Autonomous Perching Quadcopter

Yucheng Chen
School of Aeronautics and Astronautics
Purdue University
chen1359@purdue.edu

Abstract

In various UAV missions, one requires the UAV to stay stationary and make observations. This scenario is the motivation of the Autonomous Perching Quadcopter project. To save energy, it is a better choice to make the UAV perch on some support like a log instead of having it hover over the target. This paper deals with the vision part of the project: to enable a UAV to identify a log-shaped object for perching. The abstraction of the problem is to extract a cylinder (assume most of logs are cylindrical) from a 3D point cloud. After reviewing some related previous work, we believe that RANSAC algorithms can be a vital approach to solve this kind of problem because this method is able to pick out the cylinder with reasonable accuracy in a relatively short period of time because it keeps on sampling from the point cloud and avoids performing computations on the entire point cloud directly. This paper, therefore, incorporates the idea of RANSAC algorithm and, unlike the previous work that combines RANSAC and LMS(Least-Mean-Square) algorithm, focuses on RASAC itself to extract a cylinder in a 3D scene. The reason for only using RANSAC is that omitting the optimization step(LMS) can save a lot of time. The systematic approach includes two steps: the first step determines the axis(orientation) of the cylinder and the second, based on the axis computed in the first step, determines the radius and extracts the cylinder.

Introduction

The current design of our quadcopter is that a stereo camera is mounted in the front. The camera captures a stream of stereo images, which will then be converted to 3D point cloud. This paper focus on the processing of the 3D point cloud. The conversion from stereo image to 3D scene will be covered in future work.

Before starting coming up with the solution in this paper, the author reviewed some related works. A basic but interesting method of extracting cylinder is introduced in [3]. It defines a way to use 3 parameters: axis vector, radius and a point on the axis to determine a cylinder. This method is also used in this paper. However, the approach that [3] uses is impractical: it purely relies on RANSAC algorithms without any processing on the point cloud. Thus, the approach requires very large iteration times of the RANSAC. Experiments show that to achieve 90% accuracy, the running time of the MATLAB code is over 30 seconds. [5] introduces another way to define a right cylinder. Select four points on the surface of the cylinder and connect them, and if two of the lines are parallel, the cylinder is a right cylinder. [1] introduces the concept of Gauss image, which is used in this paper. Inspired by the ideas from the paper above, the idea of using RANSAC to extract a geometric model from a 3D point cloud in this paper is as follows: randomly sample a certain number of points to compute necessary parameters to determine the model (for a cylinder they are axis, radius and a point on the axis) and test whether the candidate fixes sufficiently many points to be a model. This uses two steps to extract a cylinder model, and both involve RANSAC. The first step determines the axis of the cylinder by computing the normal vector of the plane perpendicular to its axis, taking advantage of the fact the projection of a right cylinder onto that plane is a circle. The second step, using the axis obtained from the last step, keeps sampling points, fitting circles and testing until a candidate is sufficient to be a model.

The rest of the paper will discuss: 1). the two steps in detail and related mathematical concepts; 2). the rationale behind the determination of some parameters such as the iteration times and threshold to filter out false fit; 3). the MATLAB code attached in the appendix.

Detailed idea and the algorithm

1.Parameters to define a cylinder

There are many ways to define a cylinder, but this paper uses 3 parameters: an unit vector **u** parallel to the axis of the cylinder, a point **P**(x0, y0, z0) on the axis and the radius **r** of a cross section circle of the cylinder. Figure 1 illustrates a cylinder with the 3 parameters labeled.

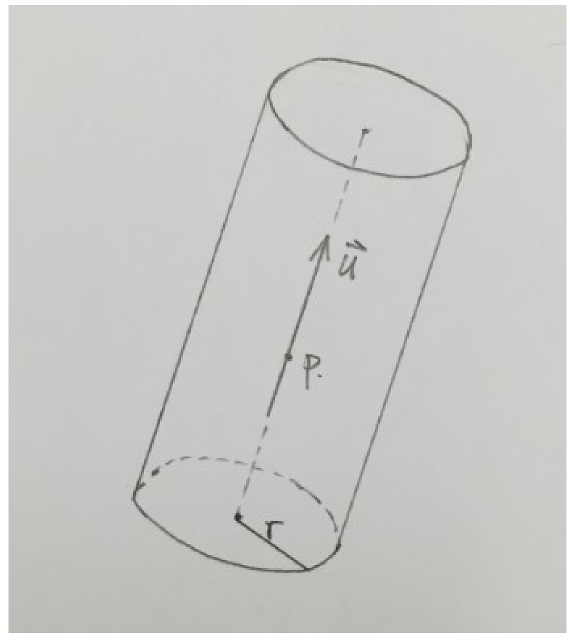

Figure 1. cylinder with 3 parameters labeled

2.Basic idea of RANSAC algorithm

The idea of the RANSAC algorithm for geometric primitive fitting is given as following:

- model
- number of iterations: *numIter*
- error tolerance: *th_err*
- threshold ratio: *th_rat* (minimum ratio of the number of data in the data set to assert a model fit the data)
- k (minimum number of data points to determine a model).

Use k data points to determine the model (i.e. compute the parameters of the model) and calculate the distance from each data point to the model (fitting process). If the distance is less than the threshold, then classify the point as an inlier, and otherwise the point is an outlier. After the fitting process, if the cardinality of the inlier is greater than the current optimal inlier size, update the inlier and the corresponding parameters Iterate the process *numIter* times.

The pseudo code is as follows:

```
bestInlrsize = -1  // size of optimally fitted inlier size
bestParam = empty set   //optimally fitted model parameters
for 1 to numIter:
      Randomly select k points;
      model parameters ← fitting model;
      model ← model parameters;
      dist ← distance (dataset, model)
      for each point p in dataset:
          if (p.distance < th_err)
                inlier.add(p);
          end
      end
      if (length(inlier) > bestInlrsize)
          bestParam ← model parameters;
      end
end
```

3. Determine the orientation of the cylinder

3.1 Gauss Image

The Gauss image is used in determining the orientation of the cylinder. The Gauss image is the mapping from original point cloud to the set of unit normal vectors of each point. The method to obtain the Gauss image, which is also called the Gaussian Sphere, is as follows: for each point, use the k-NN (k-Nearest Neighbors) algorithm to find a certain number of data points around it (100 is used in this project, which both make a good estimation of the unit vector and is not computationally expensive) and make them a subset of the data, then fit a plane to the data set such that the algebraic sum of the signed distance from each point to the plane . Then the normal vector of the plane is the desired normal vector of the point. The way to fit the plane is to use Principal Component Analysis on the subset. The eigenvector corresponding to the smallest eigenvalue is the normal vector of the plane. It is worth noting that the gauss image of a cylinder is a unit circle in a 3D space and the normal vector of the circle is the orientation of the cylinder's symmetry. It is this feature that this report takes advantage of.

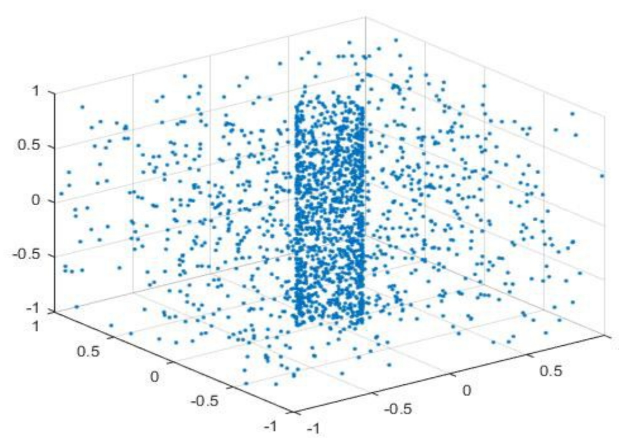

Figure 2. Original point cloud

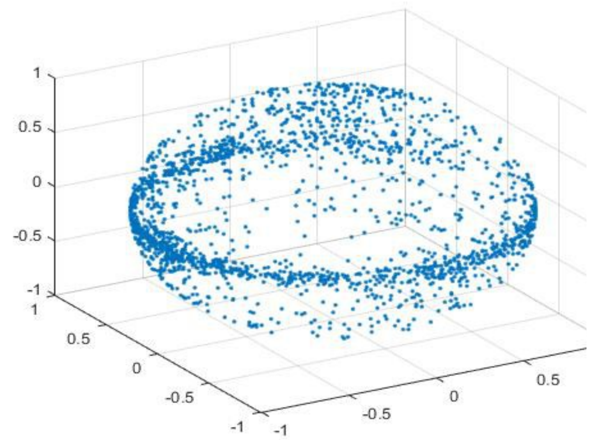

Figure 3. Gauss image

3.2 Algorithm of determining the cylinder axis

The steps to calculate the orientation of the cylinder follows the idea as follows: after mapping the original point cloud to its gauss image, the unit normal vectors of the cylinder in the data set form a great circle of the sphere (i.e. gauss image) as shown in Figure 3. Another way to look at this is that the circle is the intersection of the sphere and a plane passing the origin. Then use RANSAC algorithm to determine the plane (3 points needed). The normal vector of the plane is the orientation of the cylinder.

The pseudo code is as follows:

```
while (k < maxIteration)
    select 2 points from the gauss image;

    if (the two points and origin o are
    collinear)
        continue;
    else
        estimate inliers for the plane,
        if (inliers of new plane > inliers of the
        optimal plane)
            update the optimal plane;
        end
    end
end
return normal vector of the optimal plane u;
```

4. Extract the cylinder

4.1 Determine the radius and center

In the previous step, a plane that is perpendicular to the axis of the cylinder axis is already found. The objective of this step is to find the radius of the cylinder **r** and determine the coordinate of a point **P** on the cylinder axis. The strategy is as follows: go back to the original data set (not the gauss image) and project the data set onto the plane. Then there must be a circle on the plane. Then use the RANSAC algorithm again to fit a circle on the projected plane. Then the radius of the plane is the desired cylinder radius **r**, and the center of the circle is **P**. Then use **P** and **u** to determine the function of cylinder axis,
calculate the distance from each point to the straight line, select the optimal inliers and thus fit the cylinder Figure 1.

The pseudo code is as follows:

```
while (k < maxIteration)
    project the original data set to a plane α
    perpendicular to u;

    on the plane α, select 3 points
    randomly;

    if (3 points are collinear)
        continue;
    else
    r_est, ceneter_est ← circle_fit_3d
    (projected_data, u)

   axis function ← line( u, center_est);
```

dist ← distance (projected_data, axis function);

add a point into inlier its distance to axis is less than threshold;

if (Inlier of new cylinder > Inlier of optimal cylinder)
update parameter of current cylinder;
end
end
return **r, P, u** //all parameters required to determine a cylinder.

4.2 Filter out the false fits

The method to recognize a false fit is as follows: on the projected plane described in the last section (4.1), the points that fit the circle should be discretized in a range of 2π. That is to say, randomly select a point , then the angles between vectors and should be expected to uniformly distributed through 0 to 180 deg. The technique to measure the distribution of the angles is this: plot the histogram of the angles under the edges [0 20 40 60 80 100 120 140 160 180], then calculate the standard deviation of the counts to all bins. Consider if the cylinder is more “incomplete”, the distribution of the counts is less uniform and thus the standard deviation is greater. Therefore, select a threshold value to classify if the cylinder is “complete” or not. The approach of how to select the threshold value will be introduced in the later section.

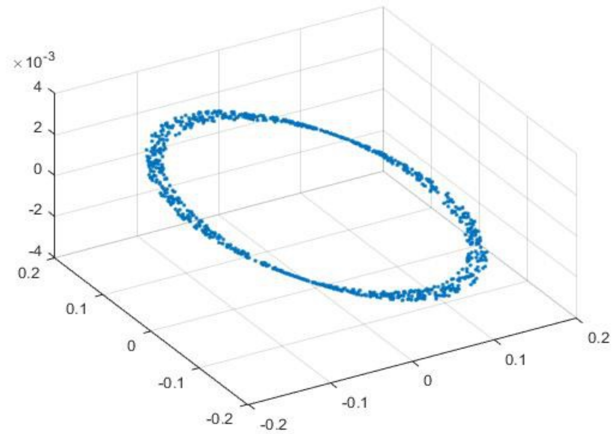

Figure 4. Projected inlier for complete cylinder

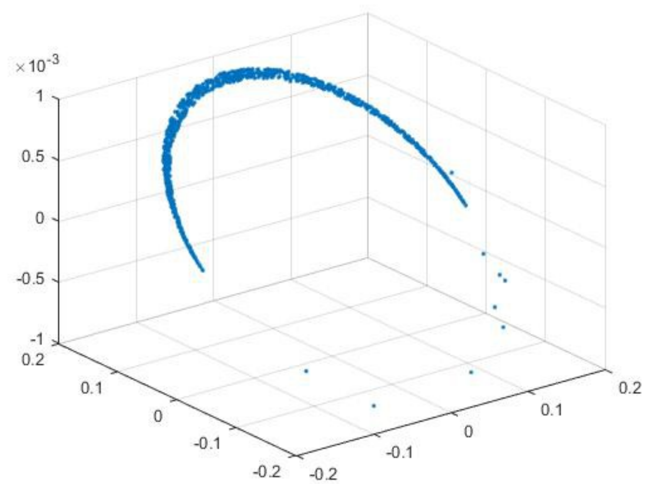

Figure 5. Projected inlier for incomplete cylinder

MATLAB Results: Cylinder Extraction

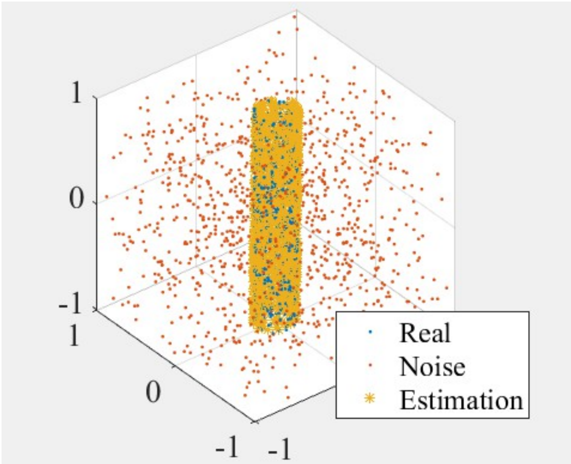

Figure 6. Test No.1

```
Test No.1
    Time elapsed: 0.75 sec.
    Given Radius 0.170000 vs est Radius 0.166370.

    Given AXIS          0      0      1

    estm AXIS          0.0246     0.0015    -0.9997
```

Figure 7. Test No.1 Result

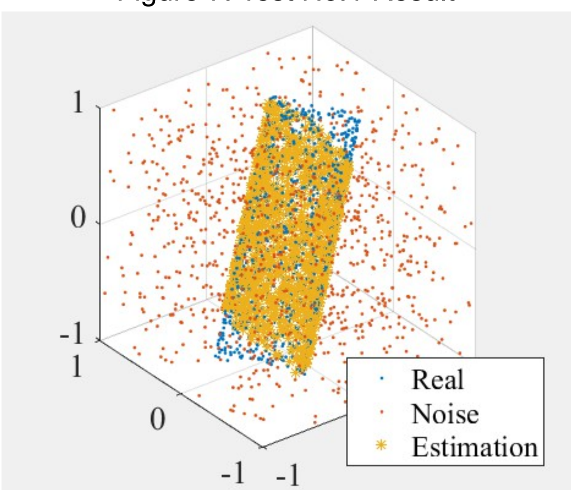

Figure 8. Test No.2

```
Test No.2
    Time elapsed: 0.78 sec.
    Given Radius 0.340000 vs est Radius 0.342005.

    Given AXIS          0.5071     0.3162     0.8018

    estm AXIS           0.5073     0.3374     0.7930
```

Figure 9. Test No.2 Result

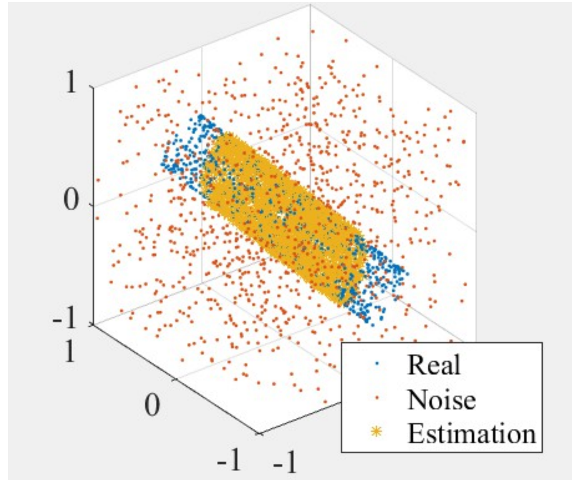

Figure 10. Test No.3

```
Test No.3
    Time elapsed: 0.88 sec.
    Given Radius 0.230000 vs est Radius 0.227025.

    Given AXIS          0.5488    -0.5555    -0.6247

    estm AXIS          -0.5598     0.5536     0.6166
```

Figure 11. Test No.3 Result

Miscellaneous: determination of some parameters

1. Determine the number of iteration in RANSAC algorithm

The process of determining the number of iterations is a trade-off between accuracy and time of execution. For a 3000 data points sample, the RANSAC algorithm iterating 300 times costs 3.88 seconds and the error percentage of the axis estimation is 2.72% and the error percentage for radius estimation is 0.26%. If the number of iterations is reduced to 100, the program takes 0.88 seconds and the error percentage of the axis estimation is 1.21%

and the error percentage for radius estimation is 1.29%.

2. Determine the threshold for a "complete cylinder"

There are 2500 experiments conducted for the standard deviation of the set of number of data points falling in each bin when the given angle is 2.

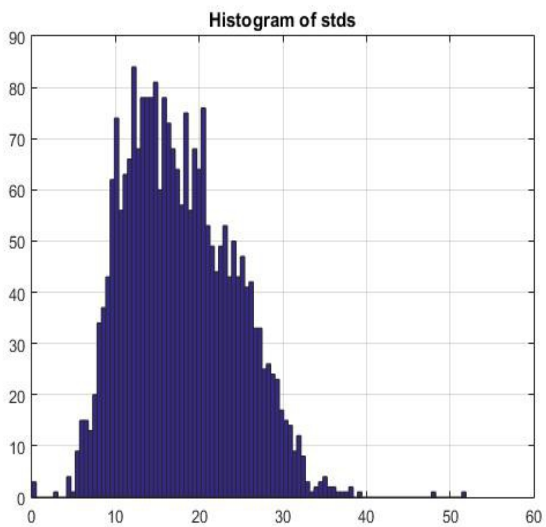

Figure 12.

Similarly,2500 experiments are also conducted for the standard deviation of the set of number of data points falling in each bin when the given angle is .

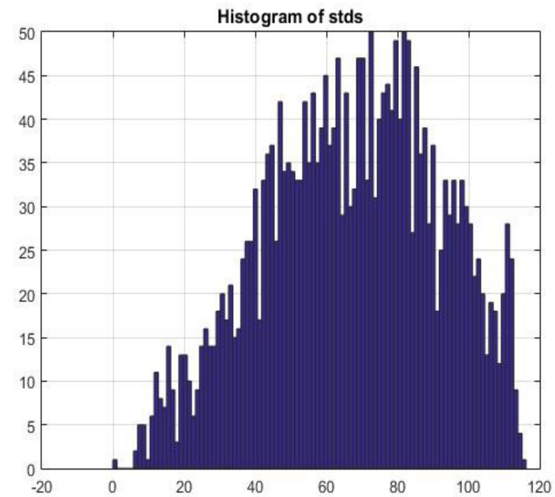

Figure 13.

As the histograms show above, a meaningful threshold to differentiate the complete and incomplete cylinder is **33.** As Figure 9 shows, almost all of the cases fall at the left of 33 and in Figure 10, most cases fall at the right of 3. Another 10,000 experiments have been conducted, the result shows that the accuracy for detecting complete cylinders is **98.26%** and the accuracy for detecting incomplete cylinder is **71.18%**. The raw data of the experiments are available in appendix 1 and 2.

* The MATLAB code for the project above is available [here].

Conclusion and future work

The algorithm works well in extracting a cylinder with high accuracy (98.26%), while the capability to discriminate the false fit (half cylinder) is relatively weak: the accuracy is 71.18%. The future work should focus on how to come up with compound criteria to determine if the cylinder is complete. Also mounting a 3D scanner on a UAV is impractical. The future work includes converting the stereo images into a 3D scene to apply the algorithm to real image.

Acknowledgement

Thanks school of Aeronautics and Astronautics for providing computing resources for this project.

Appendix

1. A portion of data of standard deviation for complete cylinder

10.84102	15.19868	32.69939	13.96822	25.29712	28.84008	12.17009	21.51421	19.79899	21.78175	10.74709
12.99145	26.18683	11.25833	11.98726	12.0185	25.90849	22.42642	10.06783	25.13519	13.95927	16.90496
7.991315	11.3002	17.18122	23.13967	12.79648	34.3394	11.51207	14.89966	26.24405	25.59839	28.01785
18.87974	25.2262	10.86278	17.5934	32.56575	26.18683	8.56511	7.304869	16.63163	7.98088	22.53393
11.18034	9.010796	9.846037	5.060742	19.2101	13.61066	24.93547	8.501634	14.15195	15.06744	21.03832
13.24764	18.83481	14.82771	25.15176	19.23538	13.67581	22.38551	11.42366	17.9312	24.54135	15.93825
24.62214	9.823441	7.314369	15.51164	14.44049	16.21042	23.77908	14.30909	12.00463	24.43358	16.7912
22.13594	27.7083	11.98958	17.07906	14.03567	27.31758	25.10367	10.13657	12.80625	14.48275	11.57704
12.7715	19.39144	26.56648	11.29651	9.275116	23.29759	13.89944	15.42545	14.2741	10.7948	22.63294
16.71825	21.89749	8.472177	18.77498	17.64936	11.31125	11.80042	20.24228	15.00926	35.05868	20.34562
15.01758	17.91647	8.482007	4.41588	12.5344	11.59502	17.5863	26.25357	16.04681	29.84311	25.05217
27.89713	22.33831	9.709674	8.870989	12.19745	14.00893	13.11488	7.729812	25.17163	20.18112	25.69047
8.530989	12.92715	14.05347	11.08803	13.82027	10.89852	17.92422	22.05926	26.55707	22.74557	23.24866
8.838049	23.91652	11.84272	35.92005	14.85018	11.31494	23.55313	18.21401	15.56438	17.37815	9.319931
19.25126	20.67271	9.820613	9.858724	11.33333	23.49527	9.033887	25.12856	28.49756	22.3296	14.50383
21.52582	18.2559	19.96107	19.05547	16.24038	21.35676	22.80594	19.25343	13.27069	12.99786	7.94425
13.11594	16.62829	18.28934	25.53919	26.19849	8.455767	13.86643	17.27072	13.9234	7.763876	13.3988

The entire data set is accessible via this link:
https://drive.google.com/open?id=14bVuLQgHEESzXGqc0GzRSmjBhEWLFHsZIM2FkP3pQT4

2. A portion of data of standard deviation for incomplete cylinder

77.42739	55.55628	82.04233	88.62953	84.29034	47.24081	80.46445	44.02588	88.59709	36.149
95.82899	94.2896	109.9591	107.7904	87.87633	79.38374	112.0057	29.38962	54.09713	28.78271
76.14369	94.31728	102.9624	83.10803	84.65239	56.07906	32.61901	23.60497	100.8551	49.72787
55.3499	47.58676	56.47148	28.07628	107.8202	24.59392	8.56511	36.58703	70.53742	49.59951
79.11033	104.2986	27.32724	76.40535	103.2874	85.0908	66.90291	102.6427	69.30448	77.42739
29.22518	37.96416	89.2386	90.23734	22.74557	90.85015	46.051	59.70785	29.48069	82.69438
36.86386	73.78686	54.06657	72.55687	91.81927	67.70422	66.89876	63.43106	59.75784	79.92809
71.82173	48.28935	86.35248	96.86431	10.80638	56.33161	8.628119	11.06923	86.03746	59.95577
58.46889	87.95706	78.08809	69.10499	110.4887	58.83829	92.54473	90.50552	61.34488	54.85764
46.40851	53.67262	66.29857	67.99632	101.7236	24.9455	64.67805	37.91145	66.75036	25.85537
60.64652	35.31682	100.1762	23.37199	104.3707	39.46025	31.05819	63.65423	97.61717	79.3244
18.60182	78.07689	37.06751	73.52739	104.7729	65.35098	23.4349	85.28987	31.75295	80.68836
86.28731	18.06239	37.4559	105.0202	69.76827	86.92254	64.4233	76.90109	80.79759	21.81233
100.2447	95.75025	13.03627	66.2057	58.00455	46.65952	38.44079	80.50845	86.13797	78.09521
81.81195	34.75629	80.41006	76.98557	29.67228	101.0789	43.74357	77.75942	14.46067	83.0072
31.67851	38.29309	86.25109	54.15487	96.04484	56.70562	97.27281	101.5346	61.6482	84.01042
97.81459	73.71642	46.14952	105.7911	96.95489	87.60058	12.57091	32.28949	110.9705	63.58415
78.8432	84.97663	58.58967	100.2062	47.56078	102.6711	44.81195	105.0512	97.23997	108.2152
93.55539	48.4831	67.29062	94.53101	51.15038	60.15397	12.93252	72.38631	29.2665	73.0504

The entire data set is accessible via this link:
https://drive.google.com/open?id=13ZDf0pD3AjXFnzy6Hnv0jGQCCv6y3Q8AnPMkOpo0NOY

3. Critical MATLAB scripts

3.1 Use RANSAC to find a circle

```
function [rds,ctr,inlier] =
circle_ransac(pts,iterNum,th_d,th_r
)
sampleNum = 3;
ptNum = size(pts,2);
thInlr = round(th_r*ptNum);
inlrsize = -1;
rds = -1;
ctr = [0;0;0];
inlier = [];
for i = 1:iterNum
    distance = zeros(1,ptNum);
    sampleIdx =
randperm(ptNum,sampleNum);
    ptSample = pts(:,sampleIdx);
    p1 = ptSample(:,1);
    p2 = ptSample(:,2);
    p3 = ptSample(:,3);
    if (iscollinear(p1,p2,p3) > 0)
        %fprintf('colliear \n');
        continue;
    end
    %[radius,u_n,center] =
compute_circle(p1,p2,p3);
    [center,radius,v1n,v2nb] =
circlefit3d(p1',p2',p3');
    xprod = cross(v1n,v2nb);
    u_n = xprod/norm(xprod);
```

```
    for p = 1:ptNum
        distance(p) =
norm(cross(u_n,(pts(:,p)-center')));
    end
    inlier_idx = find(abs(distance-
radius) < th_d);
    inlier_size =
length(inlier_idx);
    if inlier_size < thInlr,
continue; end
    if (inlier_size > inlrsize)
       inlrsize = inlier_size;
       rds = radius;
       ctr = center;
       inlier = pts(:,inlier_idx);
    end

end

end
```

3.2 Use RANSAC to fit plane

```
function [normVec,vn1,vn2] =
plane_ransac( pts,iterNum,th_d,th_r
)
sampleNum = 3;
ptNum = size(pts,2);
thInlr = round(th_r*ptNum);
inlrsize = zeros(1,iterNum);
u_ns = zeros(3,iterNum);
v1s =zeros(3,iterNum);
v3s = zeros(3,iterNum);
for i = 1:iterNum
    %pick 3 points to determine a
cylinder
    sampleIdx =
randperm(ptNum,sampleNum-1);
    ptSample = pts(:,sampleIdx);
    p1 = ptSample(:,1);
    p2 = ptSample(:,2);
    p3 = -p1;
    if (iscollinear(p1,p2,p3) > 0)
        fprintf('colliear\n');
        continue;
    end
    [u_n,v1,v3] =
fitplane(p1,p2,p3);
    distance = u_n'*(pts-
repmat(p1,1,ptNum));

    inlier_idx = find(abs(distance)
< th_d);
    inlier_size =
length(inlier_idx);
    inlrsize(i) = inlier_size;
    u_ns(:,i) = u_n;
    v1s(:,i) = v1;
    v3s(:,i) = v3;
    if inlier_size < thInlr,
continue; end
end
[~,index1] = max(inlrsize);
normVec = u_ns(:,index1);
vn1 = v1s(:,index1);
vn2 = v3s(:,index1);
end
```

3.3 Convert points to normal

```
function normals =
points2normals(points)
    % estimating a normal vector
based on nearby 100 points
    % points is 3 * n matrix for n
points

    if size(points,2)==3 &&
size(points,1)~=3
        points = points';
    end

    normals = lsqnormest(points,
100);
end
function n = lsqnormest(p, k)
m = size(p,2);
n = zeros(3,m);

v = ver('stats');
if str2double(v.Version) >= 7.5
    neighbors =
transpose(knnsearch(transpose(p),
transpose(p), 'k', k+1));
else
    neighbors =
k_nearest_neighbors(p, p, k+1);
end

for i = 1:m
    x = p(:,neighbors(2:end, i));
    p_bar = 1/k * sum(x,2);

    P = (x - repmat(p_bar,1,k)) *
transpose(x - repmat(p_bar,1,k));
%spd matrix P
    %P = 2*cov(x);

    [V,D] = eig(P);

    [~, idx] = min(diag(D)); %
choses the smallest eigenvalue
```

```
    n(:,i) = V(:,idx);    % returns
the corresponding eigenvector
end
function
[neighborIds,neighborDistances] =
k_nearest_neighbors(dataMatrix,
queryMatrix, k)

numDataPoints = size(dataMatrix,2);
numQueryPoints =
size(queryMatrix,2);

neighborIds =
zeros(k,numQueryPoints);
neighborDistances =
zeros(k,numQueryPoints);

D = size(dataMatrix, 1);
%dimensionality of points

for i=1:numQueryPoints
    d=zeros(1,numDataPoints);
    for t=1:D % this is to avoid
slow repmat()
        d=d+(dataMatrix(t,:)-
queryMatrix(t,i)).^2;
    end
    for j=1:k
        [s,t] = min(d);
        neighborIds(j,i)=t;

neighborDistances(j,i)=sqrt(s);
        d(t) = NaN; % remove found
number from d
    end
end
```

3.4 Project points to a plane

```
function point = proj2plane(pts,mat)
sol = mat\pts;
sol(3,:) = 0;
point = mat*sol;
end
```

Reference

1. Thomas Chaperon, Francois Goulette Extract cylinders in full 3D data using a random sampling method and the Gaussian image. Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.5998&rep=rep1&type=pdf
2. Konstantinos G. Derpanis Overview of the RANSAC Algorithm Retrieved from: http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf
3. G. Lukacs, R. Martin, and A.D. Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In ECCV’98, 5th European Conf. on Computer Vision, volume 1 of LNCS 1406, pages 671 – 686, Freiburg, 1998
4. R.C. Bolles and M.A. Fischler. A RANSAC based approach to model fitting and its application to finding cylinders in range data. In IJCAI, Int. Joint. Conf. on Artificial Intelligence, pages 637–643, Vancouver, Canada, 1981
5. T. Lozano-Perez, W.E.L.G. Grimson, and S.J. White. Finding cylinders in range data. In Int. Conf. on Robotics and Automation, pages
6. Hugues Hoppe, Tone DeRose, Tom Ducamp, Jon McDonald and Werner Stuetzle. Surface Reconstruction from Unorganized Points. Retrieved from: http://hhoppe.com/recon.pdf