[1]一开始我们尝试使用大圆柱的轴当作测量部份圆柱的轴，但是其实由于加工误差这两部分的轴并不完全一致，这个很小的轴向偏差最终影响很大，所以最终还是只是用测量部份进行轴的估计。

基于激光轮廓扫描的多段圆柱工件半径测量方法研究

摘要:在制造业中，多段圆柱形工件的半径精确测量至关重要。传统激光轮廓扫描方法常受分割不准确、噪声高和误差大的困扰。为此，本文提出一种半自动分割与稳健圆柱拟合方法，有效提升测量精度。该方法首先在用户界面进行初步分割，将测量工件点云分割出来，确定测量区域的半径范围。接着，采用 B 样条曲线对原始测点进行迭代平滑，计算平滑后曲线的切线斜率，并转换为二次函数进行分割。对每条点云线划分分割区域后，通过聚类筛选得到最终的测量区域。在测量区域确定后，利用 RANSC 算法识别圆柱轴线，并将点云线投影至轴线。随后，对每条投影后的点云线应用二维 RANSC 算法去噪，并采用 Hyper fit 算法拟合圆柱半径。最后，通过去除半径序列中的异常值，分析得出各段圆柱的精确半径。

关键词：多段圆柱工件，半径测量，B 样条迭代平滑，RANSC 算法，Hyper fit 算法

1 引言

1.1 背景

暂略。

1.2 一般方程

1.2.1 圆柱方程

如果只考虑单个圆柱体，我们使用点到直线距离方程推导来推导出其圆柱方程。轴线r可以表示为

$$r' = p + tv \tag{1.1}$$

其中$p_0 = (p_x, p_y, p_z)$是轴上一点，$v = (v_x, v_y, v_z)$是方向向量， 点到直线的距离可以表示为

$$\mathrm{dis}(p, r', p_0) = \frac{\|(p - p_0) \times v\|}{\|v\|} \tag{1.2}$$

所以圆柱的方程可以表示为到轴线的距离为

$$\mathrm{dis}(p, r', p_o) = r \tag{1.3}$$

代入$p = (x, y, z)$化简可得

$$\begin{aligned}&\big[(y - p_y)v_z - (z - p_z)v_y\big]^2 + [(z - p_z)v_x - (x - p_x)v_z]^2 + \\ &\big[(x - p_x)v_y - (y - p_y)v_x\big]^2 = r^2(v_x^2 + v_y^2 + v_z^2)\end{aligned} \tag{1.4}$$

化简后另一种表达方式为

$$(x-x_0)^2+(y-y_0)^2+(z-z_0)^2- \\ \frac{((x-x_0)v_1+(y-y_0)v_2+(z-z_0)v_3)^2}{v_1^2+v_2^2+v_3^2}=r^2 \tag{1.5}$$

或者圆柱还可以用[1] 中的方程表示。

1.2.2 圆的方程

$$(x-a)^2+(y-b)^2=r^2 \tag{1.6}$$

$$A(x^2+y^2)+Bx+Cy+D=0 \ \text{ st } \ B^2+C^2-4A>0 \tag{1.7}$$

上述 式(1.7) 中的约束其实完全可以写作

$$B^2+C^2-4A=1 \tag{1.8}$$

完全成立。

1.3 轮廓仪

1.3.1 轮廓仪的标定问题

对于激光轮廓扫描仪，如果未进行标定，会在 y 轴存在一个缩放系数。假使底面真实移动速度为V_{true},轴$v'=\left(v'_x,v'_y,v'_z\right)\ \&\ v'^2_x+v'^2_y+v'z^2=1$ 也就是$y=\beta y'$

$$V_{y'}=v_yV_{\text{true}} \tag{1.9}$$

后面的计算在经过标定v'后直接省略掉了这个缩放系数。下面给出仿真中有无β的 oyz 截面

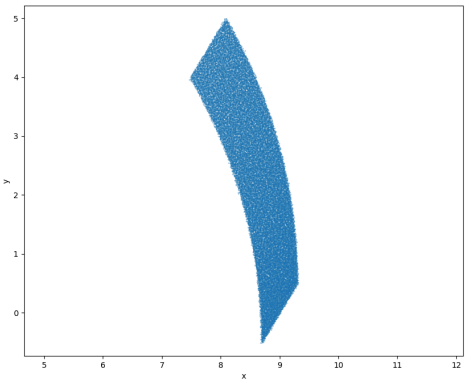

图 1.1: $(0.99,0.03,0.05),\beta=1,$ oyz

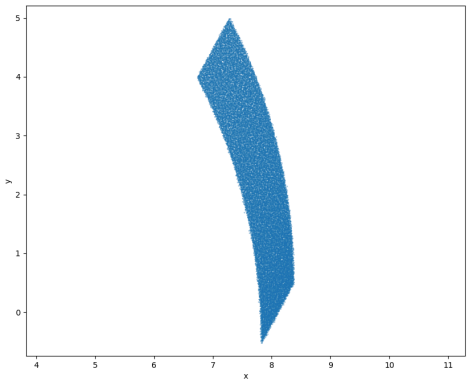

图 1.2: $(0.99,0.03,0.05),\beta=0.9,$ oyz

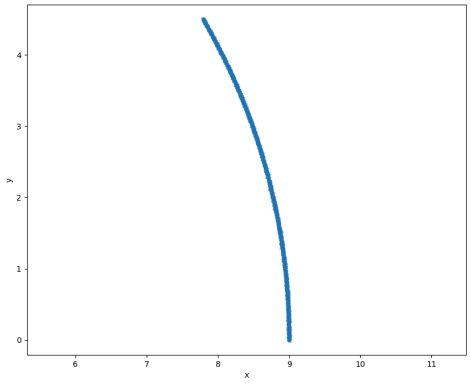

图 1.3: $(0.99, 0.09, 0.05), \beta = 1$, 轴面

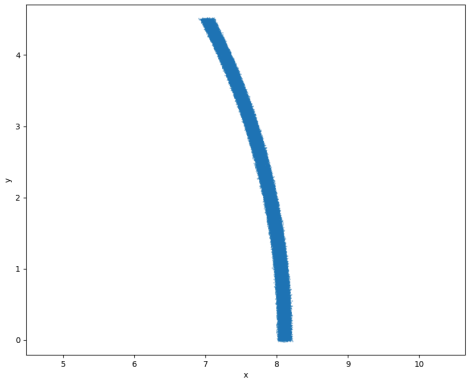

图 1.4: $(0.99, 0.09, 0.05), \beta = 0.9$, 轴面

1.3.2 参数调节

曝光参数会影响最终点云的质量,如果曝光增益调的太高,则测量部分中间会出现空洞,太低则效果不好,目前参数是

曝光	增益
200	0

ROI 直接设置为 FULL 模式

然后接下来是一键调参的参数,设置

移动速度(mm/ s)	x 轴精度	y 轴精度	行程长度(mm)
2	50	5	20

然后控制步进电机根据工件上激光痕迹将要测量区域周围测量完毕.

2 轴拟合方法/圆柱拟合[2]

根据[2] 3-D 拟合技术产生的结果比 2-D 投影差大约 25%。3-D 拟合很可能对噪声敏感,而 2-D 投影可以更好地平均化噪声,因为圆上的点密度高于圆柱上的点密度。

2.1 相关研究

现有的圆柱拟合方法主要集中在完整数据上，即假设圆柱物体被完全扫描。目前很少有针对受异常值污染的不完整点云数据中的圆柱拟合进行全面研究的文献[3] 。

[2]一开始我们尝试使用大圆柱的轴当作测量部份圆柱的轴，但是其实由于加工误差这两部分的轴并不完全一致，这个很小的轴向偏差最终影响很大，所以最终还是只是用测量部份进行轴的估计。

大多数用于几何基元（例如平面和圆柱）提取和拟合的方法都是基于最小二乘法（LS）和/或奇异值分解（SVD），这两种方法都对异常值敏感，并且在统计上不具备稳健性。

众所周知的RANdom SAmple Consensus (RANSAC)方法已被用于在存在异常值的情况下进行稳健的模型拟合。Bolles 和 Fischler 在[4] 中首次提出了使用 RANSAC 进行圆柱拟合的工作(使用最小二乘) 。后来 改进了 RANSC 算法,引入了 5-9 个点的直接代数解法.[3] 在后来的阶段，表面拟合方法被用于在 3D 数据中寻找圆柱[5] 。

(多年来, 许多修订版的RANSAC 已被开发出来。[6] 考虑RANSAC（Fischler 和 Bolles, 1981）进行比较，因为其众所周知，并且许多人（参见 Wang 等人，2016）将其作为首选，考虑到其高质量的结果、稳健性和通用性。)

Faber 和 Fisher[7] 通过约束欧几里得拟合方法解决了 3D 数据圆柱拟合的问题, 并声称他们的方法比更常用的代数拟合方法产生更好的结果。Wang 和 Suter[8] 成功地使用了稳健回归, 并结合[9] 中定义的对称距离, 用于在 2D 图像数据中进行稳健圆和椭圆拟合。然而, 指出，他们的方法并不适用于空间不对称（即**部分**）数据。

Lalonde 等人[2] 沿用了 Kwon 等人使用主成分分析（PCA）进行圆柱拟的想法，并将其应用于树干估计。基于圆拟合的方法, 包括[2], 已被许多圆柱拟合方法采用。作者[2] 展示了他们基于 PCA 的方法比其他方法显著更快且更准确，但他们没有考虑异常值的存在。

Abdul 等[3] 针对离群点的敏感性和圆柱采样不完全的问题, 提出并比较了两种稳健的圆柱拟合方。新提出的两种算法使用了稳健主成分分析（RPCA）、稳健回归。

Chaperon 和 Goulette（2001）提出了一种两步方法；第一步在 Gaussian 球面上使用 RANSAC 来找到圆柱的方向，而第二步则找到圆柱的位置和大小。

Beder 和 Förstner（2006）使用 RANSAC 在原始点云上获得直接解决方案, 用于估计圆柱参数。Vosselman 等人（2004）引入了一种基于 HT 的圆柱检测算法，该算法**使用数据点的法**来获得圆柱方向。

2.1.1 PCA

PCA 主要是受到[3,6] 这两篇文章, 但其实这两篇文章均使用的是 RPCA 得到了较好的效果，但是在我们的仿真中 RPCA 并没有得到较好的效果。

下面简单推导一下二维的 PCA

从最小点集中提取圆柱体和圆锥体 一种直接的代数求解方法 先推导二维的 PCA

计算一维数据的方差

[3]RANSAC 并不能完全摆脱异常值的影响 根据[3]

$$s^2(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{2.10}$$

对于二维数据有协方差

$$\text{cov}\ (X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{2.11}$$

将数据进行归一化后协方差矩阵 C 其实是$\left(X_1 = X - \bar{X}\right)$

$$C = \frac{1}{n-1}X_1^T X_1 \tag{2.12}$$

定义一个投影方向$v = (x_0, y_0)\ \text{st}\ \ x_0^2 + y_0^2 = 1$

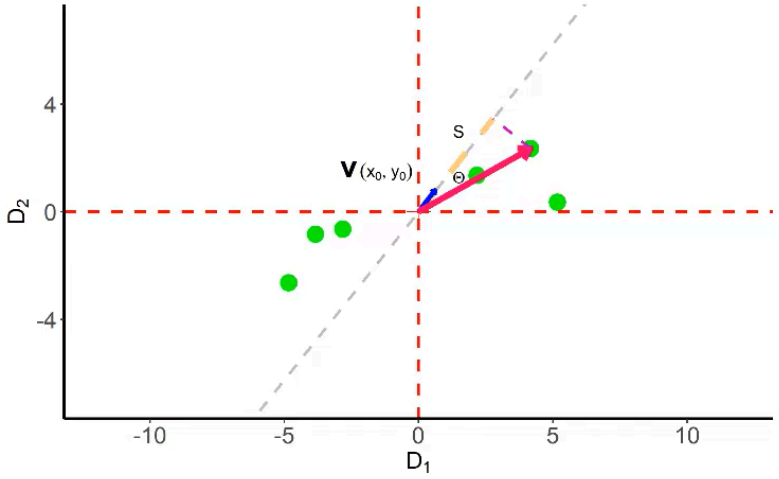

图 2.5: 示意图

$$S = \vec{v}\vec{x} = |\vec{x}|\cos\theta \tag{2.13}$$

在这个方向投影后的长度的方差其实可以表示为

$$\frac{1}{n-1}\sum S^2 = \frac{1}{n-1}(\vec{v}X_1^T)(\vec{v}X_1^T)^T = \vec{v}\frac{X_1^T X_1}{n-1}\vec{v}^T = \vec{v}C\vec{v}^T \tag{2.14}$$

最优化方差最小值

$$J = \vec{v}C\vec{v}^T\ \ \text{st}\ \ \vec{v}\vec{v}^T = 1 \tag{2.15}$$

等式约束的最优化问题，引入拉格朗日函数

$$F(\vec{v}) = \vec{v}C\vec{v}^T - \lambda(1 - \vec{v}\vec{v}^T) \tag{2.16}$$

$$\frac{\partial}{\partial v}F(\vec{v}) = 0 \Rightarrow 2Cv^T - 2\lambda v^T = 0 \qquad \left(\frac{\partial}{\partial v}f(x)v = f(x)^T, \frac{\partial}{\partial v}A^T vA = Av + A^T v\right), (2.17)C^T$$

$$Cv^T = \lambda v \tag{2.18}$$

对 C 做特征值分解，得到两个特征向量 PC1,PC2 对应沿轴方差最大和最小。

对于三维的话，我们思考一个圆柱，当其长度远远大于半径的时候，并且圆心角不是特别小，投影方差最大的方向应该是轴向，之前论文里面也是如此。

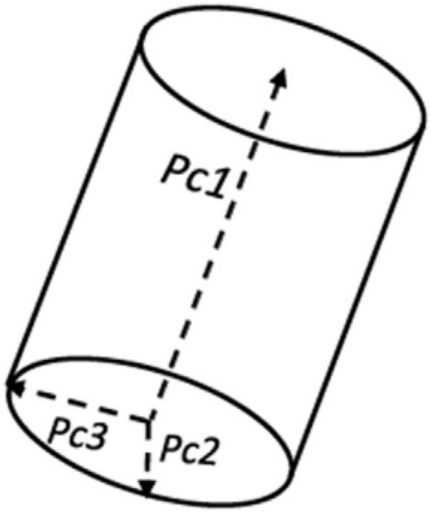

图 2.6: 主成分（PC1、PC2 和 PC3）用于定义圆柱方向的示意图；PC1 一般指向圆柱的轴线方向，而 PC2 和 PC3 则编码圆柱宽度垂直方向上的数据变异性。

但是如果对于只有部分点云的时候, 并不是如此, 但是三个特征值中必有一个是轴向（根据实验可得），所以向特征值方向重投影，计算轮廓面积，取最小的即为轴向。

仿真里面 PCA 的效果很好，但是在实际点云上效果并不佳。继续尝试鲁棒 PCA 后效果也不好, 可能是圆柱圆心角过小。之前查阅过的论文里面使用 PCA 方法寻找轴都是对于完整的圆柱点云, 如树木, 管道等, 其更容易求解。但是更有可能是缩放系数β干扰了 PCA, 使得我们的实际点云并不是严格的圆柱, 根本无法使用 PCA 求解, 因为原论文中和我们的仿真中都对于小圆心角进行了测试。也有可能是采样点过多，论文中的点都比较稀疏。

这部分仿真代码在 `Simulation/PcaTest` 。PCA 相关算法封装在了 `algorithm/PcaAxis`，鲁棒 PCA 部分使用了 `robpy` 库，其移植了一系列统计相关的方法。

TODO: RPCA（鲁棒 PCA）部分可能因为库中的实现方法和论文中不一样导致效果不是很好，可以重新修改一下，主要加入了 MCD 算法，这个在 `robpy` 库中也有实现[4]。

下面是实验结果

$$v_{\text{true}} = (0.99463613, 0.09042056, 0.05023031), \theta = 30, r_{\text{true}} = 9, \text{len} = 20 \quad (2.19)$$

[4]对于 1000 个模拟的四分之一圆柱，其中包含 10% 的聚集离群点，基于主成分分析（PCA）的方法 Lalonde 拟合圆柱的平均方向误差 θ 为 5.87 度，而基于新 RPCA 算法的拟合圆柱的平均方向误差仅为 0.36 度。

$$v_{\text{pca}} = (0.994637140.090413360.05022337) \tag{2.20}$$

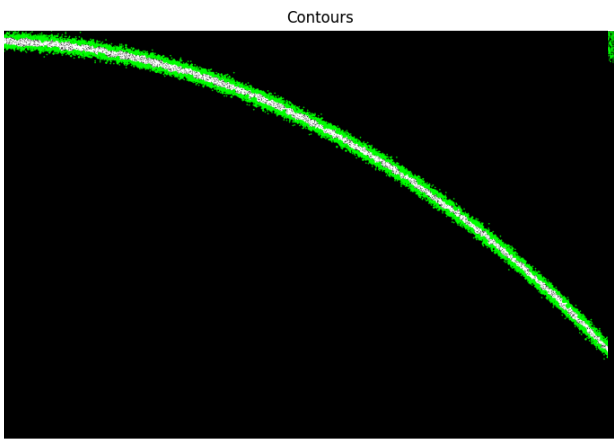

图 2.7: PCA1 轴面投影

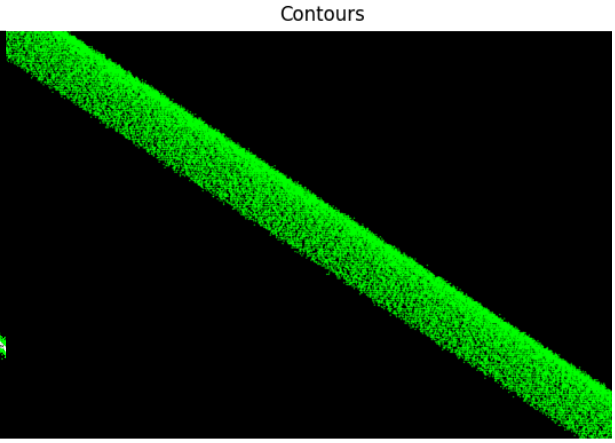

图 2.8: PCA2 轴面投影

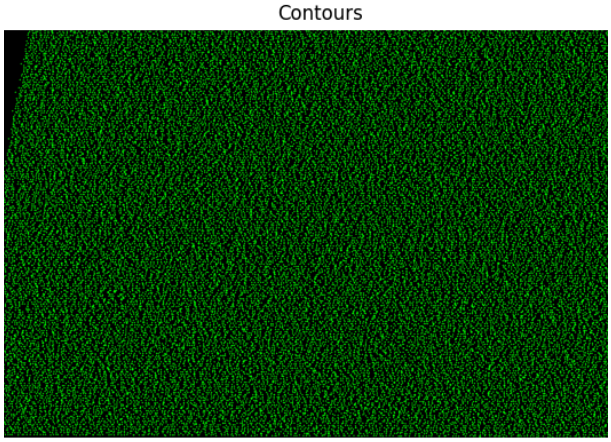

图 2.9: PCA3 轴面投影

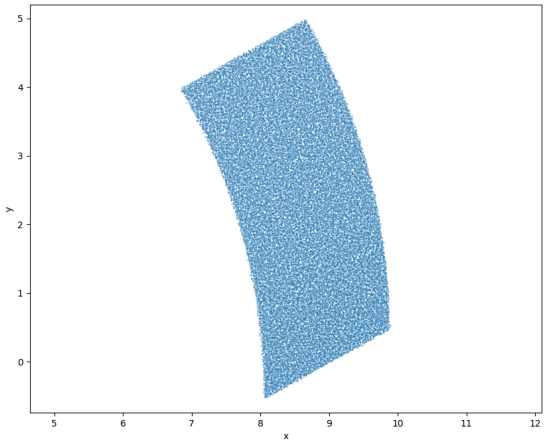

图 2.10: 原 oyz 面投影

$v_{\text{true}} = (0.891710, 0.450359, 0.045035), \theta = 10, r_{\text{true}} = 9, \text{len} = 10 \text{ noise} = 0.01$

$$v_{\text{pca}} = (0.8917198, 0.45034084, 0.04504355) \tag{2.21}$$

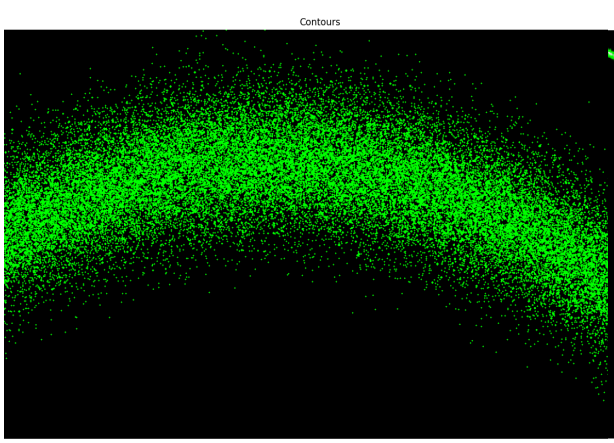

图 2.11: PCA1 轴面投影

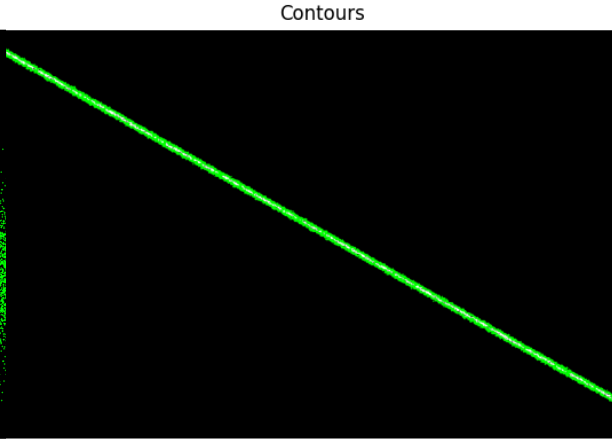

图 2.12: PCA2 轴面投影

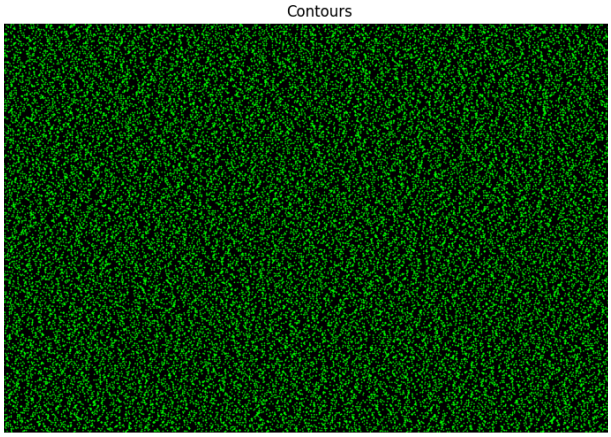

图 2.13: PCA3 轴面投影

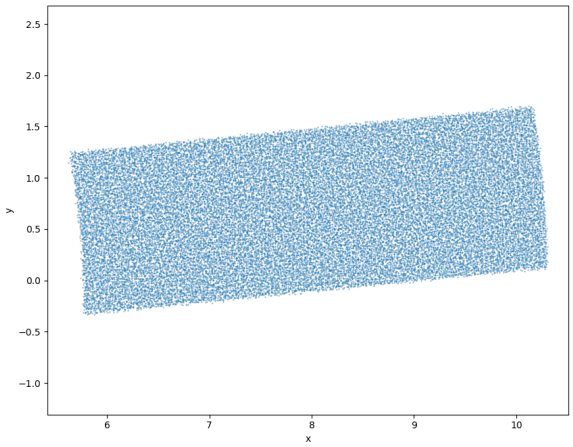

图 2.14: 原 oyz 面投影

$v_{\text{true}} = (0.891710, 0.450359, 0.045035), \theta = 10, r_{\text{true}} = 9, \text{len} = 10\beta = 0.9$

$$v_{\text{pca}} = (0.90947708, 0.41307121, 0.04715522) \tag{2.22}$$

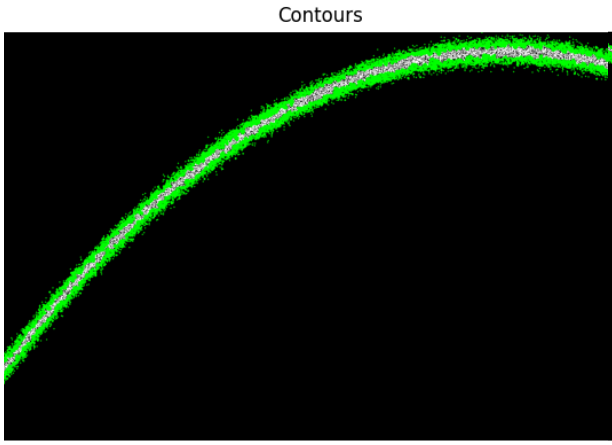

图 2.15: PCA1 轴面投影

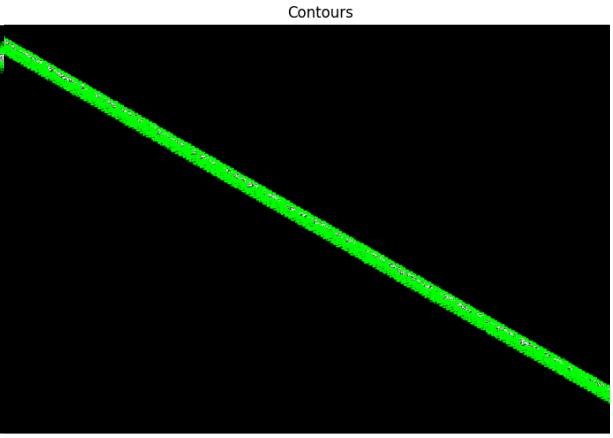

图 2.16: PCA2 轴面投影

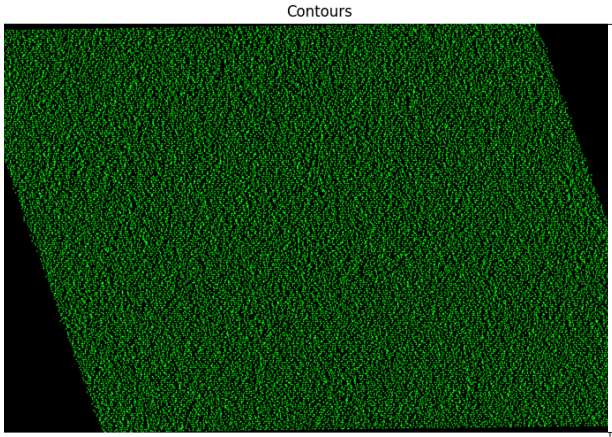

图 2.17: PCA3 轴面投影

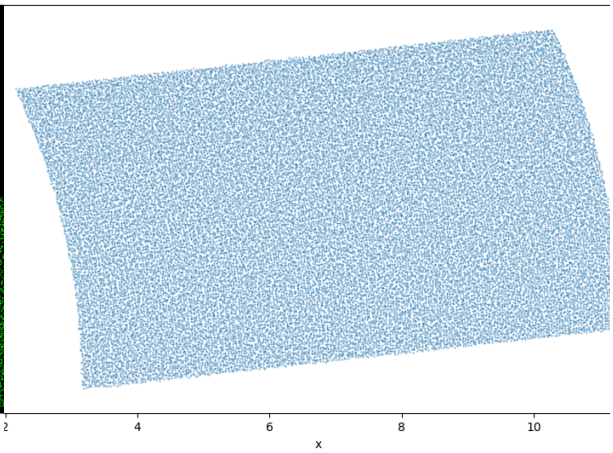

图 2.18: 原 oyz 面投影

在**实际的点云**上求解的结果如下（仅 PCA）

2.1.2 RPCA

RPCA 方法（[10] Hubert 等人，2005），该方法适用于低维（3D）点云处理（[3,6]）。

如果一个点是异常值, 那么必定存在某个一维投影使该点也是一元异常值。因此, 导出了一个“异常性”度量，通过将数据值投影到多个一维方向上来识别异常值

在每个方向上, 每个点根据其作为异常值的可能性进行评分, 其中第 i 个点在方向 v 上的异常性计算为

$$w_i = argmax_v \frac{\left|p_i v^T - \mu_{\mathrm{FMCD}}(p_i v^T)\right|}{\sum_{\mathrm{FMCD}}(p_i v^T)} \tag{2.23}$$

$\mu_{\mathrm{FMCD}}(p_i v^T)$是基于FMCD的稳健均值, $\sum_{\mathrm{FMCD}}(p_i v^T)$是基于FMCD的稳健协方差矩阵

通过将数据投影到多个单变量方向上来计算。对于每个方向，计算投影点的基于快速最小协方差行列式（FMCD；Rousuw 和 van Driessen，1999）的稳健中心 μFMCD 和协方差矩阵 ΣFMCD，如公式（1）所示。接下来，使用具有最小 wi 值的多数份额（$h > \frac{n}{2}$）的点来构建一个稳健的协方差矩阵，该矩阵随后用于主成分分析（PCA）并获得稳健的主成分。Hubert 等人（2005）声称，所得的稳健主成分具有位置和正交不变性，他们的方法对无异常值的数据产生准确的估计，对有异常值的数据产生更稳健的估计。

2.1.3 使用椭圆方程拟合

但最终问题也是圆心角过小，并且我们的噪声有的过大，拟合椭圆误差很大。

2.1.4 RANSC

这里直接使用了 PCA 的 RANSC 算法[5]，使用 pyblind11 库进行了封装，便于 python 调用。

步骤主要是先对点云构建 kdtree，计算法线。(这里需要注意不同的点云密度需要设置不同的求解法线时候使用的周围点数量)，然后从点集中随机选取若干点，计算圆柱参数，划分内点，计算误差，满足时候退出。

实现有一个这个库[11]

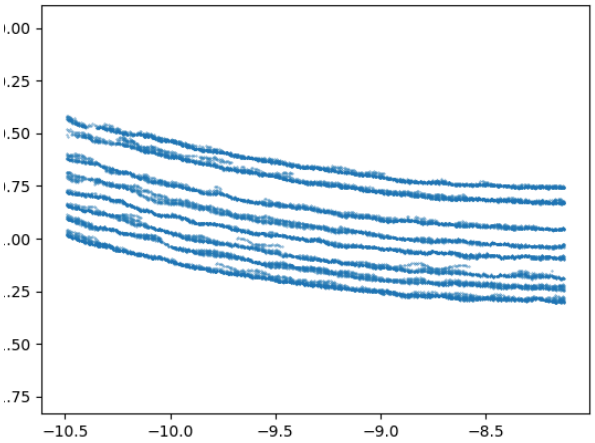

图 2.19: 原始 oyz 平面投影

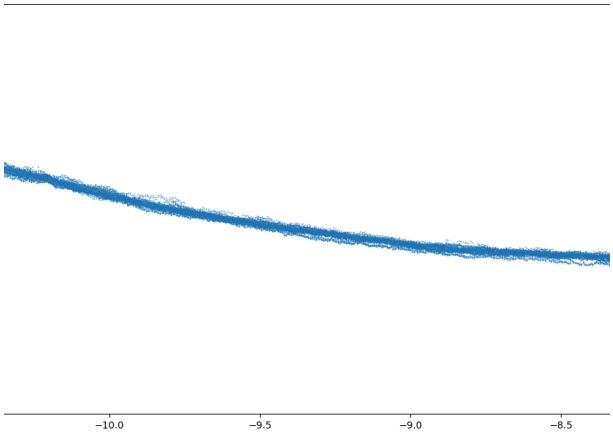

图 2.20: 沿 RANSC 轴线投影

[5]RANdom SAmple Consensus（随机抽样一致）

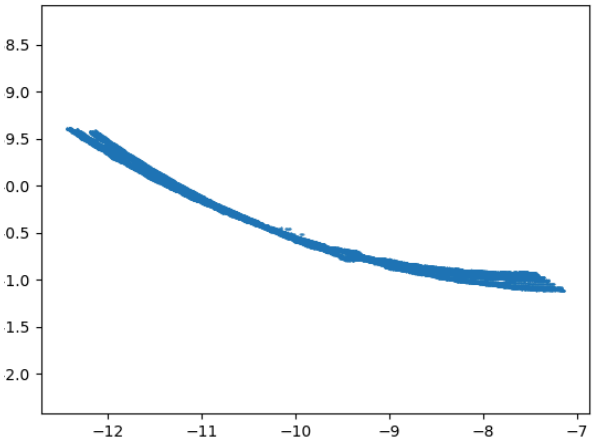

图 2.21: 使用 PCL 的投影

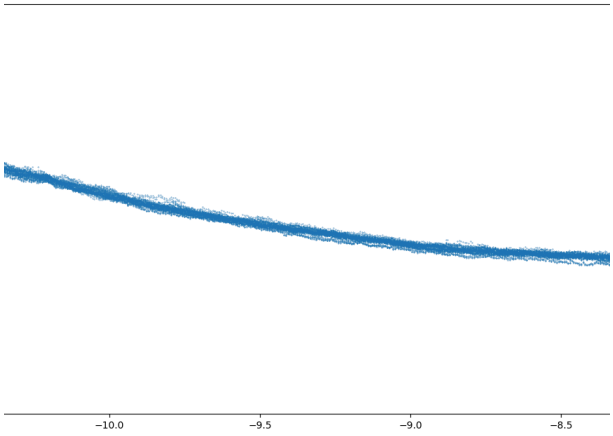

图 2.22: 沿 RANSC 轴线投影

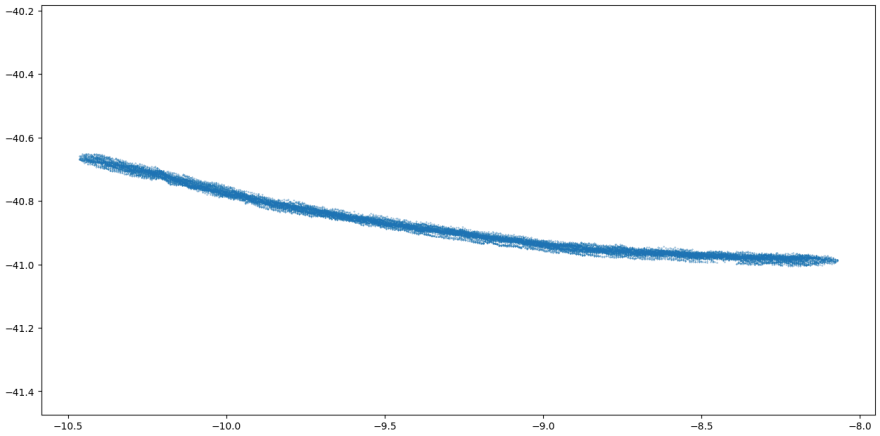

图 2.23: 对投影后点云进行半径滤波 $\mathrm{std} = 0.02, n = 100$

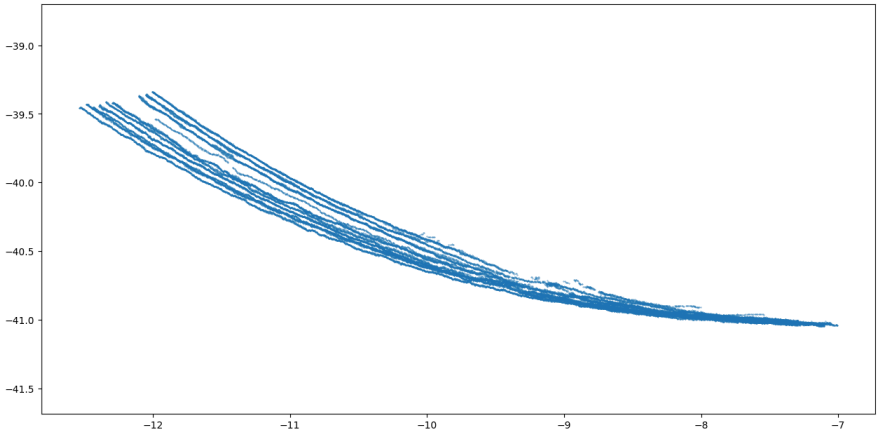

图 2.24: 使用不带 MCD 的 RobuPCA 投影面

代码

封装在 `Pyblind11Ransc` 包中，使用示例如下

```
import PclRansc
NormalDistanceWeight = 0.5
max_iterations = 10000
distance_threshold = 0.1
radius_min = 5
radius_max = 12
k=60
axis1=PclRansc.find_cylinder_axis(
    PointCloud.points, NormalDistanceWeight,
    max_iterations, distance_threshold, radius_min, radius_max,k
)
```

其中`axis1[0-2]`表示中心，`axis1[3-5]`表示方向，`axis1[6]`表示半径。`axis1[7]`表示内点。这里虽然有一部分噪点被滤除了，但是可以看到二维图中还是有一些点没有被滤除，所以又使用了一次半径滤波。

Nurunnabi 等（2014，2015）指出，RANSAC 受到了众所周知的掩蔽效应的影响（参见 Rousseeuw 和 Leroy，2003）[6]。

2.2 误差分析

参考[6]

为了评估拟合圆柱体的参数 C、R、L 和 O，我们计算真实圆柱体中心与拟合圆柱体中心之间的平均距离 AD(C)，估计的平均半径 A(R)，平均长度 A(L)，以及平均偏差方向 A(θ)的多个样本值。这些指标定义如下：

$$\mathrm{AD}(C) = \frac{1}{m-1}\sum |C_i - \widehat{C}_i|$$

$$A(R) = \frac{1}{m-1}\sum \widehat{R}_i \tag{2.24}$$

$$A(\theta) = \frac{1}{m-1}\sum \hat{\theta}$$

$$\theta = \arccos|v_2^T \hat{v}_2| \tag{2.25}$$

其中v_2是真实轴线，$\hat{v}_2$是估计轴线。

提出的圆柱拟合算法在多个人造和真实的点云上进行了演和评估。从数据完整性、稳健性、一致性以及圆柱半径的大小方面评估了新算法的性能，并将结果与三种现有方法进行了比较：（i）最小二乘法，（ii）基于主成分分析的方法（Lalonde et al., 2006），以下称为 Lalonde 方法，以及（c）RANSAC 方法。

Methods	Full cylinder				Half cylinder				Quarter cylinder			
	$\mathrm{AD}(\hat{C})$	$\mathrm{A}(\hat{R})$	$\mathrm{A}(\hat{L})$	$\mathrm{A}(\theta)$	$\mathrm{AD}(\hat{C})$	$\mathrm{A}(\hat{R})$	$\mathrm{A}(\hat{L})$	$\mathrm{A}(\theta)$	$\mathrm{AD}(\hat{C})$	$\mathrm{A}(\hat{R})$	$\mathrm{A}(\hat{L})$	A(
LS	0.55	1.17	13.20	8.78	1.85	1.34	12.79	1.16	3.76	2.87	12.73	1.4
Lalonde	0.75	1.31	13.16	7.64	1.41	1.10	12.98	5.64	4.50	3.63	12.99	5.8
RANSAC	0.40	1.01	12.80	0.96	0.42	1.00	12.80	0.98	0.69	1.19	12.77	1.4
RLTS	0.08	1.00	10.06	0.25	0.09	1.00	10.07	0.31	0.12	1.02	10.07	0.3

Table 1. Evaluation of estimated parameters for full, half and quarter cylinders
: Average Distance between the real cylinder centres and the fitted cylinder centres, $\mathrm{A}(\hat{R})$ = estimated Aver
stimated Average length, and $A(\theta)$ = Average bias orientation]

图 2.25: 论文[6] 中实验的结果

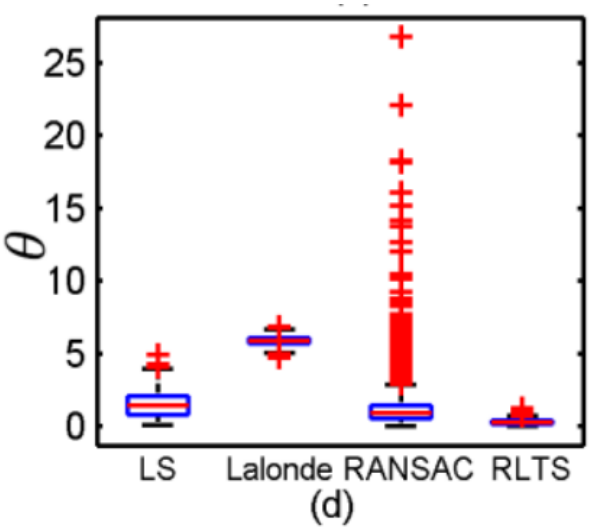

图 2.26: 论文[6] 中实验的结果（包含 10% 的集群异常）

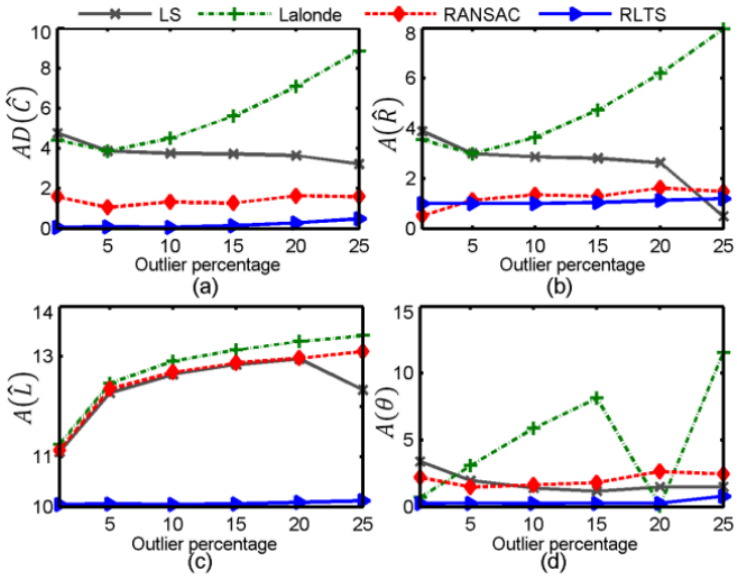

Figure 4. Influence of different percentages (1%,5%,10%,15%, 20% and 25%) of clustered outliers on cylinder fitting methods; line diagrams for outlier percentages versus performance measures (a) AD($\hat{C}$), (b) A($\hat{R}$), (c) A($\hat{L}$), and (d) A(θ)

图 2.27: 论文[6] 中实验的结果

Methods	n=100				n=1,000				n=10,000			
	AD($\hat{C}$)	A($\hat{R}$)	A($\hat{L}$)	A(θ)	AD($\hat{C}$)	A($\hat{R}$)	A($\hat{L}$)	A(θ)	AD($\hat{C}$)	A($\hat{R}$)	A($\hat{L}$)	A(θ)
LS	3.88	2.98	11.16	2.60	3.77	2.88	12.67	1.52	3.74	2.85	13.87	1.37
Lalonde	4.66	3.77	11.42	5.90	4.52	3.65	12.93	5.89	4.49	3.61	14.14	5.84
RANSAC	0.85	1.27	11.22	1.63	0.73	1.21	12.71	1.37	0.59	1.14	13.92	1.26
RLTS	0.37	1.11	9.80	0.82	0.09	1.01	10.06	0.24	0.03	1.00	10.02	0.09

Table 2. Evaluation of estimated parameters for cylinders of different number of points
AD($\hat{C}$) = Average Distance between the real cylinder centres and the fitted cylinder centres, A($\hat{R}$) = estimated Average radi
A($\hat{L}$) = estimated Average length, and $A(\theta)$ = Average bias orientation]

Methods	R = 0.05				R = 0.10				R = 0.50			
	AD($\hat{C}$)	A($\hat{R}$)	A($\hat{L}$)	A(θ)	AD($\hat{C}$)	A($\hat{R}$)	A($\hat{L}$)	A(θ)	AD($\hat{C}$)	A($\hat{R}$)	A($\hat{L}$)	A(θ)
LS	624.16	553.22	12.16	14.78	592.08	529.96	11.96	16.01	3.18	2.39	12.16	9.36
Lalonde	390.46	390.45	13.11	7.19	630.54	630.52	13.09	7.08	7.68	7.28	13.00	6.46
RANSAC	119.21	2.20	12.83	2.19	6.06	3.71	12.76	2.92	1.19	1.01	12.75	1.55
RLTS	0.09	0.06	10.05	0.04	0.14	0.09	10.08	0.04	0.10	0.51	10.06	0.14

Table 3. Evaluation of estimated parameters for cylinders with different radii (in metre)
AD($\hat{C}$) = Average Distance between the real cylinder centres and the fitted cylinder centres, A($\hat{R}$) = estimated Average radi
A($\hat{L}$) = estimated Average length, and $A(\theta)$ = Average bias orientation]

图 2.28: 论文[6] 中实验的结果

3 圆拟合方法

在选取测量区域后, 我们先将每根线（x相同的点）投影到以圆柱轴为法线的平面上, 然后接下来对于每条线X_i, Y_i,进行一个圆拟合问题。

圆拟合方法主要包括两类：（i）几何拟合法，和（ii）代数拟合法。

第一类方法最小化几何距离, 而第二类方法最小化代数函数。几何拟合计算量大, 而代数拟合速度更快, 因为它们不像几何拟合那样需要迭代。我们强调更快的方法, 因为我们处理的是大型点云数据 (PCD)。下文中我们将不考虑几何圆拟合, 只考虑代数圆拟合

3.1 去噪

我们拿到的数据一定程度上是具有噪声的, 对于每一条

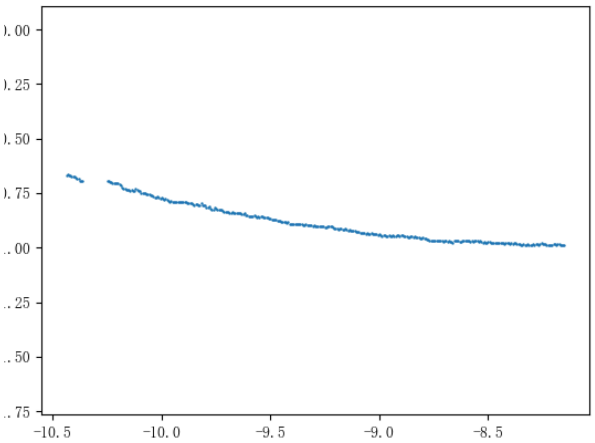

图 3.29: 噪点图像 1

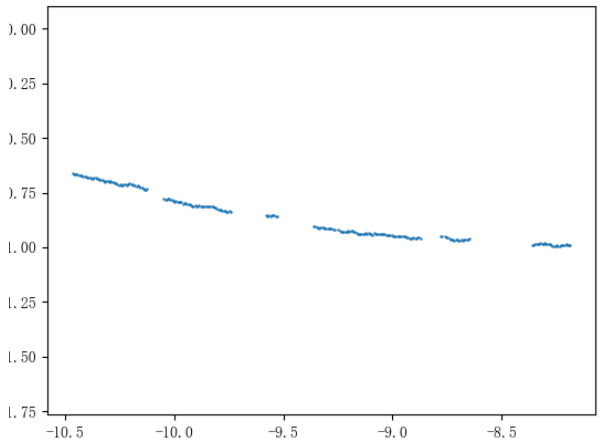

图 3.30: 噪点图像 2

所以我们必须要对噪点进行剔除, 或者直接剔除该条数据。上述这种噪点, 有的因为拟合值偏差较大会被剔除, 但是还是有绝大多数保留了下来对最终的测量结果造成影响。

3.1.1 RANSC 去除噪点

下面先仿真一下并且撰写 RANSC 去噪算法

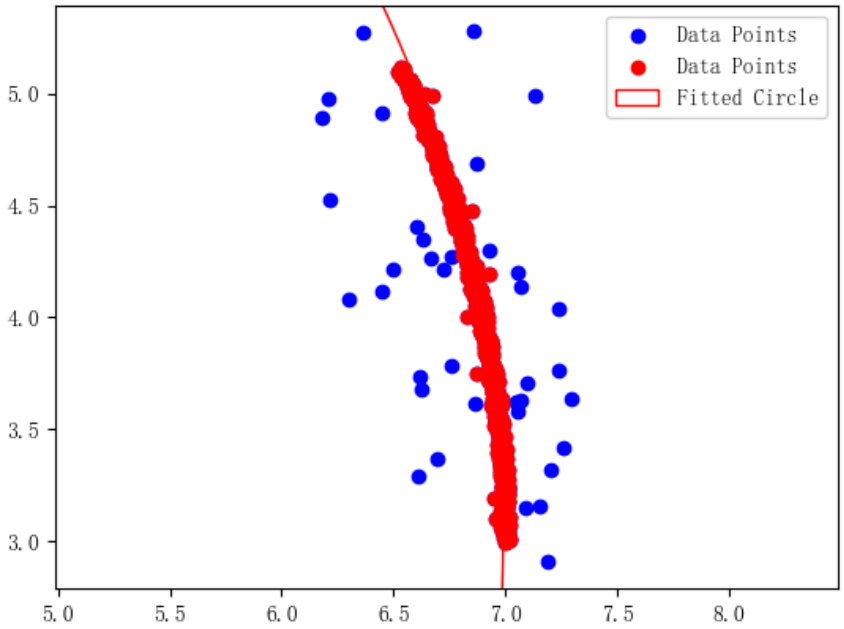

图 3.31: RANSC 算法仿真 $r_{\text{true}} = 5, r_{\text{ransc}} = 5.69, r_{\text{final}} = 5.14$

实际上还可以调节参数使得结果更加真实。我们将这个算法用于实际点云上, 效果如下:

3.1.2 LTS 回归

参考[6] 文中的做法， 我们将 LTS 回归原理与超圆拟合方法结合。使用 LTS 回归，新算法修剪掉残差平方和最大的(n − h)个点，并用最接近圆弧的一致的 h (h ≥ n/2)个点进行圆拟合。

3.2 代数方法

[12] 这篇文章讲述了很多方法，同时也给出了误差分析。[13] 这里是使用 Matlab 实现的各种圆拟合算法, 可以很简单的转为 python。也有人实现了上述各个算法的 python 版本, 使用 circle-fit 包。 拟合圆的标准方程为:

$$x^2 + y^2 + Ax + By + C = 0 \tag{3.26}$$

目标是从点云数据 (x_i, y_i) 中找到最优参数 (A)、(B)、(C)， 以使得这些点尽可能满足上述方程。 将方程写成矩阵形式:

$$(x^2 + y^2) + A \cdot x + B \cdot y + C = 0 \tag{3.27}$$

定义设计矩阵 A_{design}:

$$A_{design} = \begin{bmatrix} x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ x_2^2 + y_2^2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^2 + y_n^2 & x_n & y_n & 1 \end{bmatrix} \tag{3.28}$$

定义参数向量 $P = [A, B, C, D]^T$ ，其中 (D) 表示偏置项。

然后问题可以表示为:

$$A_{design} \cdot \mathbf{P} = 0 \tag{3.29}$$

为了非平凡解 $\mathbf{P}$，需要约束 $\mathbf{P}$ 的长度。这里使用二次约束条件：[6]

$$P^T Q P = 1 \tag{3.30}$$

其中 Q 是二次约束矩阵，用来保证拟合圆弧的几何意义:

$$Q = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix} \tag{3.31}$$

最终问题转化为一个广义特征值问题:

A_{design}简称为A_d 问题转换为

$$\text{minimize } \|A_d P\|^2 = P^T A_d^T A_d P \quad \text{st.} \quad P^T Q P = 1 \tag{3.32}$$

构建拉格朗日函数

$$G(P) = P^T A_d^T A_d P - \lambda(P^T Q P - 1) \tag{3.33}$$

应用矩阵求导法则 $\frac{\partial}{\partial P} P^T A P = A^T P + A P$

$$\frac{\partial}{\partial P} G(P) = 2 A_d^T A P - \lambda(Q^T P + Q P) = 0 \Rightarrow A_d^T A_d P = \lambda Q P \tag{3.34}$$

$$\begin{aligned} Q^{-1} A_d^T A_d P &= \lambda p \\ M P &= \lambda P \end{aligned} \tag{3.35}$$

求解 M 的特征值,求解该广义特征值问题, 得到所有特征值 λ¥ 和对应的特征向量 P。特征值对应的是不同解的代价，选择最小非负特征值对应的特征向量作为解P。

通过变形可以得出圆心和半径:

$$(h, k) : h = -\frac{A}{2}, k = -\frac{B}{2} \tag{3.36}$$

[6]由于参数 (A, B, C, D) 只需确定到标量倍数，因此施加一个约束是合理的。

$$r : r = \sqrt{\frac{A^2 + B^2 - 4C}{4}} \tag{3.37}$$

这种方法结合了最小二乘和约束优化，能够较好地拟合圆弧，特别适合包含部分圆弧点的情形。

该算法封装在 `algorithm/PreciseArcFitting` 中，使用示例如下：

```
from algorithm.PreciseArcFitting import CircleArc
arc=CircleArc()
points=...
arc.fit_circle_arc(np.array(points))
center = arc.first_param[0:2]
radius = arc.first_param[2]
```

3.2.1 对约束矩阵的拓展

当约束矩阵为

$$Q = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix} \tag{3.38}$$

这种方法叫做 **Pratt fit**.

$$Q = \begin{bmatrix} 4\bar{z} & 2\bar{x} & 2\bar{y} & 0 \\ 2\bar{x} & 1 & 0 & 0 \\ 2\bar{y} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.39}$$

这种方法加 **Taubin fit**.

$$Q = \begin{bmatrix} 8\bar{z} & 4\bar{x} & 4\bar{y} & 2 \\ 4\bar{x} & 1 & 0 & 0 \\ 4\bar{y} & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} \tag{3.40}$$

这种方法加 **Hyper fit**.[7]

3.2.2 adam 优化的梯度下降法

主要参考的是[14]，这篇文章。虽然作者自己说

[7]我们的误差分析引出了另一个惊人的发现——一种完全没有本质偏差的代数拟合。据我们所知，这是曲线拟合问题中此类算法的首次出现[12]。

Finally, simulation and experimental results show that our ICF method is more robust and high-precision than TLSF and Hyper method, so it is very competent to measure short arcs with noise even their central angles are **close to 5°**.

但是我们仿真测试下来并非如此。

3.2.3 优化的 L-M 方法

上一篇文章的参考文献中提到的方法，计算量较大。

3.2.4 使用半径或者圆心约束或者其他几何约束的最优化

暂时没有尝试，因为缺少一般性质，我们不只要测量一个工件。

3.3 其他优化方法

[6] 也是使用 Hyper，但他使用了迭代的方法。

3.4 误差分析

直线和圆拟合的精度并不依赖于线或圆的厚度[15] 。

根据[12] 代数拟合速度更快，但通常精度较低。同时，对它们准确性的评估仅基于实际经验，尚未有人对各种圆拟合的精度进行详细的理论比较。

同时这篇文章也介绍了一种超精确拟合，跟我们第一个推到的类似，只是约束矩阵变为

4 去噪和测量区域选取

4.1 三维深度方法

使用 PointNet++ 选取测量部分

4.2 三维配准方法

先手动选取部分后，将该点云和测量点云做点云配准，精配准后使用固定的部分提取测量部份。

4.3 二维方法

先对每一根线做 B 样条优化平滑，考虑圆弧切线方程 正常一个圆弧的切线方程（只考虑上半部分，应该是）

$$m(x) = -\frac{x - x_0}{\sqrt{r^2 - (x - x_0)^2}} \tag{4.41}$$

$$f(x) = \frac{1}{\frac{1}{m(x)}^2 + 1} = \left(\frac{x - x_0}{r}\right)^2 = a(x - x_0)^2 = a(x^2 + x_0^2 - 2x_0 x) \tag{4.42}$$

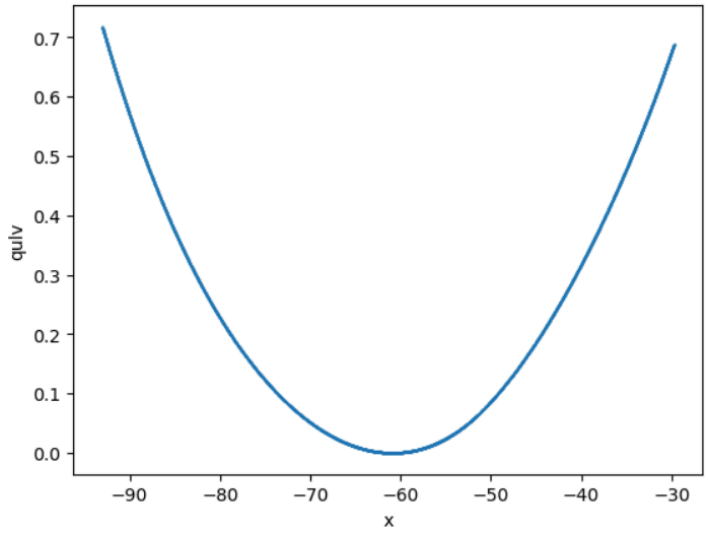

图 4.32: 函数完全没有断点

如果函数是分段的情况下

$$\begin{aligned} r = r_1, x_0 = x_1 \quad & x < m_1 \\ r = r_2, x_0 = x_2 \quad & x \geq m_1 \& x < m_2 \\ r = r_3, x_0 = x_3 \quad & x > m_2 \end{aligned} \tag{4.43}$$

$$f(x) = \begin{cases} \frac{1}{r_1^2}(x - x_1)^2 & x < m_1 \\ \frac{1}{r_2^2}(x - x_2)^2 & x \geq m_1 \& x < m_2 \\ \frac{1}{r_3^2}(x - x_3)^2 & x > m_2 \end{cases} \tag{4.44}$$

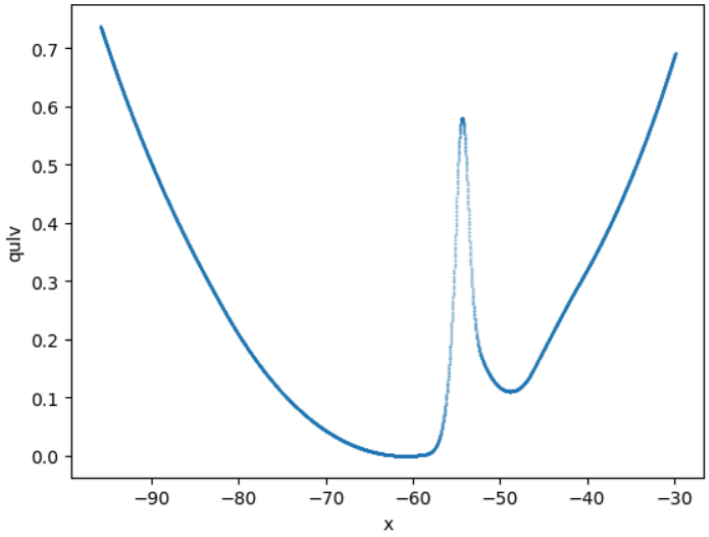

图 4.33: 函数完全没有断点

一种可能的方法是对其求二阶导数，然后直接分割

$$f(x)'' = \begin{cases} \frac{2}{r_1} & x < m_1 \\ -\frac{2}{r_2} & x \geq m_1 \& x < m_2 \\ \frac{2}{r_3} & x > m_2 \end{cases} \tag{4.45}$$

但是实际效果其实并不好。

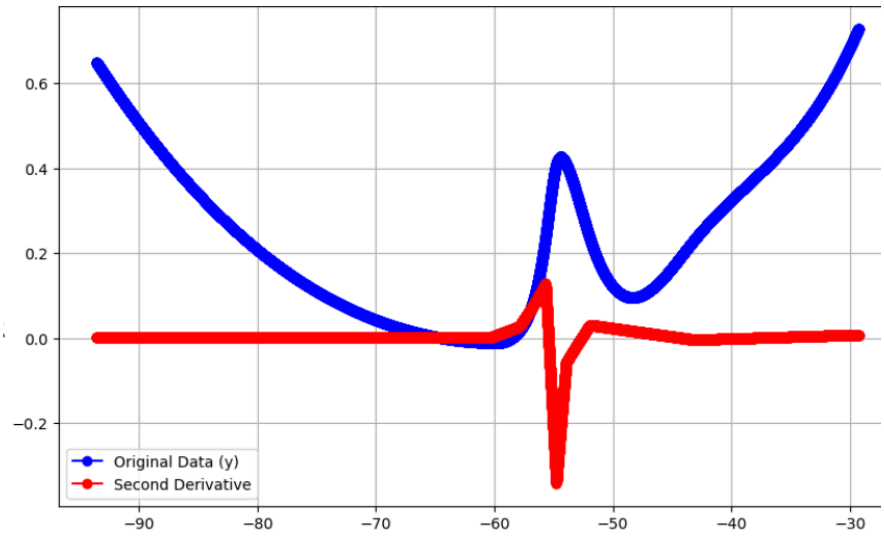

图 4.34: 中间部分二阶导数并不是直线

但是其实也能分割，除了传统的分割方法，也可以训练一个简单的全连接神经网络。

如果下半部分的圆弧，切线方程只是变成

$$m(x) = \frac{x - x_0}{\sqrt{r^2 - (x - x_0)^2}} \tag{4.46}$$

$$f(x) = \frac{1}{\frac{1}{m(x)}^2 + 1} = \left(\frac{x - x_0}{r}\right)^2 = a(x - x_0)^2 = a(x^2 + x_0^2 - 2x_0 x) \tag{4.47}$$

方程是一样的表达形式。

4.4 去噪

去噪暂时就是最简单的半径滤波，效果很好。

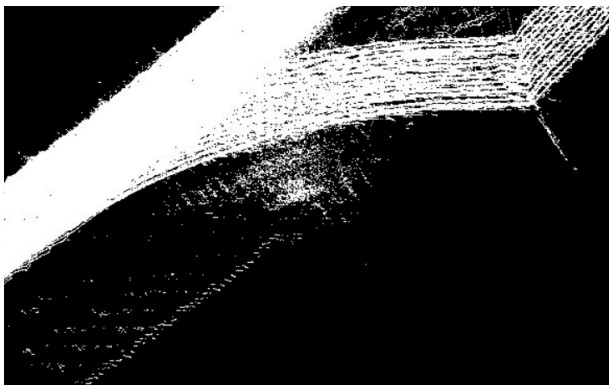

图 4.35: 去噪前截面

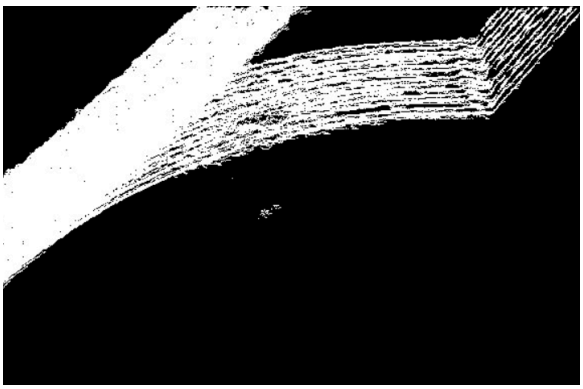

图 4.36: 去噪后截面

5 最优化方法

5.1 无约束优化方法

考虑无约束优化问题

$$\min_{x \in \mathbb{R}^n} f(x) \tag{5.48}$$

5.1.1 梯度下降法

找到可微函数的局部最小值

给定**初始值** $x_0, k = 0$
while
  计算出$\frac{\partial f}{\partial x}, \text{cost}$
  $x = x - \alpha\left(\frac{\partial f}{\partial x}\right)$
  if $\text{cost} < \varepsilon$
    break
end

$$\begin{aligned}\theta(j)' &= \theta(j) - \alpha\frac{\partial}{\partial x_j}f(x) \ \ \& \ \text{每一个参数都更新完再下一步} \\ \theta(j) &= \theta(j)'\end{aligned} \tag{5.49}$$

容易走出锯齿路线，从而增加迭代次数。

这里优化的话有 adam 法优化学习率，随机梯度下降，小批量梯度下降。

5.1.2 牛顿法

梯度下降法基本策略是沿着一阶最速下降方向迭代，当$\nabla^2 f(x)$较大时，收敛较慢。如果我们的$f(x)$足够光滑（数据量足够大），我们可以利用二阶信息改进下降方向。

对于函数$f(x)$考虑其在x_k点处的二阶泰勒近似

$$f(x_k + d_k) = f(x_k) + \nabla f(x_k)^T d_k + \frac{1}{2}d_k^T \nabla^2 f(x_k)^T d_k + o(\|x_k\|^2) \tag{5.50}$$

忽略高阶项，右边看作关于d_k的函数，并且最小化（矩阵求导）

$$\left(f(x_k)^T\right)^T + \frac{1}{2}\left(\nabla^2 f d_k + (\nabla^2 f)^T d_k\right) = 0 \tag{5.51}$$

这里也能看出来为什么其一定要满足正定条件。

$$\nabla^2 f(x_k) d_k = -\nabla f(x_k) \tag{5.52}$$

$$x_{k+1} = x_k - \alpha_k \nabla^2 f(x_k)^{-1} \nabla f(x_k) \tag{5.53}$$

5.1.3 拟牛顿法

5.1.4 高斯牛顿法

$$f(x_k + \Delta x) \approx f(x_k) + J(x_k)^T \Delta x \tag{5.54}$$

我们的目标函数可以近似为：

$$min\frac{1}{2} \parallel f(x_k) + J(x_k)^T \Delta x \parallel_2^2 \tag{5.55}$$

现在，我们将近似的目标函数展开：

$$\begin{aligned}M(\Delta X) = \frac{1}{2}f(x+\Delta X)^2 &= \frac{1}{2}(f(x) + J^T\Delta X)^T(f(x) + J^T\Delta X) \\ &= \frac{1}{2}\Big[\underbrace{f(x)^T + \Delta X^T J}_{\text{E}}\Big]\Big[\underbrace{f(x) + J^T\Delta X}_{\text{F}}\Big] \\ &= \frac{1}{2}\Big[\underbrace{\|f(x)\|^2}_{\text{A}} + \underbrace{f(x)^T J^T \Delta X}_{\text{B}} + \underbrace{\Delta X^T J f(x)}_{\text{C}} + \underbrace{\Delta X^T J J^T \Delta X}_{\text{D}}\Big]\end{aligned} \tag{5.56}$$

$$
\begin{aligned}
dC &= d\,\mathrm{Tr}[f(x)^T J^T \Delta X] = \mathrm{Tr}[d(f(x)^T J^T \Delta X)] \\
&= \mathrm{Tr}[f(x)^T J^T d\Delta X] \Rightarrow \frac{\partial C}{\partial \Delta X} = Jf(x)
\end{aligned} \tag{5.57}
$$

$$
\begin{aligned}
\frac{\partial M(\Delta X)}{\partial \Delta X} &= \frac{1}{2}\Big(0 + Jf(X) + Jf(x) + \overbrace{2(JJ^T\Delta X)}^{\text{使用了矩阵求导的定理}}\Big) \\
&= Jf(x) + JJ^T\Delta X = 0
\end{aligned} \tag{5.58}
$$

$$
H\Delta X = g \;\;\&\;\; H = JJ^T \;\;\&\;\; g = -Jf(x) \tag{5.59}
$$

所以高斯牛顿法的步骤是

1 给定**初始值** $x_0, k = 0$
2 **while**
3 | 计算出$J(x_k), f(x_k)$ & $H = JJ^T$ & $g = -Jf(x)$
4 | 使用$H\Delta X = g$求出Δx
5 | **if** $\Delta x < \varepsilon$
6 | | **break**
7 | **else**
8 | | $x_{k++} = x_k + \Delta x$
9 **end**

它的缺点是要求H矩阵可逆，而且计算量大，有时候可能无解。并且Δx过大会导致其局部近似不精确，严重的时候，可能无法保证迭代收敛。同时也会锯齿状增大迭代次数（和梯度下降一样）。

5.1.4.1 列文伯格-马夸特法(LM)

为了避免其迭代次数过长的缺点，在高斯牛顿的基础上进行优化，提出一个信赖区域。

$$
\rho = \frac{f(x + \Delta x) - f(x)}{J^T \Delta x} \tag{5.60}
$$

如果它接近一就不需要更改，如果过大就需要缩小步长，如果过小就需要增大步长，这样的话，就可以动态调整步长了。最优化问题变为

$$
\min \frac{1}{2}\left\| f(x) + J^T\Delta x \right\|_2^2 \;\; s.t \;\; \|D\Delta x\|_2^2 < \mu \tag{5.61}
$$

构建拉格朗日函数[8]

[8]目标函数为 f 在约束条件 g 下的极值与其拉格朗日函数的极值相同。λ 被称为拉格朗日算子

$$L(\Delta x, \lambda) = \frac{1}{2}\left\| f(x) + J^T \Delta x \right\| + \lambda(\|D\Delta x\|_2^2 - \mu) \tag{5.62}$$

$$\frac{\partial L(\Delta x, \lambda)}{\partial \Delta x} = Jf(x) + JJ^T \Delta X + \lambda \frac{\partial(\Delta x^T D^T D \Delta x)}{\partial \Delta x} = Jf(x) + JJ^T \Delta x + \lambda D^T D \Delta x \tag{5.63}$$

之前的H变为$JJ^T + \lambda D^T D$,求解步骤变为

1 给定**初始值** $x_0, k = 0$
2 **while**
3 计算出$J(x_k), f(x_k)$ & $H = JJ^T + \lambda D^T D$ & $g = -Jf(x)$
4 计算出Δx
5 计算$\rho_k = \frac{f(x+\Delta x) - f(x)}{J^T \Delta x}$
6 **if** $\rho_k < \frac{1}{4}$
7 $\Delta x_k = \frac{1}{4}\Delta x_k$
8 **else**
9 **if** $\rho_k > \frac{3}{4}$
10 $\Delta x_k = \min(2\Delta x_k, \mu)$
11 **else**
12 $\Delta x_k = \Delta x_k$
13 **if** $\rho_k > \xi$
14 $x_{k+1} = x_k + \Delta x$
15 **else**
16 $x_{k+1} = x_k$
17 k=k+1
18 **if** $\Delta x_k < \varepsilon$
19 **break**
20 **end**

参考文献

1. Beder, C. & Förstner, W. Direct Solutions for Computing Cylinders from Minimal Sets of 3D Points. *Computer Vision – ECCV 2006* Bd. 3951 135–146 (2006)
2. Lalonde, J.-F., Vandapel, N. & Hebert, M. Automatic Three-Dimensional Point Cloud Processing for Forest Inventory.
3. Nurunnabi, A. Robust Cylinder Fitting in Laser Scanning Point Cloud Data.
4. Bolles, R. C. & Fischler, M. A. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. in *IJCAI* Bd. 1981 637–643 (1981).
5. Lukács, G., Martin, R. & Marshall, D. Faithful Least-Squares Fitting of Spheres, Cylinders, Cones and Tori for Reliable Segmentation. *Computer Vision — ECCV'98* Bd. 1406 671–686 (1998)
6. Nurunnabi, A., Sadahiro, Y. & Lindenbergh, R. ROBUST CYLINDER FITTING IN THREE-DIMENSIONAL POINT CLOUD DATA. (2017)
7. Faber, P. & Fisher, R. B. A Buyer's Guide to Euclidean Elliptical Cylindrical and Conical Surface Fitting. in *Procedings of the British Machine Vision Conference 2001* 54 (British Machine Vision Association, Manchester, 2001). doi:10.5244/C.15.54
8. Wang, H. & Suter, D. Using Symmetry in Robust Model Fitting. *Pattern Recognition Letters* **24,** 2953–2966 (2003)
9. Su, M.-C. & Chou, C.-H. A modified version of the K-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23,** 674–680 (2001)
10. Hubert, M., Rousseeuw, P. J. & Vanden Branden, K. ROBPCA: A New Approach to Robust Principal Component Analysis. *Technometrics* **47,** 64–79 (2005)
11. Leyder, S., Raymaekers, J., Rousseeuw, P. J., Servotte, T. & Verdonck, T. RobPy: A Python Package for Robust Statistical Methods. (2024) doi:10.48550/arXiv.2411.01954
12. Al-Sharadqah, A. & Chernov, N. Error Analysis for Circle Fitting Algorithms. *Electronic Journal of Statistics* **3,** (2009)
13. Chernov, N. MATLAB Codes for Circle Fitting. https://people.cas.uab.edu/~mosya/cl/MATLABcircle.html (2023)
14. Fei, Z., Wu, Z., Xiao, Y., Ma, J. & He, W. A New Short-Arc Fitting Method with High Precision Using Adam Optimization Algorithm. *Optik* **212,** 164788 (2020)
15. Vosselman, G. Performance Analysis of Line and Circle Fitting in Digital Images. (1996)