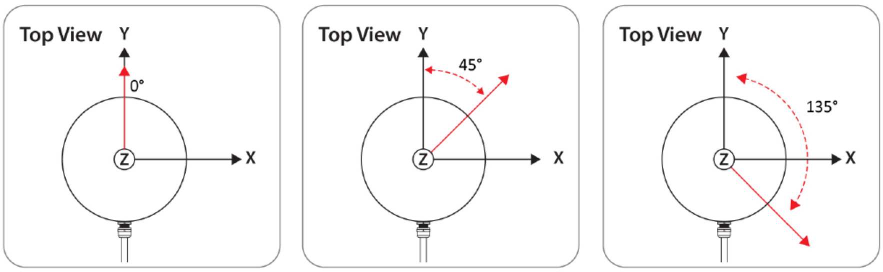
Velodyne VLP-32C 激光雷达测试总结

考察指标

Velodyne VLP-32C 距离精度指标：±3cm（大多数通道上的环境墙测试性能）。

测试环境及测试方法

坐标系定义



Phase offset at 0°　　Phase offset at 45°　　Phase offset at 135°

测试环境及方法

以米尺为标准，水平放置 Lidar，0°航向方向正对墙面，读取 Lidar 0°俯仰角、一定航向角度范围内（此范围内激光线能发射到墙面）Y 方向至墙面的距离，其与米尺的标准值之差记为距离误差。

测试中，由于条件所限，结果受地面水平度、墙面垂直度、米尺弯曲及准确度、Lidar 自

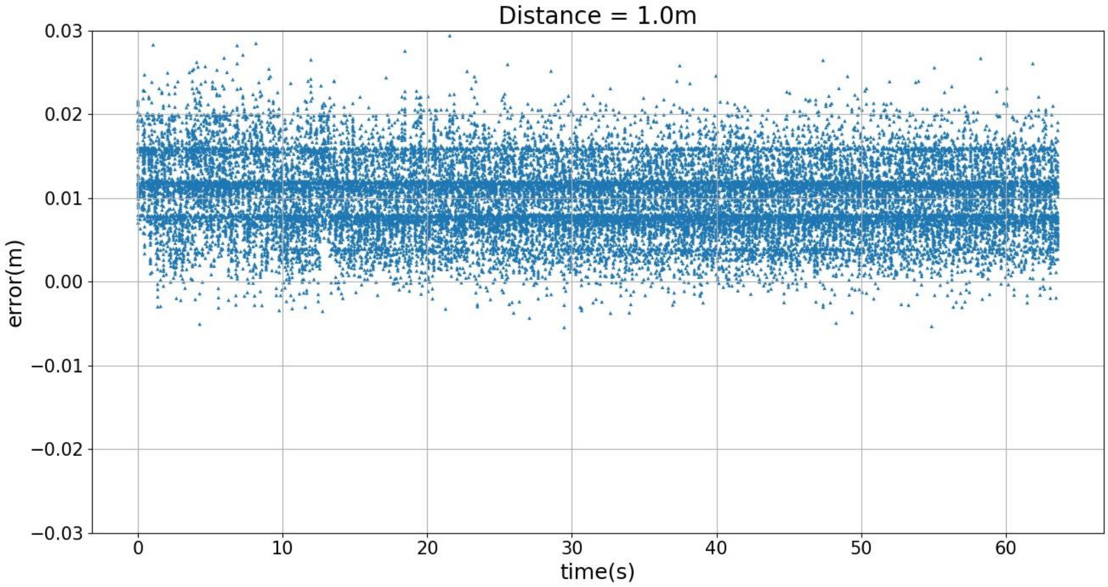
身旋转导致的振动、激光雷达人工放置位置误差等因素的影响。

受环境和测量设备限制，测量距离有：1m、2m、3m、4m、5m、6m。

测试结果

1m

记录共 21892 组数据，仅一组误差绝对值大于 3cm，为 3.054cm，其余数据误差分布在-0.545cm～+2.943cm，数据误差分布如下：

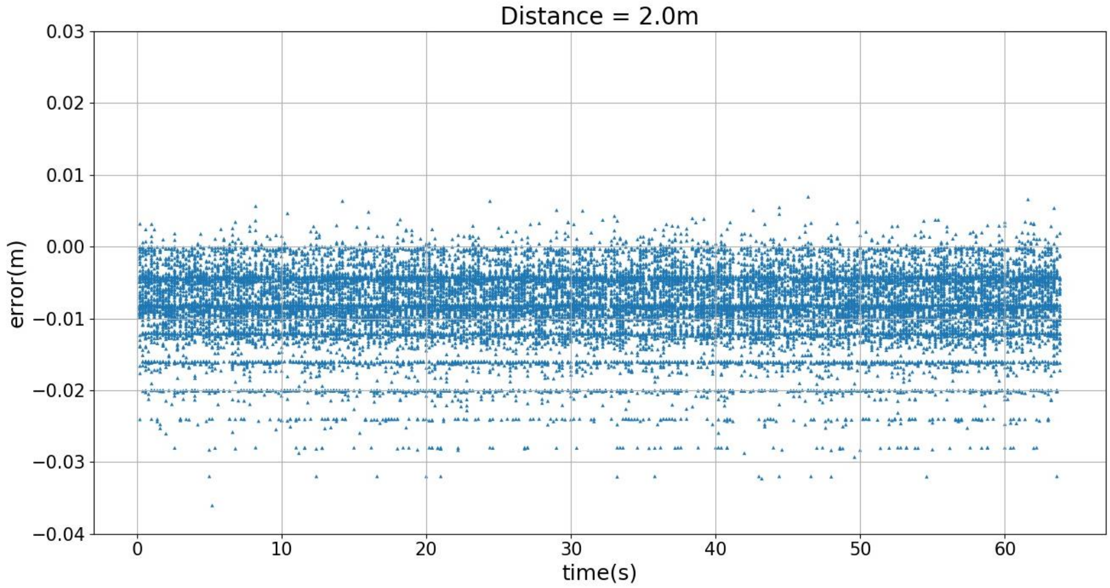


2m

共 17702 组数据，其中有 15 组数据误差绝对值大于 3cm，如下图：

17671	63.7997	357.2667	1.992	-0.09499	1.989734	0	-0.02808
17672	63.80002	357.3667	1.992	-0.09152	1.989896	0	-0.02811
17673	63.8003	357.4667	1.992	-0.08805	1.990053	0	-0.02814
17674	63.80059	357.5667	1.996	-0.08474	1.9942	0	-0.02814
17675	63.80086	357.665	1.992	-0.08116	1.990346	0	-0.02814
17676	63.80115	357.7567	1.992	-0.07797	1.990473	0	-0.02817
17677	63.80143	357.8567	1.988	-0.07435	1.986609	0	-0.02817
17678	63.8017	357.9567	1.988	-0.07088	1.986736	0	-0.02817
17679	63.802	358.0567	2	-0.06782	1.99885	0	-0.02827
17680	63.80334	0.546667	1.988	0.018967	1.98791	0	-0.02831
17681	63.80362	0.646667	1.992	0.022482	1.991873	0	-0.02831
17682	63.80392	0.746667	1.988	0.025906	1.987831	0	-0.02832
17683	63.8042	0.846667	1.984	0.029317	1.983783	0	-0.02835
17684	63.80448	0.946667	1.996	0.032977	1.995728	0	-0.02835
17685	63.80475	1.046667	1.996	0.03646	1.995667	0	-0.02838
17686	63.80503	1.146667	1.996	0.039943	1.9956	0	-0.02877
17687	63.8053	1.246667	1.996	0.043426	1.995528	0	-0.02877
17688	63.80558	1.346667	1.992	0.046815	1.99145	0	-0.02932
17689	63.8059	1.445	1.996	0.050334	1.995365	0	-0.032
17690	63.80617	1.536667	1.988	0.053312	1.987285	0	-0.032
17691	63.80644	1.636667	1.988	0.05678	1.987189	0	-0.032
17692	63.80778	1.736667	1.992	0.060369	1.991085	0	-0.032
17693	63.80809	1.836667	1.992	0.063844	1.990977	0	-0.032
17694	63.80852	1.936667	1.996	0.067454	1.99486	0	-0.032
17695	63.80909	2.036667	1.984	0.070509	1.982747	0	-0.03201
17696	63.80953	2.136667	2	0.074566	1.998609	0	-0.03201
17697	63.80985	2.236667	1.988	0.077586	1.986485	0	-0.03201
17698	63.81016	2.336667	1.992	0.081216	1.990344	0	-0.03201
17699	63.81045	2.436667	1.996	0.08486	1.994195	0	-0.03204
17700	63.81074	2.536667	1.996	0.08834	1.994044	0	-0.03204
17701	63.81102	2.636667	2.004	0.092189	2.001878	0	-0.03205
17702	63.81129	2.736667	1.996	0.0953	1.993724	0	-0.03232
17703	63.81156	2.836667	1.988	0.098384	1.985564	0	-0.03606
17704							

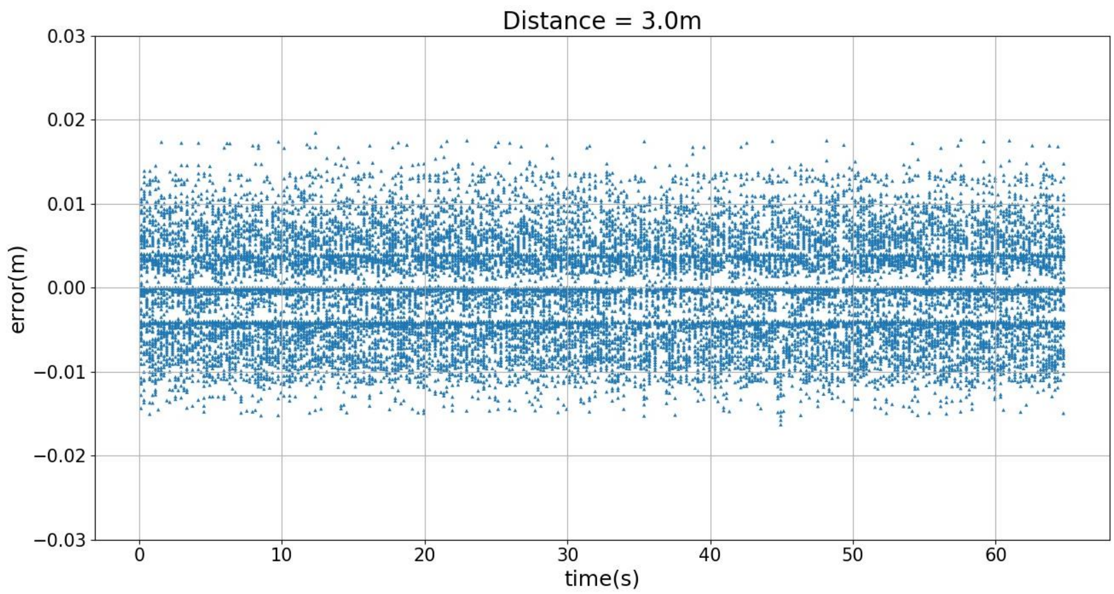
其余数据误差在-2.932cm～+0.6946cm 之间，分布如下图：



分析误差分布发现误差值主要为负，15 组误差绝对值大于 3cm 的数据组很可能是受到 Lidar 放置位置和俯仰角度的影响。

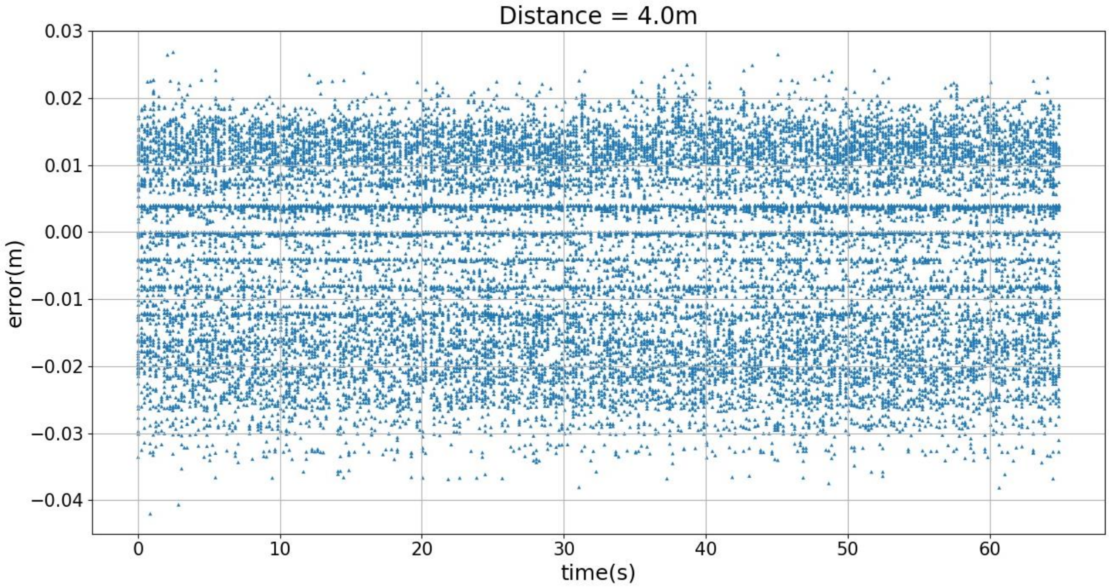
3m

共 17554 组数据，数据误差在-1.63cm～+1.847cm 之间，其分布如下：



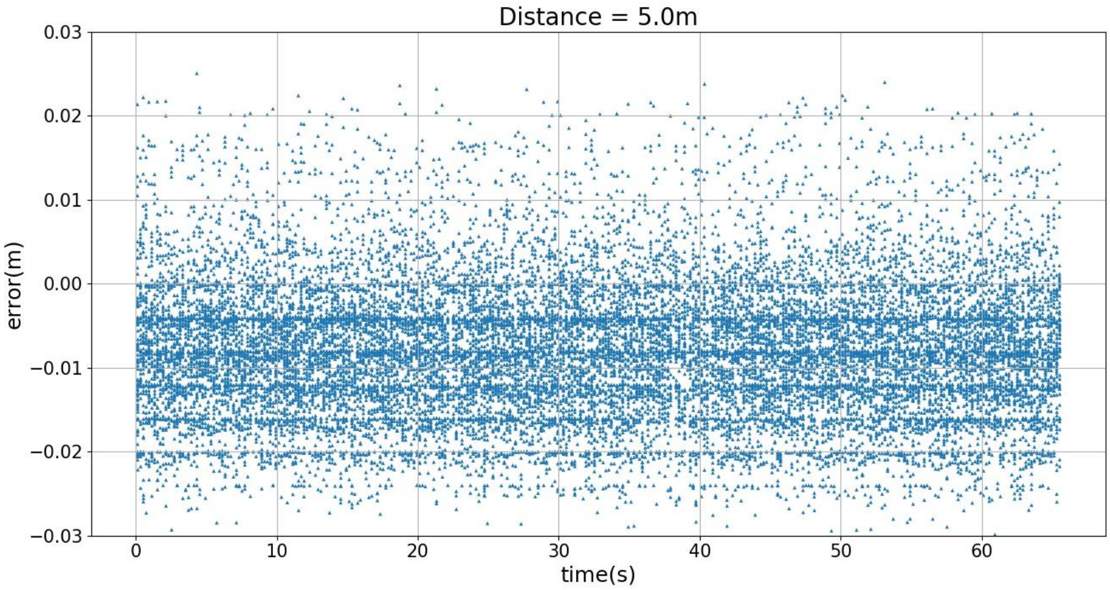
4m

共 17946 组数据，其中有 281 组数据误差绝对值大于 3cm，但最大不超过 4.3cm。整体数据误差分布如下：



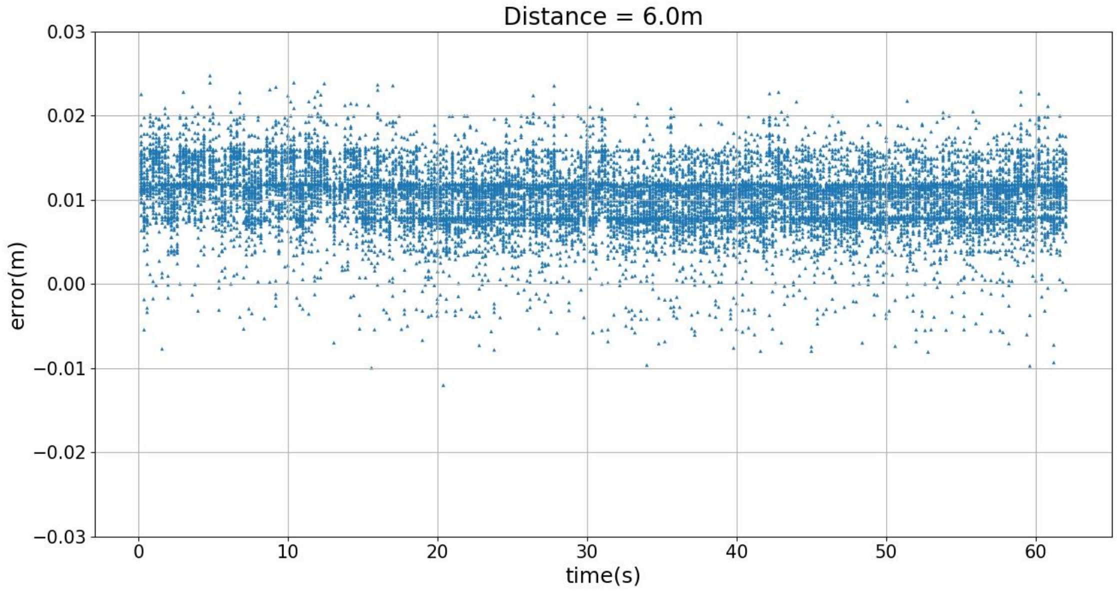
5m

共 18142 组数据，数据误差在-2.988cm～+2.508cm 之间，其分布如下：



6m

共 12812 组数据，数据误差在-1.202cm～+2.478cm 之间，其分布如下：



总结

测量距离	样本量（组）	误差±3cm以内样本量（组）	误差绝对值大于 3cm 样本量（组）	误差符合指标范围样本百分比	误差超过指标范围，最大误差绝对值	备注
1m	21892	21891	1	99.995%	3.054cm	误差的出现不排除地面平整度、墙面垂直度、Lidar 自身旋转导致的振动、人为放置Lidar的位置误差等因素的影响
2m	17702	17687	15	99.915%	3.606cm	
3m	17554	17554	0	100%	-	
4m	17946	17665	281	98.43%	4.203cm	
5m	18142	18142	0	100%	-	
6m	12812	12812	0	100%	-	

若考虑 1~2cm 的外在测试条件引起的不确定性，在测试的几种距离工况下，Velodyne VLP-32C Lidar 的测量距离精度满足其声明的指标要求。

附：通信及数据解析程序（Python）

```python
import socket
import math
import time
import sys
from time import sleep

# UDP 通信
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('', 2368))
s.connect(('192.168.1.201', 2368))

# 传感器信息
marker = b'\xffee'
n_azimuth_bytes = 2
n_channels = 32
n_data_block_bytes = 100
n_channel_data_bytes = 3
n_data_blocks = 12
pitch_deg = [-25.0, -1.0, -1.667, -15.639, -11.31, 0.0, -0.667, -8.843,
            -7.254, 0.333, -0.333, -6.148, -5.333, 1.333, 0.667, -4.0,
            -4.667, 1.667, 1.0, -3.667, -3.333, 3.333, 2.333, -2.667,
            -3.0, 7.0, 4.667, 2.333, -2.0, 15.0, 10.333, -1.333]
delta_deg = [1.4, -4.2, 1.4, -1.4, 1.4, -1.4, 4.2, -1.4,
            1.4, -4.2, 1.4, -1.4, 4.2, -1.4, 4.2, -1.4,
            1.4, -4.2, 1.4, -4.2, 4.2, -1.4, 1.4, -1.4,
            1.4, -1.4, 1.4, -4.2, 4.2, -1.4, 1.4, -1.4]

deg2rad = math.pi / 180.0
rad2deg = 180.0 / math.pi
dist_meas = 4.0
s_name = str(dist_meas) + '.csv'  # 存储的文件名


# 输入三字节数据，返回距离值
def get_distance(data_per_channel):
    distance = (int(data_per_channel[1]) * 256 + int(data_per_channel[0])) * 4.0 / 1000.0
    return distance


# 输入 channel 值、距离值、航向角度，返回(x,y,z)
```

```python
def get_xyz(channel, dist, azimuth):
    """
    x = dist * math.cos(pitch_deg[channel] * deg2rad) * math.sin((azimuth +
delta_deg[channel]) * deg2rad)
    y = dist * math.cos(pitch_deg[channel] * deg2rad) * math.cos((azimuth +
delta_deg[channel]) * deg2rad)
    z = dist * math.sin(pitch_deg[channel] * deg2rad)
    """
    x = dist * math.cos(pitch_deg[channel] * deg2rad) * math.sin(azimuth * deg2rad)
    y = dist * math.cos(pitch_deg[channel] * deg2rad) * math.cos(azimuth * deg2rad)
    z = dist * math.sin(pitch_deg[channel] * deg2rad)
    return x, y, z


# print(get_XYZ(21, 2.386, 0.16))  # (-0.051546641,2.381406307,0.138719708)


# 输入两字节数据，返回角度值
def get_azimuth(azimuth_data):
    angle = (int(azimuth_data[1]) * 256 + int(azimuth_data[0])) / 100.0
    return angle

# 精确航向角度计算
def get_precision_azimuth(data_packet):  # 传引用
    i = 0
    azimuth_gap = 0.0
    for i in range(12):
        if i < 11:
            if data_packet[i + 1]['Azimuth'] < data_packet[i + 1]['Azimuth']:
                data_packet[i + 1]['Azimuth'] += 360
            azimuth_gap = data_packet[i + 1]['Azimuth'] - data_packet[i]['Azimuth']
        j = 0
        while j < 32:
            data_packet[i]['channel %d' % (j)][0] = data_packet[i]['Azimuth'] + azimuth_gap
* j * 2.304 / 55.296 + \
                                                    delta_deg[j]
            data_packet[i]['channel %d' % (j + 1)][0] = data_packet[i]['Azimuth'] +
azimuth_gap * j * 2.304 / 55.296 + \
                                                        delta_deg[j + 1]
            if data_packet[i]['channel %d' % (j)][0] > 360:
                data_packet[i]['channel %d' % (j)][0] -= 360

            elif data_packet[i]['channel %d' % (j)][0] < 0:
```

```
                data_packet[i]['channel %d' % (j)][0] += 360

            if data_packet[i]['channel %d' % (j + 1)][0] > 360:
                data_packet[i]['channel %d' % (j + 1)][0] -= 360

            elif data_packet[i]['channel %d' % (j + 1)][0] < 0:
                data_packet[i]['channel %d' % (j + 1)][0] += 360

            j += 2


# 每个 data_packet 包含 12 个 data_block, data_block 形式如下，每个 channel 包含 azimuth 和 distance:
"""
data_block = {'Azimuth': .0,
              'channel 0': [.0, .0], 'channel 1': [.0, .0], 'channel 2': [.0, .0], ' channel 3': [.0, .0],
              'channel 4': [.0, .0], 'channel 5': [.0, .0], 'channel 6': [.0, .0], 'channel 7': [.0, .0],
              'channel 8': [.0, .0], 'channel 9': [.0, .0], 'channel 10': [.0, .0], 'channel 11': [.0, .0],
              'channel 12': [.0, .0], 'channel 13': [.0, .0], 'channel 14': [.0, .0], 'channel 15': [.0, .0],
              'channel 16': [.0, .0], 'channel 17': [.0, .0], 'channel 18': [.0, .0], 'channel 19': [.0, .0],
              'channel 20': [.0, .0], 'channel 21': [.0, .0], 'channel 22': [.0, .0], 'channel 23': [.0, .0],
              'channel 24': [.0, .0], 'channel 25': [.0, .0], 'channel 26': [.0, .0], 'channel 27': [.0, .0],
              'channel 28': [.0, .0], 'channel 29': [.0, .0], 'channel 30': [.0, .0], 'channel 31': [.0, .0]}
"""


# 返回 data packet
def getinfo(recv_Data):
    data_packet = []
    for i in range(n_data_blocks):
        data_block = dict()
        block_azimuth = get_azimuth(
            recv_Data[i * n_data_block_bytes + 2: i * n_data_block_bytes + 4])  # ffee 标志位后的两字节为 azimuth
        data_block.update({'Azimuth': block_azimuth})
        for j in range(n_channels):
```

```
            s_t = 'channel ' + str(j)
            dist = get_distance(recv_Data[
                                 i * n_data_block_bytes + 4 + j * n_channel_data_bytes: i *
n_data_block_bytes + 7 + j * n_channel_data_bytes])
            data_block.update({s_t: [block_azimuth, dist]})  # 此处尚未计及精确的 azimuth
        data_packet.append(data_block)
    get_precision_azimuth(data_packet)
    return data_packet


with open(s_name, 'a') as f:
    f.write("time, Azimuth, channel 5 distance, channel5 X, channel5 Y, channel5 Z,
error\n")
    f.close()

time_start = time.clock()

while True:
    recvData = s.recv(1248)  # 接收 udp 数据
    # print(recvData)
    li1 = []
    d_p = getinfo(recvData)  # 解析 data_packet
    for i in range(12):
        # li = [d_p[i]['Azimuth']]
        t = d_p[i]['channel 5'][1]  # 默认为 channel 5 距离
        # 1-(-1), 5-(0), 9-(0.333),10-(-0.333), 18-(1), 考虑垂直放置误差，取用最短距离
        """
        if min(d_p[i]['channel 5'][1], d_p[i]['channel 9'][1], d_p[i]['channel 10'][1]) !=
0:
            t = min(d_p[i]['channel 5'][1], d_p[i]['channel 9'][1], d_p[i]['channel 10'][1])
            # t = (d_p[i]['channel 5'][1] + d_p[i]['channel 9'][1] + d_p[i]['channel
10'][1]) / 3
        """
        # li = [d_p[i]['channel 5'][0], d_p[i]['channel 5'][1]]  # data_block 的 azimuth、
channel5（0 度俯仰角）的 distance
        li = [d_p[i]['channel 5'][0], t]
        li1.append(li)  # li1 包含 12 个 data_block 的 azimuth、channel5（0 度俯仰角）的 distance
    for i in range(12):
        # print("Azimuth:", li1[i][0], "Channel distance", li1[i][1:33])
        if (li1[i][0] < 4.0 or li1[i][0] > 356.0) and li1[i][1] != .0:  # 限定航向角度范围
            with open(s_name, 'a') as f:
                time_elapse = time.clock() - time_start
```

```
            # 写入时间、角度、距离、X、Y、Z、error
            f.write("%f, %f, %f, %f, %f, %f, %f\n" % (
                time_elapse, li1[i][0], li1[i][1], get_xyz(5, li1[i][1], li1[i][0])[0],
                get_xyz(5, li1[i][1], li1[i][0])[1], get_xyz(5, li1[i][1], li1[i][0])[2],
                get_xyz(5, li1[i][1], li1[i][0])[1] - dist_meas))
        f.close()
        print(time_elapse, "Azimuth:", li1[i][0], "channel 5 distance", li1[i][1],
"channel 5 XYZ",
              get_xyz(5, li1[i][1], li1[i][0]))
    # sleep(1)
        # print("delta_Azimuth:%f" % (li1[11][0] - li1[10][0]))  # 5hz:0.09、0.1、0.11,
10hz:0.18、0.19、0.2
```