating that no sphere is formed by f. This is done because a very large radius implies a planar segment instead of a spherical patch.

We build a spherical part S using breadth-first-search. For each face in Q, we find the adjacent faces using the DCEL. If an adjacent face f_i, thus found, is not marked ON, then we check whether there exists a point a_i in the discretized circum-normal of f_i such that $\phi(p_i, \varepsilon)$ has a non-empty intersection with the existing parameter space corresponding to S. If it has a non-empty intersection, then the face f_i is marked ON and enqueued in Q. We continue expansion on each face in Q, until Q stops growing or all faces in Q have been considered.

Speeding up the Algorithm To speed up the procedure of searching the matching points of discretization, if any, while verifying the intersection of one ε-ball with the solution space of the current sphere S, we use the range of indices of the discrete points that correspond to the current solution. Note that, the index varies from $-N$ to N, 0 inclusive. Let $[s_{\min}, s_{\max}]$ be the range of indices that have produced the current solution. Here, $s_{\min}$ is the lowest index of one or more discrete points of some of the circum-normals corresponding to S, and $s_{\max}$ is the highest. If a new face f_i belongs to S, then it is expected

that the index of the discrete point a_i from the circum-normal of f_i would lie in or close to the range $[s_{\min}, s_{\max}]$. In fact, if the face f_i is comparable in size with the 'marked faces' corresponding to S, then the index of a_i would lie in $[s_{\min}, s_{\max}]$; otherwise, a_i would lie outside but close to $[s_{\min}, s_{\max}]$. The length of $[s_{\min}, s_{\max}]$ is also quite small if the sizes of the 'marked faces' have a small variation. Hence, the search for the matching discrete point for the new face f_i is guided by $[s_{\min}, s_{\max}]$. The first point picked up from the discretized circum-normal of f_i has index $s_0 = \frac{1}{2}\lfloor s_{\min} + s_{\max} + 1 \rfloor$. If it matches, then no more search is necessary; otherwise the indices of two subsequent points under consideration are $s_0 \pm 1$, next are $s_0 \pm 2$, and so on, until the match is found or the face f_i is rejected for a diverging match with the current solution. A diverging match is reported when for two points in succession, the ε-ball is found to be digressing away from the current solution.

Post-processing The co-spherical components are colored by the algorithm SPREAD using distinct colors. The coloring is constrained by the condition that each spherical part must have a minimum number of faces. This is to avoid detection of small undesired spherical components and also to avoid detection of cylindrical or similar shapes whose faces may be misinterpreted to be forming a spherical part. Hence, an important restriction is that a spherical component will be colored only if its area is greater than $F \cdot A_{\max}$, where F is the tuning parameter lying in $(0, 1)$ and $A_{\max}$ denotes the area of the largest spherical component found. Post-processing is done by SPREAD using the color information in order to fill up the gaps created by the undetected faces in the detected spherical parts. If a set A of adjacent faces is uncolored and all the faces adjacent to the boundary faces of this set have the same color (i.e., belong to the same sphere, S), then we check whether the area formed by the set A is less than a threshold (taken as 0.1 times the area of the corresponding sphere). If so, the set A is considered to be 'falsely undetected', an hence the color of each face of A is set to the color of the sphere S.

2.2 Complexity Analysis

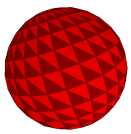
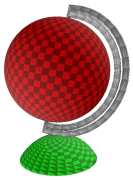
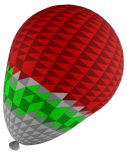
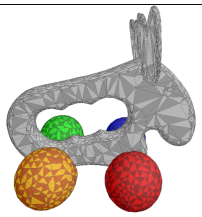
Let the number of vertices be v, the number of faces be n, and the number of edges be e. Then the DCEL takes $O(v + n + e) = O(n)$ space, since $v = O(n)$ and $e = O(n)$. Apart from the conventional information as mentioned earlier, the DCEL also stores the details of faces, vertices, face areas, normal equations, and circum-centers, taking $O(n)$ space. The queue Q stores all the faces that are currently co-spherical. The *best case* occurs when there is no overlap between two co-spherical sets, resulting in a space complexity of $O(n)$. In the *worst case*, there can be many overlaps among these co-spherical sets. An instance of the queue may contain as many as $O(n)$ faces and there can be $O(n)$ such instances of the queue. However, when a particular instance of Q is processed, other instances do not occupy any space, since they either have been processed earlier or would be processed later. This gives us a space complexity of $O(n)$ for Q.

It takes $O(v+n)$ time to read the object, $O(v^2)$ time to normalize the vertices by computing the maximum distance, and $O(n^2)$ time to prepare the DCEL. The algorithm checks no more faces adjacent to a face f_{ki} when f_{ki} is found to be not matching with the spherical part S_k currently under consideration. For each spherical part, the first two faces are obtained in $O(N)$ time by index searching. Let the number of faces of S_k be n_k. All these n_k faces can have $O(n_k)$ adjacent faces which are not part of S_k; and each of these faces will be tested, resulting to no positive result. Hence, the time complexity to report S_k is given by the sum of time complexity to report the faces of S_k and that to check the mismatching faces. The *best-case* time complexity is, therefore, given by $O(N)+(n_k-2)O(1)+O(n_k)O(1) = O(n_k+N)$; summing up over all spherical parts and observing that n_k is bounded by $O(n)$ for this sum, we get $O(n+mN)$, where m is the number of spherical parts reported by SPREAD. For the *worst case*, we sum up $O(N) + (n_k - 2)O(N) + O(n_k)O(N) = O(n_kN)$ over all parts, hence getting $O(nN)$.

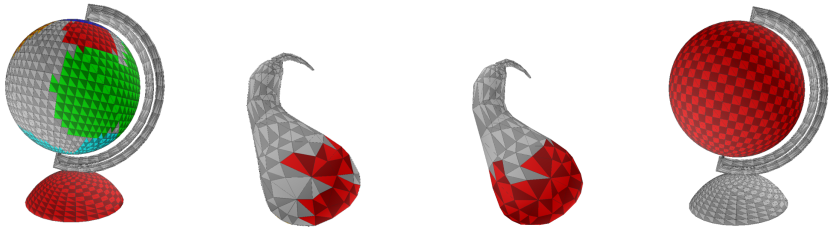
3 Experiment and Results

We have implemented the algorithm SPREAD in C using OpenGL on an Intel(R) Core(TM)2 Duo CPU E4500 2.20 GHz machine, the OS being Ubuntu Release 10.10. Table 1 shows its results on some objects having one or more spherical parts. The object `Sphere` is completely covered with $\varepsilon = 0.03$; note that 759 out of 840 triangles are initially detected, and the remaining 81 triangles get included during post-processing (Sec. 2.1).

Table 1. Summary of results for running SPREAD on different objects ($\varepsilon - 0.03, F = 0.10, N = 1000$). n = number of faces; m = number of spheres detected; n_k = number(s) of faces for each sphere (shown comma-separated).

Parameter	 `Sphere`	 `Globe`	 `Balloon`	 `Elk`
n	840	7618	2080	6518
m	1	2	2	4
n_k	759	3968, 927	1384, 134	513, 492, 483, 476
CPU time (sec.)	0.596	2.610	0.759	3.542

We take N in the range of 1000–2000; for a low value, the points on the normal are largely separated from each other, thereby giving a bad estimate (Fig. 2). The specified value of ε determines the accuracy with which spheres



(a) $(0.04, 0.10, 200)$ (b) $(0.04, 0.08, 2000)$ (c) $(0.06, 0.08, 2000)$ (d) $(0.04, 0.3, 1000)$.

Fig. 2. Output instances showing effect of user paramester (ε, F, N): (a) too small value of N; (b, c) change in ε; (d) large value of F. (For proper results, see Table 1.)

are detected (Fig. 2). Changing ε by even 0.02 units (in normalized object scale of unity) significantly affects the number of co-spherical triangles. A high value of ε would encompass unwanted triangles in a spherical component, whereas a low value would discard many desired triangles, thereby hindering the formation of a larger spherical part. For F, a high value may not correctly report a spherical part (Fig. 2); when $F = 0.3$ (see correct result in Table 1), base part of the `globe` is not reported as spherical as its area is appreciably less than the red one.

Post-processing takes care of missing triangles in a spherical part, possibly due to staggering circum-normals. Figure 3(a) shows the result of running SPREAD on `elk`, without post-processing. For testing SPREAD against noisy or rough surface, we have prepared some rough surfaces by introducing Gaussian noise. Figure 3(b) shows that on introducing Gaussian noise with a peak of 1% of the maximum distance (0.05), fairly good results are obtained without post-processing. However, for a Gaussian noise of 2%, we get better results on doing post-processing (Fig. 3(c, d)).

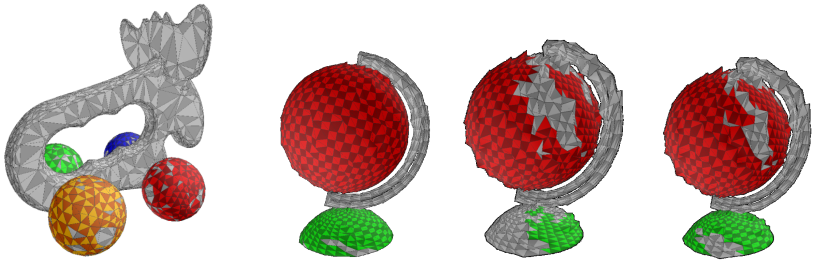


Fig. 3. Role of post-processing: (a) Result on `elk` without post-processing (see Table 1 for result after post-processing. (b-d) Results for rough surfaces, with and without 2nd stage processing. (b) 1% noise, no post-processing; (c) 2% noise, no post-processing; (d) 2% noise, after post-processing ($N = 1000, \varepsilon = 0.06, F = 0.1$ for all three cases).

4 Conclusion

The proposed algorithm SPREAD can detect the adjacent co-spherical triangles for a given object using axial discretization based on the contemporary idea of restricted Hough transform, resulting in usage of less space as compared to the generalized Hough transform. We have tested the algorithm on various 3D objects, have analyzed the effects of different input parameters, and have tested its robustness rough or noisy surfaces. All these results exhibit that the algorithm is quite resistant to noise and can generate fairly accurate results with a space- and time-efficient manner. As a future work, the algorithm can be extended to detect other 3D shapes like ellipsoids and paraboloids by properly applying the concept of axial discretization.

Acknowledgement
A part of this work has been sponsored by the NSF project: NEBULA (CNS-1038695).

References

1. Ballard, D.H.: Readings in computer vision: Issues, problems, principles, and paradigms, 1987.
2. Berg, M.D. *et al.*: *Computational Geometry Algorithms and Applications*, 2000.
3. Cao, M.Y. *et al.*: Spherical parameter detection based on hierarchical Hough transform. *PRL*, 27:980–986, 2006.
4. Chiu, S.H. and Liaw, J.J.: An effective voting method for circle detection. *PRL*, 26(2):121–133, 2005.
5. Davies, E.R.: A high speed algorithm for circular object detection. *PRL*, 6:323–333, 1987.
6. Duda, R.O. and Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15:11–15, 1972.
7. Ecabert, O. and Thiran, J.P.: Adaptive Hough transform for the detection of natural shapes under weak affine transformations. *PRL*, 25(12):1411–1419, 2004.
8. Foresti, G.L. *et al.*: Circular arc extraction by direct clustering in a 3D Hough parameter space. *Signal Processing*, 41:203–224, 1995.
9. Gelfand, N. and Guibas, L.J.: Shape segmentation using local slippage analysis. In *Proc. SGP '04*, pages 214–223, 2004.
10. Gonzalez, R.C. and Woods, R.E.: *Digital Image Processing.* Addison-Wesley, California, 1993.
11. Hofer, M.: *et al.* 3d shape recognition and reconstruction based on line element geometry. In *Proc. ICCV '05*, pages 1532–1538, 2005.
12. Ioannoua, D. *et al.*: Circle recognition through a 2D Hough transform and radius histogramming. *IVC*, 17:15–26, 1999.
13. Kim, H.S. and Kim, J.H.: A two-step circle detection algorithm from the intersecting chords. *PRL*, 22(6-7):787–798, 2001.
14. Leavers, V.: Survey: Which Hough transform? *CVGIP*, 58:250–264, 1993.
15. Nain, D. *et al.*: Statistical shape analysis of brain structures using spherical wavelets. In *Proc. ISBI '07*, pages 209–212, 2007.
16. Ogundana, O.O. *et al.*: Fast Hough transform for automated detection of spheres in three-dimensional point clouds. *Opt. Eng. 46*, 2007.

17. Rong, F. *et al.*: A novel Hough transform algorithm for multi-objective detection. In *Proc. IITA '09*, pages 705–708, 2009.
18. Yang, M.C.K. *et al.*: Hough transform modified by line connectivity and line thickness. *IEEE Trans. PAMI*, 19:905–910, 1997.
19. Zhang, X. *et al.*: A new method for spherical object detection and its application to computer aided detection of pulmonary nodules in CT images. In *Proc. MICCAI '07*, pages 842–849, 2007.