Livox Mid-360 を MATLAB（Win）で取り込む

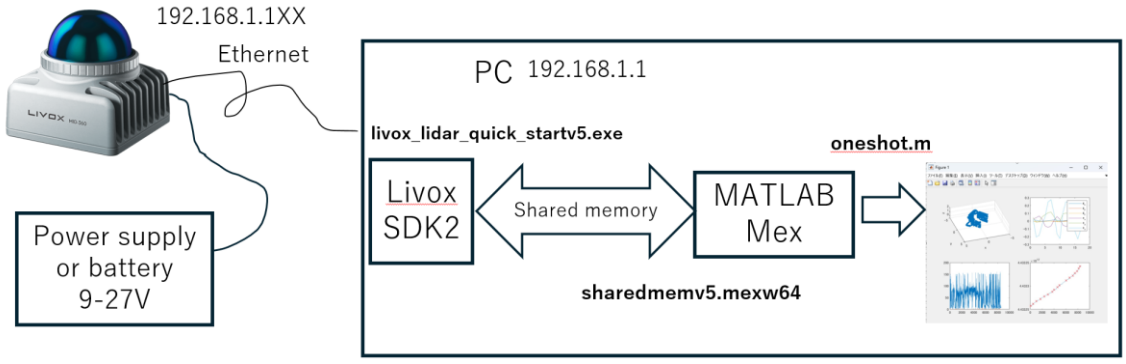
はじめに

LiDAR（Light Detection and Ranging）センサは、距離測定や 3 次元地図作成などのアプリケーションで非常に有用である。Livox Mid-360 は、その中でも 3D LiDAR として、特に安価で、高性能なセンサである。ここでは、Windows 環境で動作する MATLAB を使用して Livox Mid-360 からデータを取得する方法について解説する。

テスト環境

Windows 11
MATLAB 2023a
Visual Studio Community 2022
必要なインストールオプション（C++によるデスクトップ環境）

ここでの説明の最終的な構成イメージを示す。



Mid-360 は、PC と Ethernet で繋がれ、Livox SDK2 のサンプルをベースにした、livox_lidar_quick_startv5.exe でデータが取り込まれる。そのデータは、Shared memory に転送され、Mex 関数 **sharedmemv5.mexw64** を介し、MATLAB 変数として取り込まれ、oneshot 関数により、XYZ データおよび、IMU データに変換される。

Livox Mid-360 の接続と動作確認

接続する PC の設定

ホスト PC 側の設定は静的 IP アドレスモードで IP アドレスは、192.168.1.1/24 として Mid-360 を直接接続した例で説明する。ちなみに、ホスト PC の IP アドレスは、コマンドプロンプトから

```
> ipconfig
イーサネット アダプター Ethernet:
   接続固有の DNS サフィックス . . . . .: lan
   IPv4 アドレス . . . . . . . . . . . .: 192.168.1.1
   サブネット マスク . . . . . . . . . .: 255.255.255.0
   デフォルト ゲートウェイ . . . . . . .: 192.168.1.1
```

でチェックできる。ホスト PC の IP アドレスは、Mid-360 の IP アドレス（192.168.1.1XX）とは、異なるので注意！

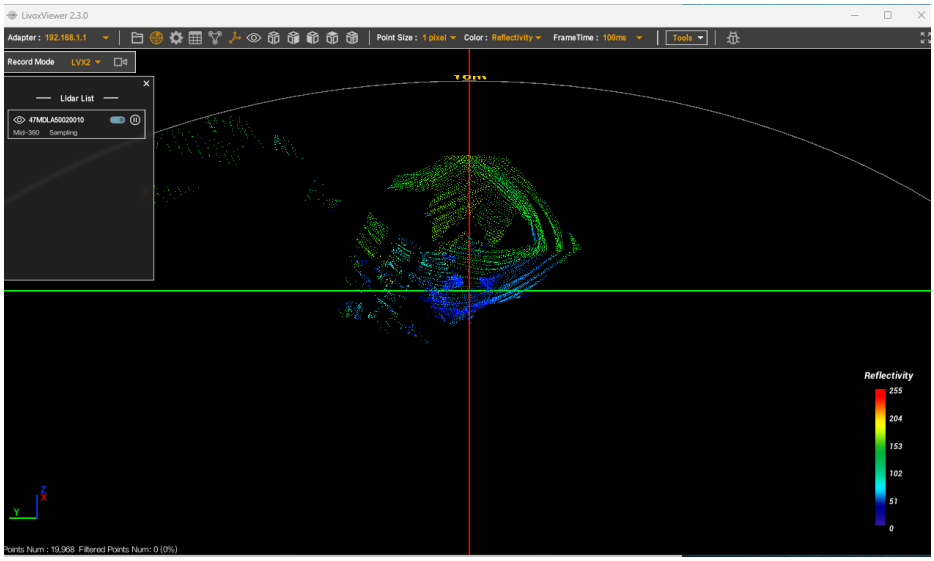
ホスト PC（自分の PC）で行う場合、コントロールパネルのネットワークアダプタ設定からイーサネットの IP アドレスを 192.168.1.1、255.255.255.0、192.168.1.1 に設定する必要がある。

Mid-360 の動作確認

https://www.livoxtech.com/jp/mid-360/downloads

から Livox Viewer 2 をダウンロードし、起動しチェックする。繋がらない場合には、左上の Adapter プルダウンメニューからホスト PC の IP アドレス（192.168.1.1）に変更する。

サンプルプログラムの動作確認



Livox-SDK2 の準備

https://github.com/Livox-SDK/Livox-SDK2

から Releases　v1.2.5 の Source code をダウンロードし、解凍する。

必要なサンプルのみ更新するようにする。具体的には、

Livox-SDK2-1.2.5¥samples¥CmakeLists.txt

を編集し、最後の add_subdirectory を全部消し、つぎのように変更する。

オリジナル	新
add_subdirectory(livox_lidar_quick_start) add_subdirectory(multi_lidars_upgrade) add_subdirectory(logger) add_subdirectory(debug_point_cloud) add_subdirectory(lidar_cmd_observer) add_subdirectory(livox_lidar_rmc_time_sync)	add_subdirectory(livox_lidar_quick_start) add_subdirectory(livox_lidar_quick_startv5)

参考にディレクトリ構造のイメージを示す。黄色の部分が追加・修正する箇所である。

```
¥Livox-SDK2-1.2.5
├─sdk_core
├─samples
│  ├─CMakeLists.txt
│  ├─livox_lidar_quick_start
│  │  ├─CMakeLists.txt
│  │  ├─config.json
│  │  ├─main.cpp
│  │  └─sharedmem2.c
│  └─livox_lidar_quick_startv5
│     ├─CMakeLists.txt
│     ├─main.cpp
│     └─sharedmemv5.cpp
├─include
├─build
│  ├─samples
│  │  ├─livox_lidar_quick_start
│  │  │  └─Release
│  │  │      ├─livox_lidar_quick_start.exe
│  │  │      └─config.json
│  │  ├─livox_lidar_quick_startv5
│  │  │  └─Release
│  │  │      ├─livox_lidar_quick_startv5.exe
│  │  │      └─config.json
└─3rdparty
```

プロジェクトファイルの生成とコンパイル

Visual Studio をインストールすると、Visual C++が使えるようになる。

ここでは、IDE を使わず、Command Prompt で作業を行っていく。

まずは、Visual C++のコマンドが実行できるように、

Developer Command Prompt を起動する。ちなみ、メニューバーの検索で「Developer」と入力すると候補に出てくる。

```
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.8.3
** Copyright (c) 2022 Microsoft Corporation
**********************************************************************
C:¥Program Files¥Microsoft Visual Studio¥2022¥Community>
```

Developer Command Prompt から Livox-SDK2 ディレクトリに移動し、build フォルダを

作る。

```
> cd Livox-SDK2-1.2.5
> mkdir build
> cd build
```

CMake によるプロジェクトの生成

```
> cmake .. -A x64
```

msbuild による実行ファイルの生成

```
> msbuild livox_sdk2.sln /p:Configuration=Release /p:Platform=x64 /p:RuntimeLibrary=MultiThreadedDLL
```

コンパイルが成功すると、Release フォルダ内に実行ファイルができる。

```
Livox-SDK2-1.2.5¥build¥samples¥livox_lidar_quick_start¥Release
```

サンプルプログラムは、config ファイルが必要なので、以下のフォルダにある config.json を Release フォルダにコピーする。

```
Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_start¥config.json
```

config.json ファイルの編集

host_net_info の IP アドレスを修正する。なお、ここでの IP アドレス（192.168.1.1）は、Mid-360 の IP アドレスではなく、ホスト PC の IP アドレスである。

config.json の設定例

```
{
	"MID360": {
		"lidar_net_info" : {
			"cmd_data_port": 56100,
			"push_msg_port": 56200,
			"point_data_port": 56300,
			"imu_data_port": 56400,
			"log_data_port": 56500
		},
		"host_net_info" : {
			"cmd_data_ip" : "192.168.1.1",
			"cmd_data_port": 56101,
			"push_msg_ip": "192.168.1.1",
			"push_msg_port": 56201,
			"point_data_ip": "192.168.1.1",
			"point_data_port": 56301,
			"imu_data_ip" : "192.168.1.1",
			"imu_data_port": 56401,
			"log_data_ip" : "192.168.1.1",
			"log_data_port": 56501
		}
	}
}
```

サンプルプログラムの動作確認

コマンドプロンプトから

```
> livox_lidar_quick_start.exe  config.json
```

として実行し、エラーが出ないか確認する。

MATLAB と Livox SDK2 の接続

MATLAB と Livox SDK2 の連携には、いくつかの方法が考えられる。

1．MEX 関数から直接リンクした Static Library を呼び出す
2．Dynamic DLL 機能を使い、DLL library を直接呼び出す
3．共有メモリを用いて相互にデータのやり取りができる MEX 関数とデータ取り込みプログラムを作成する

第 1 の方法は、MATLAB 上から直接、静的ライブラリを呼び出す標準的な方法である。MEX 関数内では、標準出力の stdout,stderr を呼び出しているとクラッシュし異常終了してしまう。そのため、stdout などを使う fprintf 関数などは、使えないため、mexPrintf 関数などに書き換える必要がある。実際、Livox SDK2 のライブラリソースコードで呼び出しており、SDK の将来のバージョンアップなどを考えた場合、多くの箇所を修正する必要が出てくるため、この方法は用いない。第 2 の方法は、ヘッダファイルと、DLL ライブラリがあれば、MATLAB 上では、コンパイルすることなく、直接呼び出せる方法である。基本、ライブラリ関数の呼び出しはできるが、それ以外の処理は、MATLAB で行うことから、処理速度の面で不利になる。第 3 の方法は、データ取り込み部と、MATLAB へのデータ取り込み部の 2 つのプロセスでメモリを共有する方法である。この方法は、機種依存のライブラリ部分を分離し、共有メモリを介して行われるため、汎用性が高く、データ取り込みでの安定性の面からも問題の切り分けがし易いなどの特徴をもつ。ここでは、Livox SDK2 のサンプルプログラムに共有メモリにデータ書き込み部を実装し、それに対応する共有メモリデータ読み込み用 MATLAB　MEX プログラムを作る例をステップバイステップで解説していく。

手順の概要

Mid-360 の API と必要な共有メモリの大きさ

https://livox-wiki-en.readthedocs.io/en/latest/tutorials/new_product/mid360/livox_eth_protocol_mid360.html#point-cloud-imu-data-protocol

によれば、共有したいデータタイプは、0 から 3 まで 4 種類あり、ここでは、この中の IMU data と　Point cloud data 1〜3 のどれか 1 つが必要である。

データフォーマット

Data Type	Sampling Type	Echo Mode	Coordinate Mode	N
0	IMU data			
1	Point cloud data 1	Single echo mode	Cartesian coordinates(32bit)	96
2	Point cloud data 2	Single echo mode	Cartesian coordinates(16bit)	96
3	Point cloud data 3	Single echo mode	Spherical coordinates	96

ちなみに、DataType とバイト数の関係は、

DataType	Bytes
uint8_t	1
int16_t	2
int32_t	4
float	4

で定義されているので、必要なメモリサイズは、

IMU data(6*4=24 bytes)

Field	Offset (bytes)	Data Type	Description
gyro_x	0	float	Unit: rad/s
gyro_y	4	float	Unit: rad/s
gyro_z	8	float	Unit: rad/s
acc_x	12	float	Unit: g
acc_y	16	float	Unit: g
acc_z	20	float	Unit: g

Data Type1(14 bytes*96=1536 bytes)

Field	Offset (bytes)	Data Type	Description
x	0	int32_t	X axis, Unit: mm
y	4	int32_t	Y axis, Unit: mm
z	8	int32_t	Z axis, Unit: mm
Reflectivity	12	uint8_t	Reflectivity
Tag	13	uint8_t	According to the poin field, match the speci

Data Type2(8 bytes*96 = 768 bytes)

Field	Offset (bytes)	Data Type	Description (10mm is
x	0	int16_t	X axis, Unit: 10mm
y	2	int16_t	Y axis, Unit: 10mm
z	4	int16_t	Z axis, Unit: 10mm
Reflectivity	6	uint8_t	Reflectivity
Tag	7	uint8_t	According to the point field, match the specifi

Data Type3(10 bytes*96 = 960 bytes)

Field	Offset (bytes)	Data Type	Description (10mm is
depth	0	uint32_t	Depth, Unit: mm
theta	4	uint16_t	Zenith angle[0, 18000
phi	6	uint16_t	Azimuth[0, 36000], Ur
Reflectivity	8	uint8_t	Reflectivity
Tag	9	uint8_t	According to the point field, match the specifi

と計算できる。Data Type1 がデフォルト出力であり、一番 bytes 数が大きいので、このサイズでデザインする。

Livox SDK2 のサンプルプログラムへの機能追加

Livox SDK2 のサンプルプログラムを編集し、共有メモリにデータを書き込む機能を追加する。具体的には、Livox SDK2 ライブラリから取り込まれたデータをシリアライズして共有メモリに書き込むコードを実装する。通常、2 つのプログラムで共通のヘッダファイルに共有メモリの構造体で定義するが、ここでは、動作確認を優先した実装を考え、フラットにシリアライズして実装した。

```
int SHMEMSIZE = 96*3*4+6*4+1;
```

出力の MATLAB 行列は、3 行(2+96)列で出力することを想定し、以下のように共有メモリのアドレスの割り振りを行った。

Data	Field	Type	shmem　(i=0～95)	MATLAB 行列
Data Type1	x	int32_t	[0,1,2,3]+i*12	1,i
	y	int32_t	[4,5,6,7]+i*12	2,i
	z	int32_t	[8,9,10,11]+i*12	3,i
IMU Data	gyro_x	float	12*96+[0,1,2,3]	1,97
	gyro_y	float	12*96+[4,5,6,7]	2,97
	gyro_z	float	12*96+[8,9,10,11]	3,97
	acc_x	float	12*96+[12,13,14,15]	1,98
	acc_y	float	12*96+[16,17,18,19]	2,98
	acc_z	float	12*96+[20,21,22,23]	3,98
	data->dot_num	uint8_t	12*96+24	

データのシリアライズには、わかりやすく動作確認をするため整数型変換には、シフト演算子を使った方法と、float 型変換には、union を使った方法で実装している。

¥Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_start¥main.cpp

```
//
// The MIT License (MIT)
//
// Copyright (c) 2022 Livox. All rights reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in
// all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.
//

#include "livox_lidar_def.h"
#include "livox_lidar_api.h"

#ifdef _WIN32
#include <winsock2.h>
#else
#include <unistd.h>
#include <arpa/inet.h>
#endif

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <thread>
#include <chrono>
#include <iostream>

LPCTSTR lpName = "(c)2024 www.gerox.com";
int SHMEMSIZE = 96*3*4+6*4+1;
static int initialized = 0;
static uint8_t *shmem;
static HANDLE hFile;


void PointCloudCallback(uint32_t handle, const uint8_t dev_type, LivoxLidarEthernetPacket* data, void*
client_data) {
  if (data == nullptr) {
    return;
  }
//  printf("point cloud handle: %u, data_num: %d, data_type: %d, length: %d, frame_counter: %d¥n",
//      handle, data->dot_num, data->data_type, data->length, data->frame_cnt);
```

```
  if (data->data_type == kLivoxLidarCartesianCoordinateHighData) {
    shmem[SHMEMSIZE-1] = (uint8_t) data->dot_num;
    LivoxLidarCartesianHighRawPoint *p_point_data = (LivoxLidarCartesianHighRawPoint *)data->data;
    for (uint32_t i = 0;i < data->dot_num;i++) {
      int32_t tmp = p_point_data[i].x;
           printf(" %d ",tmp);
      shmem[ 0 + i * 12] = (unsigned char)(tmp        & 0xff);
      shmem[ 1 + i * 12] = (unsigned char)((tmp >> 8)  & 0xff);
      shmem[ 2 + i * 12] = (unsigned char)((tmp >> 16) & 0xff);
      shmem[ 3 + i * 12] = (unsigned char)((tmp >> 24) & 0xff);
      tmp = p_point_data[i].y;
           printf(" %d ",tmp);
      shmem[ 4 + i * 12] = (unsigned char)( tmp        & 0xff);
      shmem[ 5 + i * 12] = (unsigned char)((tmp >>  8) & 0xff);
      shmem[ 6 + i * 12] = (unsigned char)((tmp >> 16) & 0xff);
      shmem[ 7 + i * 12] = (unsigned char)((tmp >> 24) & 0xff);
      tmp = p_point_data[i].z;
           printf(" %d¥n",tmp);
      shmem[ 8 + i * 12] = (unsigned char)( tmp        & 0xff);
      shmem[ 9 + i * 12] = (unsigned char)((tmp >>  8) & 0xff);
      shmem[10 + i * 12] = (unsigned char)((tmp >> 16) & 0xff);
      shmem[11 + i * 12] = (unsigned char)((tmp >> 24) & 0xff);
    }
  }
  else if (data->data_type == kLivoxLidarCartesianCoordinateLowData) {
    LivoxLidarCartesianLowRawPoint *p_point_data = (LivoxLidarCartesianLowRawPoint *)data->data;
  } else if (data->data_type == kLivoxLidarSphericalCoordinateData) {
    LivoxLidarSpherPoint* p_point_data = (LivoxLidarSpherPoint *)data->data;
  }
}
union F2B {
    float f;
    uint8_t bytes[sizeof(float)];
};

void ImuDataCallback(uint32_t handle, const uint8_t dev_type,  LivoxLidarEthernetPacket* data, void*
client_data) {
  if (data == nullptr) {
    return;
  }
//  printf("Imu data callback handle:%u, data_num:%u, data_type:%u, length:%u, frame_counter:%u.¥n",
//      handle, data->dot_num, data->data_type, data->length, data->frame_cnt);
  if (data->data_type == kLivoxLidarImuData) {
    LivoxLidarImuRawPoint *p_point_data = (LivoxLidarImuRawPoint *)data->data;
//    printf("%lf ",p_point_data[0].gyro_x);
//    printf("%lf ",p_point_data[0].gyro_y);
//    printf("%lf ",p_point_data[0].gyro_z);
//    printf("%lf ",p_point_data[0].acc_x);
//    printf("%lf ",p_point_data[0].acc_y);
//    printf("%lf¥n",p_point_data[0].acc_z);
    shmem[0] = (uint8_t) data->data_type;
    union F2B f2b;
    int i;
    f2b.f = p_point_data[0].gyro_x;for(i = 0; i < 4;i++) shmem[12*96 +      i] = f2b.bytes[i];
    f2b.f = p_point_data[0].gyro_y;for(i = 0; i < 4;i++) shmem[12*96 +  4 + i] = f2b.bytes[i];
    f2b.f = p_point_data[0].gyro_z;for(i = 0; i < 4;i++) shmem[12*96 +  8 + i] = f2b.bytes[i];
    f2b.f = p_point_data[0].acc_x;for (i = 0; i < 4;i++) shmem[12*96 + 12 + i] = f2b.bytes[i];
    f2b.f = p_point_data[0].acc_y;for (i = 0; i < 4;i++) shmem[12*96 + 16 + i] = f2b.bytes[i];
    f2b.f = p_point_data[0].acc_z;for (i = 0; i < 4;i++) shmem[12*96 + 20 + i] = f2b.bytes[i];
  }
}

// void OnLidarSetIpCallback(livox_vehicle_status status, uint32_t handle, uint8_t ret_code, void*) {
//   if (status == kVehicleStatusSuccess) {
//     printf("lidar set ip slot: %d, ret_code: %d¥n",
//       slot, ret_code);
//   } else if (status == kVehicleStatusTimeout) {
//     printf("lidar set ip number timeout¥n");
//   }
// }

void WorkModeCallback(livox_status status, uint32_t handle,LivoxLidarAsyncControlResponse *response, void
*client_data) {
  if (response == nullptr) {
    return;
  }
  printf("WorkModeCallack, status:%u, handle:%u, ret_code:%u, error_key:%u",
      status, handle, response->ret_code, response->error_key);
```

```
}
void PclDataTypeCallback(livox_status status, uint32_t handle,LivoxLidarAsyncControlResponse *response, void
*client_data) {
  if (response == nullptr) {
    return;
  }
  printf("PclDataTypeCallack, status:%u, handle:%u, ret_code:%u, error_key:%u",
      status, handle, response->ret_code, response->error_key);
}

void  RebootCallback(livox_status  status,  uint32_t  handle,  LivoxLidarRebootResponse*  response,  void*
client_data) {
  if (response == nullptr) {
    return;
  }
  printf("RebootCallback, status:%u, handle:%u, ret_code:%u",
      status, handle, response->ret_code);
}

void SetIpInfoCallback(livox_status status, uint32_t handle, LivoxLidarAsyncControlResponse *response, void
*client_data) {
  if (response == nullptr) {
    return;
  }
  printf("LivoxLidarIpInfoCallback, status:%u, handle:%u, ret_code:%u, error_key:%u",
      status, handle, response->ret_code, response->error_key);

  if (response->ret_code == 0 && response->error_key == 0) {
    LivoxLidarRequestReboot(handle, RebootCallback, nullptr);
  }
}

void QueryInternalInfoCallback(livox_status status, uint32_t handle,
    LivoxLidarDiagInternalInfoResponse* response, void* client_data) {
  if (status != kLivoxLidarStatusSuccess) {
    printf("Query lidar internal info failed.¥n");
    QueryLivoxLidarInternalInfo(handle, QueryInternalInfoCallback, nullptr);
    return;
  }

  if (response == nullptr) {
    return;
  }

  uint8_t host_point_ipaddr[4] {0};
  uint16_t host_point_port = 0;
  uint16_t lidar_point_port = 0;

  uint8_t host_imu_ipaddr[4] {0};
  uint16_t host_imu_data_port = 0;
  uint16_t lidar_imu_data_port = 0;

  uint16_t off = 0;
  for (uint8_t i = 0; i < response->param_num; ++i) {
    LivoxLidarKeyValueParam* kv = (LivoxLidarKeyValueParam*)&response->data[off];
    if (kv->key == kKeyLidarPointDataHostIpCfg) {
      memcpy(host_point_ipaddr, &(kv->value[0]), sizeof(uint8_t) * 4);
      memcpy(&(host_point_port), &(kv->value[4]), sizeof(uint16_t));
      memcpy(&(lidar_point_port), &(kv->value[6]), sizeof(uint16_t));
    } else if (kv->key == kKeyLidarImuHostIpCfg) {
      memcpy(host_imu_ipaddr, &(kv->value[0]), sizeof(uint8_t) * 4);
      memcpy(&(host_imu_data_port), &(kv->value[4]), sizeof(uint16_t));
      memcpy(&(lidar_imu_data_port), &(kv->value[6]), sizeof(uint16_t));
    }
    off += sizeof(uint16_t) * 2;
    off += kv->length;
  }

  printf("Host point cloud ip addr:%u.%u.%u.%u, host point cloud port:%u, lidar point cloud port:%u.¥n",
      host_point_ipaddr[0],      host_point_ipaddr[1],      host_point_ipaddr[2],      host_point_ipaddr[3],
host_point_port, lidar_point_port);

  printf("Host imu ip addr:%u.%u.%u.%u, host imu port:%u, lidar imu port:%u.¥n",
    host_imu_ipaddr[0],  host_imu_ipaddr[1],  host_imu_ipaddr[2],  host_imu_ipaddr[3],  host_imu_data_port,
lidar_imu_data_port);

}

void LidarInfoChangeCallback(const uint32_t handle, const LivoxLidarInfo* info, void* client_data) {
  if (info == nullptr) {
```

```
    printf("lidar info change callback failed, the info is nullptr.¥n");
    return;
  }
  printf("LidarInfoChangeCallback Lidar handle: %u SN: %s¥n", handle, info->sn);

  // set the work mode to kLivoxLidarNormal, namely start the lidar
  SetLivoxLidarWorkMode(handle, kLivoxLidarNormal, WorkModeCallback, nullptr);
  SetLivoxLidarPclDataType(handle, kLivoxLidarCartesianCoordinateHighData, PclDataTypeCallback, nullptr);
  QueryLivoxLidarInternalInfo(handle, QueryInternalInfoCallback, nullptr);

  // LivoxLidarIpInfo lidar_ip_info;
  // strcpy(lidar_ip_info.ip_addr, "192.168.1.10");
  // strcpy(lidar_ip_info.net_mask, "255.255.255.0");
  // strcpy(lidar_ip_info.gw_addr, "192.168.1.1");
  // SetLivoxLidarLidarIp(handle, &lidar_ip_info, SetIpInfoCallback, nullptr);
}

void LivoxLidarPushMsgCallback(const uint32_t handle, const uint8_t dev_type, const char* info, void*
client_data) {
  struct in_addr tmp_addr;
  tmp_addr.s_addr = handle;
  std::cout << "handle: " << handle << ", ip: " << inet_ntoa(tmp_addr) << ", push msg info: " << std::endl;
  std::cout << info << std::endl;
  return;
}

int main(int argc, const char *argv[]) {
//  if (argc != 2) {
//    printf("Params Invalid, must input config path.¥n");
//    return -1;
//  }
//  const std::string path = argv[1];

  // REQUIRED, to init Livox SDK2
  if (!LivoxLidarSdkInit("config.json")) {
    printf("Livox Init Failed¥n");
    LivoxLidarSdkUninit();
    return -1;
  }

  // REQUIRED, to get point cloud data via 'PointCloudCallback'
  SetLivoxLidarPointCloudCallBack(PointCloudCallback, nullptr);

  // OPTIONAL, to get imu data via 'ImuDataCallback'
  // some lidar types DO NOT contain an imu component
  SetLivoxLidarImuDataCallback(ImuDataCallback, nullptr);

  SetLivoxLidarInfoCallback(LivoxLidarPushMsgCallback, nullptr);

  // REQUIRED, to get a handle to targeted lidar and set its work mode to NORMAL
  SetLivoxLidarInfoChangeCallback(LidarInfoChangeCallback, nullptr);

  HANDLE hFile = CreateFileMapping(INVALID_HANDLE_VALUE,NULL,PAGE_READWRITE,0,SHMEMSIZE,lpName);
  if(hFile == NULL) {
    printf("Could not open file mapping object¥n");
  }
  shmem = (uint8_t *)MapViewOfFile(hFile,FILE_MAP_ALL_ACCESS,0,0,SHMEMSIZE);


#ifdef WIN32
  Sleep(300000);
#else
  sleep(300);
#endif
  LivoxLidarSdkUninit();
  UnmapViewOfFile(shmem);
  CloseHandle(hFile);
  printf("Livox Quick Start Demo End!¥n");
  return 0;
}
```

修正したら、`msbuild` によりコンパイルする。

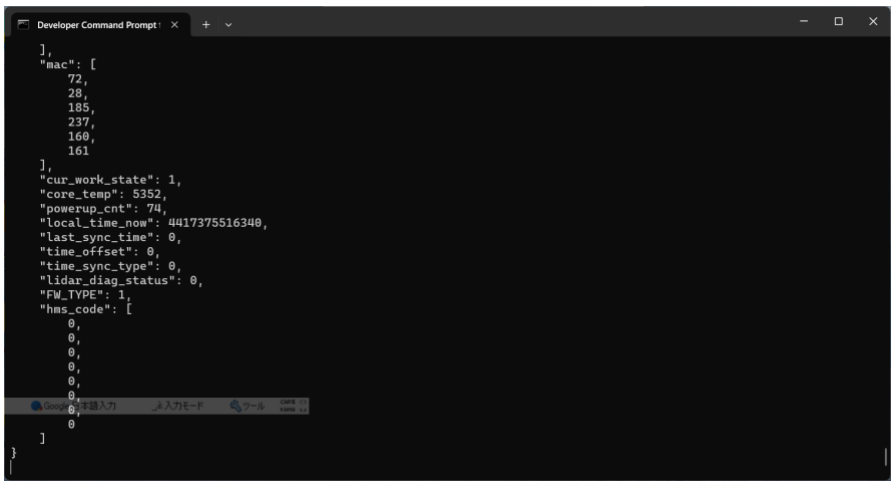
```
> msbuild livox_sdk2.sln /p:Configuration=Release /p:Platform=x64 /p:RuntimeLibrary=MultiThreadedDLL
```

コンパイルできたら、`Release` フォルダに移動し、実行する。

上記の変更では、`config.json` を直接指定している。そのため、コマンドプロンプト上で、

```
> livox_lidar_quick_start.exe
```

として実行できる。うまく動いている場合、このような情報が表示される。



MATLAB　MEX プログラムの作成

MATLAB コンパイラの指定

ここから MATLAB のコマンドウィンドウ

```
>> mex -setup C++
MEX は C++ 言語のコンパイルに 'Microsoft Visual C++ 2022' を使用するよう設定されています。
```

とした。

MEX ファイルのコンパイル

カレントフォルダに下記の C プログラムを保存し、

```
>> mex sharedmem2.c
```

でコンパイルする。

このプログラムでは、シリアライズされた、Point Cloud と imu の情報を MATLAB 行列（3×(96+2)）として取り込むプログラムである。

¥Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_start¥sharedmem2.c

```
#include"mex.h"
#include <stdio.h>
#include <windows.h>

LPCTSTR lpName = "(c)2024 www.gerox.com";
int SHMEMSIZE = 96*3*4+6*4+1;
static int initialized = 0;
static unsigned char *shmem;
static HANDLE hFile;

void cLeanup(void) {
          if(initialized != 0) {
                    mexPrintf("MEX-file is safely terminated.¥n");
                    UnmapViewOfFile(shmem);
                    CloseHandle(hFile);
                    initialized = 0;
          }
}

void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray * prhs[]) {
          int i;
          if(nrhs != 0) {mexErrMsgTxt("zero input must be required for shared memory.");return;}
          if(initialized == 0) {
                    mexPrintf("Shared Memory Access for MATLAB by Gerox(c) 2024¥n");
```

```
                    mexPrintf("http://www.gerox.com Build %s¥n",__DATE__);
                    hFile                                                                                    =
CreateFileMapping(INVALID_HANDLE_VALUE,NULL,PAGE_READWRITE,0,SHMEMSIZE,lpName);
                    if(hFile == NULL) {
                              mexErrMsgTxt("Could not open file mapping object¥n");
                              return;
                    }
                    shmem = (unsigned char *)MapViewOfFile(hFile,FILE_MAP_ALL_ACCESS,0,0,SHMEMSIZE);
                    initialized = 1;
                    mexMakeMemoryPersistent(shmem);
                    mexAtExit(cLeanup);
          }
          if(nlhs == 1) {
                    plhs[0] = mxCreateDoubleMatrix(3,96+2,mxREAL);
                    double *data = mxGetPr(plhs[0]);
                    for(i = 0;i < 96;i++) {
                              int tmp;
                              tmp = 0;
                              tmp |= ( shmem[ 0 + i * 12] & 0xff);
                              tmp |= ((shmem[ 1 + i * 12] & 0xff) << 8);
                              tmp |= ((shmem[ 2 + i * 12] & 0xff) << 16);
                              tmp |= ((shmem[ 3 + i * 12] & 0xff) << 24);
                              data[i * 3] = (double)tmp;
//                        mexPrintf("%lg ",(double)tmp);
                              tmp = 0;
                              tmp |= ( shmem[ 4 + i * 12] & 0xff);
                              tmp |= ((shmem[ 5 + i * 12] & 0xff) << 8);
                              tmp |= ((shmem[ 6 + i * 12] & 0xff) << 16);
                              tmp |= ((shmem[ 7 + i * 12] & 0xff) << 24);
                              data[i * 3 + 1] = (double)tmp;
//                        mexPrintf("%lg ",(double)tmp);
                              tmp = 0;
                              tmp |= ( shmem[ 8 + i * 12] & 0xff);
                              tmp |= ((shmem[ 9 + i * 12] & 0xff) << 8);
                              tmp |= ((shmem[10 + i * 12] & 0xff) << 16);
                              tmp |= ((shmem[11 + i * 12] & 0xff) << 24);
                              data[i * 3 + 2] = (double)tmp;
//                        mexPrintf("%lg¥n",(double)tmp);
                    }
                    union _f2b {
                              float f;
                              unsigned char bytes[sizeof(float)];
                    } f2b;
                    for(i    =    0;    i    <    sizeof(float);    i++)    f2b.bytes[i]=shmem[96*3*4    +
i];data[96*3+0]=f2b.f;
                    for(i   =   0;   i   <   sizeof(float);   i++)   f2b.bytes[i]=shmem[96*3*4   +     4   +
i];data[96*3+1]=f2b.f;
                    for(i   =   0;   i   <   sizeof(float);   i++)   f2b.bytes[i]=shmem[96*3*4   +     8   +
i];data[96*3+2]=f2b.f;
                    for(i   =   0;   i   <   sizeof(float);   i++)   f2b.bytes[i]=shmem[96*3*4   +   12   +
i];data[96*3+3]=f2b.f;
                    for(i   =   0;   i   <   sizeof(float);   i++)   f2b.bytes[i]=shmem[96*3*4   +   16   +
i];data[96*3+4]=f2b.f;
                    for(i   =   0;   i   <   sizeof(float);   i++)   f2b.bytes[i]=shmem[96*3*4   +   20   +
i];data[96*3+5]=f2b.f;
                    return;
          }
          mexPrintf("------------------------------¥n");
          mexPrintf("Usage: sharedmem2            ¥n");
          mexPrintf("data=sharedmem2;             ¥n");
          mexPrintf("Shared Memory Access for MATLAB by Gerox(c) 2024¥n");
          mexPrintf("http://www.gerox.com Build %s¥n",__DATE__);
          mexPrintf("------------------------------¥n");
          mexErrMsgTxt("One input and one output must be required.");
}
```

MATLAB のコマンドウィンドウで一度実行すると、メモリに常駐する。開放するには、clear all コマンドを使う。

```
>> a=sharedmem2;
>> clear all
MEX-file is safely terminated.
```

MATLAB からのデータ取り込み

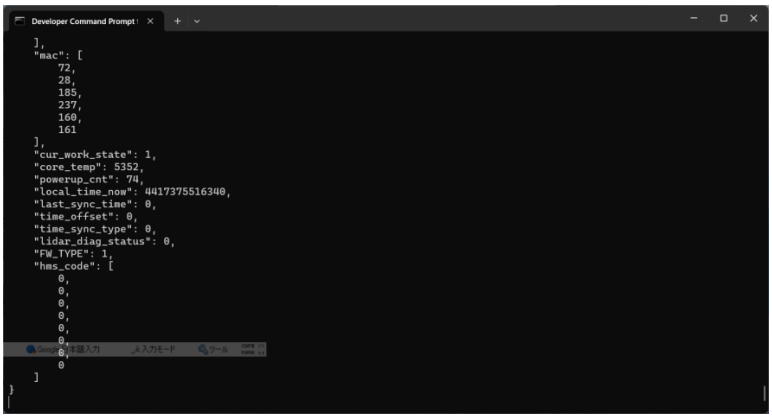
a=sharedmem2 で取り込まれたデータは、行列変数 a に入る。lidar=a(:,1:96)で Point Cloud データ、imu=a(:,97:98)で imu データを取り込むことができる。

a=sharedmem2 で取り込むデータは Developer Command Prompt で¥Livox-SDK2-1.2.5¥build¥samples¥livox_lidar_quick_start¥Release の
>> livox_lidar_quick_start.exe
を実行しているときのリアルタイムデータを取り込んでいる。そのため、mid360 に接続し、上記のコマンドを実行した状態で下記のコマンドを MATLAB で実行する。

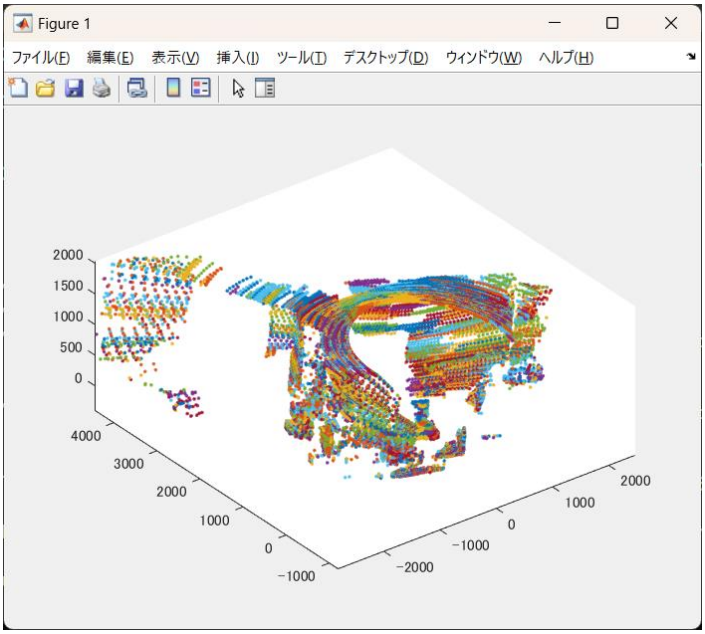
¥Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_start¥untitled2.m

```
clear all;close all
a=sharedmem2;
lidar=a(:,1:96);
imu=a(:,97:98);
plot3(lidar(1,:),lidar(2,:),lidar(3,:),'.');
axis equal;
hold on
for i=1:1000
    a = sharedmem2;
    lidar=a(:,1:96);
    imu=a(:,97:98)
    plot3(lidar(1,:),lidar(2,:),lidar(3,:),'.');
    drawnow
end
```

とする。




```
imu =
   -0.0016    0.0649
   -0.0046   -0.0765
   -0.0057    1.0135
imu =
   -0.0048    0.0644
    0.0050   -0.0757
   -0.0057    0.9830
imu =
   -0.0016    0.0646
   -0.0046   -0.0753
   -0.0089    1.0140
imu =
   -0.0069    0.0619
    0.0061   -0.0760
   -0.0046    0.9823
```



Point Cloud データは、96 点毎に、色を変えて表示される。imu(1:3,1)が 3 軸ジャイロ、imu(1:3,2)は、3 軸加速度を表している。なお、imu(3,2)は、ほぼ、1[g]を示しているので、つまり、z 軸下向きが正であることを意味している。（IMU は、NED 座標系（x-north, y-east,z-down）?）これで、一応、動作確認ができたので、処理速度を重視した実装を行っていく。

汎用性、高速化を考慮した Livox SDK2 のサンプルプログラムの修正

共有メモリの再デザイン

IMU と LiDAR の Timestamp の追加

IMU と LiDAR は、非同期取り込みであるため、Timestamp はそれぞれ異なる。そのため、それぞれ IMU、LiDAR から Callback 関数が呼ばれたとき pupdate,imupdate にシーケンス時間を保存するようにする。シーケンス時間は、nsec 単位で、64 ビットデータであるため、共有メモリサイズ 8 バイトを確保する。

```
int SHMEMSIZE = 1+8+8+(3+3)*4+1+(4*3+1+1)*96;
```

timestamp_type	data_type	time type
1	uint64_t	ns

Timestamp の Data type は、uint64_t であるが、Livox のヘッダファイル内では、uint8[8] として定義されているので、両方の型からアクセスできるように

```
 union _Data8Union {
    uint64_t u64;
    uint8_t u8[8];
};
```

を追加する。MATLAB の出力では、5 行(2+96)列で出力することを想定し、shmem を以下のように定義する。

Data	Field	Type	shmem　(i=0～95)	MATLAB 行列
Data Type		uint8_t	0	1,1
pupdate		uint8_t[8]	1,2,3,4,5,6,7,8	2,1
imupdate		uint8_t[8]	9,10,11,12,13,14,15,16	4,2
IMU Data	gyro_x	float	17,18,19,20	3,1
	gyro_y	float	21,22,23,24	4,1
	gyro_z	float	25,26,27,28	5,1
	acc_x	float	29,30,31,32	1,2
	acc_y	float	33,34,35,36	2,2
	acc_z	float	37,38,39,40	3,2
	reserved			5,2
Data Type1	x	int32_t	41+i*14	1,3+i
	y	int32_t	41+i*14+4	2,3+i
	z	int32_t	41+i*14+8	3,3+i
	reflectivity	uint8_t	41+i*14+12	4,3+i
	tag	uint8_t	41+i*14+13	5,3+i
Data Type2	x	int16_t	41+i*8	1,3+i
	y	int16_t	41+i*8+2	2,3+i
	z	int16_t	41+i*8+4	3,3+i
	reflectivity	uint8_t	41+i*8+6	4,3+i
	tag	uint8_t	41+i*8+7	5,3+i
Data Type3	depth	int32_t	41+i*10	1,3+i

	theta	int16_t	41+i*10+4	2,3+i
	phi	int16_t	41+i*10+6	3,3+i
	reflectivity	uint8_t	41+i*10+8	4,3+i
	tag	uint8_t	41+i*10+9	5,3+i

¥Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_startv4¥main.cpp

```
//
// The MIT License (MIT)
//
// Copyright (c) 2022 Livox. All rights reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in
// all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.
//

#include "livox_lidar_def.h"
#include "livox_lidar_api.h"

#ifdef _WIN32
#include <winsock2.h>
#else
#include <unistd.h>
#include <arpa/inet.h>
#endif

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <thread>
#include <chrono>
#include <iostream>

#include <mutex>
#include <condition_variable>
#include <map>

#include <csignal>
#include <stdint.h>

std::condition_variable quit_condition;
std::mutex mtx;

LPCTSTR lpName = "(c)2024 www.gerox.com v5";
int SHMEMSIZE = 1+8+8+(3+3)*4+(4*3+1+1)*96;
static int initialized = 0;
static uint8_t *shmem;
static HANDLE hFile;

union _Data2Union {
    int16_t i16;
    uint16_t u16;
    uint8_t u8[2];
};

union _Data4Union {
    float f;
    int32_t i32;
    uint32_t u32;
    uint8_t u8[4];
};
```

```
union _Data8Union {
    uint64_t u64;
    uint8_t u8[8];
};

union _Data8Union Uint64Bytes;
union _Data4Union FloatBytes;
union _Data4Union Int32Bytes;
union _Data4Union Uint32Bytes;
union _Data2Union Int16Bytes;
union _Data2Union Uint16Bytes;

void PointCloudCallback(uint32_t handle, const uint8_t dev_type, LivoxLidarEthernetPacket* data, void*
client_data) {
  if (data == nullptr) {
    return;
  }
//  printf("point cloud handle: %u, data_num: %d, data_type: %d, length: %d, frame_counter: %d¥n",
//      handle, data->dot_num, data->data_type, data->length, data->frame_cnt);
  if (data->data_type == kLivoxLidarCartesianCoordinateHighData) {
    shmem[0] = (uint8_t) data->data_type;
    shmem[1] = (uint8_t) data->timestamp[0];
    shmem[2] = (uint8_t) data->timestamp[1];
    shmem[3] = (uint8_t) data->timestamp[2];
    shmem[4] = (uint8_t) data->timestamp[3];
    shmem[5] = (uint8_t) data->timestamp[4];
    shmem[6] = (uint8_t) data->timestamp[5];
    shmem[7] = (uint8_t) data->timestamp[6];
    shmem[8] = (uint8_t) data->timestamp[7];
          LivoxLidarCartesianHighRawPoint *p_point_data = (LivoxLidarCartesianHighRawPoint *)data->data;
    for (int i = 0;i < 96;i++) {
      int ind = 41 + i * 14;
      Int32Bytes.i32 = p_point_data[i].x;
      shmem[ind    ] = (uint8_t) Int32Bytes.u8[0];
      shmem[ind + 1] = (uint8_t) Int32Bytes.u8[1];
      shmem[ind + 2] = (uint8_t) Int32Bytes.u8[2];
      shmem[ind + 3] = (uint8_t) Int32Bytes.u8[3];
//      printf("%lg ",(double)Int32Bytes.value);
      Int32Bytes.i32 = p_point_data[i].y;
      shmem[ind + 4] = (uint8_t) Int32Bytes.u8[0];
      shmem[ind + 5] = (uint8_t) Int32Bytes.u8[1];
      shmem[ind + 6] = (uint8_t) Int32Bytes.u8[2];
      shmem[ind + 7] = (uint8_t) Int32Bytes.u8[3];
//      printf("%lg ",(double)Int32Bytes.value);
      Int32Bytes.i32 = p_point_data[i].z;
      shmem[ind + 8] = (uint8_t) Int32Bytes.u8[0];
      shmem[ind + 9] = (uint8_t) Int32Bytes.u8[1];
      shmem[ind +10] = (uint8_t) Int32Bytes.u8[2];
      shmem[ind +11] = (uint8_t) Int32Bytes.u8[3];
//      printf("%lg ",(double)Int32Bytes.value);
      shmem[ind +12] = (uint8_t) p_point_data[i].reflectivity;
      shmem[ind +13] = (uint8_t) p_point_data[i].tag;
//      printf("%x %x¥n", p_point_data[i].reflectivity, p_point_data[i].tag);
    }
  } else {
    if (data->data_type == kLivoxLidarCartesianCoordinateLowData) {
      shmem[0] = (uint8_t) data->data_type;
      shmem[1] = (uint8_t) data->timestamp[0];
      shmem[2] = (uint8_t) data->timestamp[1];
      shmem[3] = (uint8_t) data->timestamp[2];
      shmem[4] = (uint8_t) data->timestamp[3];
      shmem[5] = (uint8_t) data->timestamp[4];
      shmem[6] = (uint8_t) data->timestamp[5];
      shmem[7] = (uint8_t) data->timestamp[6];
      shmem[8] = (uint8_t) data->timestamp[7];
      LivoxLidarCartesianLowRawPoint *p_point_data = (LivoxLidarCartesianLowRawPoint *)data->data;
      for (int i = 0;i < 96;i++) {
        int ind = 41 + i * 8;
        Int16Bytes.i16 = p_point_data[i].x;
        shmem[ind    ] = (uint8_t) Int16Bytes.u8[0];
        shmem[ind + 1] = (uint8_t) Int16Bytes.u8[1];
        Int16Bytes.i16 = p_point_data[i].y;
        shmem[ind + 2] = (uint8_t) Int16Bytes.u8[0];
        shmem[ind + 3] = (uint8_t) Int32Bytes.u8[1];
        Int16Bytes.i16 = p_point_data[i].z;
        shmem[ind + 4] = (uint8_t) Int16Bytes.u8[0];
        shmem[ind + 5] = (uint8_t) Int16Bytes.u8[1];
        shmem[ind + 6] = (uint8_t) p_point_data[i].reflectivity;
        shmem[ind + 7] = (uint8_t) p_point_data[i].tag;
      }
```

```
    } else {
            if (data->data_type == kLivoxLidarSphericalCoordinateData) {
        shmem[0] = (uint8_t) data->data_type;
        shmem[1] = (uint8_t) data->timestamp[0];
        shmem[2] = (uint8_t) data->timestamp[1];
        shmem[3] = (uint8_t) data->timestamp[2];
        shmem[4] = (uint8_t) data->timestamp[3];
        shmem[5] = (uint8_t) data->timestamp[4];
        shmem[6] = (uint8_t) data->timestamp[5];
        shmem[7] = (uint8_t) data->timestamp[6];
        shmem[8] = (uint8_t) data->timestamp[7];
        LivoxLidarSpherPoint* p_point_data = (LivoxLidarSpherPoint *)data->data;
        for (int i = 0;i < 96;i++) {
          int ind = 41 + i * 10;
          Uint32Bytes.u32 = p_point_data[i].depth;
          shmem[ind    ] = (uint8_t) Uint32Bytes.u8[0];
          shmem[ind + 1] = (uint8_t) Uint32Bytes.u8[1];
          shmem[ind + 2] = (uint8_t) Uint32Bytes.u8[2];
          shmem[ind + 3] = (uint8_t) Uint32Bytes.u8[3];
          Uint16Bytes.u16 = p_point_data[i].theta;
          shmem[ind + 4] = (uint8_t) Uint16Bytes.u8[0];
          shmem[ind + 5] = (uint8_t) Uint16Bytes.u8[1];
          Uint16Bytes.u16 = p_point_data[i].phi;
          shmem[ind + 6] = (uint8_t) Uint16Bytes.u8[0];
          shmem[ind + 7] = (uint8_t) Uint16Bytes.u8[1];
          shmem[ind + 8] = (uint8_t) p_point_data[i].reflectivity;
          shmem[ind + 9] = (uint8_t) p_point_data[i].tag;
        }
      }
    }
  }
}
void ImuDataCallback(uint32_t handle, const uint8_t dev_type,  LivoxLidarEthernetPacket* data, void*
client_data) {
  if (data == nullptr) {
    return;
  }
//  printf("Imu data callback handle:%u, data_num:%u, data_type:%u, length:%u, frame_counter:%u.¥n",
//      handle, data->dot_num, data->data_type, data->length, data->frame_cnt);
  if (data->data_type == kLivoxLidarImuData) {
    shmem[0] = (uint8_t) data->data_type;
    shmem[9] = (uint8_t) data->timestamp[0];
    shmem[10] = (uint8_t) data->timestamp[1];
    shmem[11] = (uint8_t) data->timestamp[2];
    shmem[12] = (uint8_t) data->timestamp[3];
    shmem[13] = (uint8_t) data->timestamp[4];
    shmem[14] = (uint8_t) data->timestamp[5];
    shmem[15] = (uint8_t) data->timestamp[6];
    shmem[16] = (uint8_t) data->timestamp[7];
    LivoxLidarImuRawPoint *p_point_data = (LivoxLidarImuRawPoint *)data->data;
/*
    printf("%lf ",p_point_data[0].gyro_x);
    printf("%lf ",p_point_data[0].gyro_y);
    printf("%lf ",p_point_data[0].gyro_z);
    printf("%lf ",p_point_data[0].acc_x);
    printf("%lf ",p_point_data[0].acc_y);
    printf("%lf¥n",p_point_data[0].acc_z);
*/
    FloatBytes.f = p_point_data[0].gyro_x;
    shmem[17] = FloatBytes.u8[0];
    shmem[18] = FloatBytes.u8[1];
    shmem[19] = FloatBytes.u8[2];
    shmem[20] = FloatBytes.u8[3];
    FloatBytes.f = p_point_data[0].gyro_y;
    shmem[21] = FloatBytes.u8[0];
    shmem[22] = FloatBytes.u8[1];
    shmem[23] = FloatBytes.u8[2];
    shmem[24] = FloatBytes.u8[3];
    FloatBytes.f = p_point_data[0].gyro_z;
    shmem[25] = FloatBytes.u8[0];
    shmem[26] = FloatBytes.u8[1];
    shmem[27] = FloatBytes.u8[2];
    shmem[28] = FloatBytes.u8[3];
    FloatBytes.f = p_point_data[0].acc_x;
    shmem[29] = FloatBytes.u8[0];
    shmem[30] = FloatBytes.u8[1];
    shmem[31] = FloatBytes.u8[2];
    shmem[32] = FloatBytes.u8[3];
    FloatBytes.f = p_point_data[0].acc_y;
    shmem[33] = FloatBytes.u8[0];
```

```
    shmem[34] = FloatBytes.u8[1];
    shmem[35] = FloatBytes.u8[2];
    shmem[36] = FloatBytes.u8[3];
    FloatBytes.f = p_point_data[0].acc_z;
    shmem[37] = FloatBytes.u8[0];
    shmem[38] = FloatBytes.u8[1];
    shmem[39] = FloatBytes.u8[2];
    shmem[40] = FloatBytes.u8[3];
  }
}

// void OnLidarSetIpCallback(livox_vehicle_status status, uint32_t handle, uint8_t ret_code, void*) {
//   if (status == kVehicleStatusSuccess) {
//     printf("lidar set ip slot: %d, ret_code: %d¥n",
//       slot, ret_code);
//   } else if (status == kVehicleStatusTimeout) {
//     printf("lidar set ip number timeout¥n");
//   }
// }

void WorkModeCallback(livox_status status, uint32_t handle,LivoxLidarAsyncControlResponse *response, void
*client_data) {
  if (response == nullptr) {
    return;
  }
  printf("WorkModeCallack, status:%u, handle:%u, ret_code:%u, error_key:%u",
      status, handle, response->ret_code, response->error_key);

}
void PclDataTypeCallback(livox_status status, uint32_t handle,LivoxLidarAsyncControlResponse *response, void
*client_data) {
  if (response == nullptr) {
    return;
  }
  printf("PclDataTypeCallack, status:%u, handle:%u, ret_code:%u, error_key:%u",
      status, handle, response->ret_code, response->error_key);
}

void RebootCallback(livox_status status, uint32_t handle, LivoxLidarRebootResponse* response, void*
client_data) {
  if (response == nullptr) {
    return;
  }
  printf("RebootCallback, status:%u, handle:%u, ret_code:%u",
      status, handle, response->ret_code);
}

void SetIpInfoCallback(livox_status status, uint32_t handle, LivoxLidarAsyncControlResponse *response, void
*client_data) {
  if (response == nullptr) {
    return;
  }
  printf("LivoxLidarIpInfoCallback, status:%u, handle:%u, ret_code:%u, error_key:%u",
      status, handle, response->ret_code, response->error_key);

  if (response->ret_code == 0 && response->error_key == 0) {
    LivoxLidarRequestReboot(handle, RebootCallback, nullptr);
  }
}

void QueryInternalInfoCallback(livox_status status, uint32_t handle,
    LivoxLidarDiagInternalInfoResponse* response, void* client_data) {
  if (status != kLivoxLidarStatusSuccess) {
    printf("Query lidar internal info failed.¥n");
    QueryLivoxLidarInternalInfo(handle, QueryInternalInfoCallback, nullptr);
    return;
  }

  if (response == nullptr) {
    return;
  }

  uint8_t host_point_ipaddr[4] {0};
  uint16_t host_point_port = 0;
  uint16_t lidar_point_port = 0;

  uint8_t host_imu_ipaddr[4] {0};
  uint16_t host_imu_data_port = 0;
  uint16_t lidar_imu_data_port = 0;
```

```
  uint16_t off = 0;
  for (uint8_t i = 0; i < response->param_num; ++i) {
    LivoxLidarKeyValueParam* kv = (LivoxLidarKeyValueParam*)&response->data[off];
    if (kv->key == kKeyLidarPointDataHostIpCfg) {
      memcpy(host_point_ipaddr, &(kv->value[0]), sizeof(uint8_t) * 4);
      memcpy(&(host_point_port), &(kv->value[4]), sizeof(uint16_t));
      memcpy(&(lidar_point_port), &(kv->value[6]), sizeof(uint16_t));
    } else if (kv->key == kKeyLidarImuHostIpCfg) {
      memcpy(host_imu_ipaddr, &(kv->value[0]), sizeof(uint8_t) * 4);
      memcpy(&(host_imu_data_port), &(kv->value[4]), sizeof(uint16_t));
      memcpy(&(lidar_imu_data_port), &(kv->value[6]), sizeof(uint16_t));
    }
    off += sizeof(uint16_t) * 2;
    off += kv->length;
  }

  printf("Host point cloud ip addr:%u.%u.%u.%u, host point cloud port:%u, lidar point cloud port:%u.¥n",
      host_point_ipaddr[0],      host_point_ipaddr[1],      host_point_ipaddr[2],      host_point_ipaddr[3],
host_point_port, lidar_point_port);

  printf("Host imu ip addr:%u.%u.%u.%u, host imu port:%u, lidar imu port:%u.¥n",
    host_imu_ipaddr[0],  host_imu_ipaddr[1],  host_imu_ipaddr[2],  host_imu_ipaddr[3],  host_imu_data_port,
lidar_imu_data_port);

}

void LidarInfoChangeCallback(const uint32_t handle, const LivoxLidarInfo* info, void* client_data) {
  if (info == nullptr) {
    printf("lidar info change callback failed, the info is nullptr.¥n");
    return;
  }
  printf("LidarInfoChangeCallback Lidar handle: %u SN: %s¥n", handle, info->sn);

  // set the work mode to kLivoxLidarNormal, namely start the lidar
  SetLivoxLidarWorkMode(handle, kLivoxLidarNormal, WorkModeCallback, nullptr);
  SetLivoxLidarPclDataType(handle, kLivoxLidarCartesianCoordinateHighData, PclDataTypeCallback, nullptr);
//  SetLivoxLidarPclDataType(handle, kLivoxLidarCartesianCoordinateLowData,PclDataTypeCallback, nullptr);
//  SetLivoxLidarPclDataType(handle, kLivoxLidarSphericalCoordinateData,PclDataTypeCallback, nullptr);
  QueryLivoxLidarInternalInfo(handle, QueryInternalInfoCallback, nullptr);

  // LivoxLidarIpInfo lidar_ip_info;
  // strcpy(lidar_ip_info.ip_addr, "192.168.1.10");
  // strcpy(lidar_ip_info.net_mask, "255.255.255.0");
  // strcpy(lidar_ip_info.gw_addr, "192.168.1.1");
  // SetLivoxLidarLidarIp(handle, &lidar_ip_info, SetIpInfoCallback, nullptr);
}
void Stop(int signal) {
  quit_condition.notify_all();
}

void  LivoxLidarPushMsgCallback(const  uint32_t  handle,  const  uint8_t  dev_type,  const  char*  info,  void*
client_data) {
  struct in_addr tmp_addr;
  tmp_addr.s_addr = handle;
  std::cout << "handle: " << handle << ", ip: " << inet_ntoa(tmp_addr) << ", push msg info: " << std::endl;
  std::cout << info << std::endl;
  return;
}

int main(int argc, const char *argv[]) {
//  if (argc != 2) {
//    printf("Params Invalid, must input config path.¥n");
//    return -1;
//  }
//  const std::string path = argv[1];

          HANDLE hFile = CreateFileMapping(INVALID_HANDLE_VALUE,NULL,PAGE_READWRITE,0,SHMEMSIZE,lpName);
          if(hFile == NULL) {
                    printf("Could not open file mapping object¥n");
          }
          shmem = (uint8_t *)MapViewOfFile(hFile,FILE_MAP_ALL_ACCESS,0,0,SHMEMSIZE);

  // REQUIRED, to init Livox SDK2
  if (!LivoxLidarSdkInit("config.json")) {
    printf("Livox Init Failed¥n");
    LivoxLidarSdkUninit();
    return -1;
  }
```

```
  // REQUIRED, to get point cloud data via 'PointCloudCallback'
  SetLivoxLidarPointCloudCallBack(PointCloudCallback, nullptr);

  // OPTIONAL, to get imu data via 'ImuDataCallback'
  // some lidar types DO NOT contain an imu component
  SetLivoxLidarImuDataCallback(ImuDataCallback, nullptr);

  SetLivoxLidarInfoCallback(LivoxLidarPushMsgCallback, nullptr);

  // REQUIRED, to get a handle to targeted lidar and set its work mode to NORMAL
  SetLivoxLidarInfoChangeCallback(LidarInfoChangeCallback, nullptr);

  // capture Ctrl + C signal.
  std::signal(SIGINT, Stop);

  std::unique_lock<std::mutex> lock(mtx);
  quit_condition.wait(lock);

  LivoxLidarSdkUninit();
  UnmapViewOfFile(shmem);
  CloseHandle(hFile);
  printf("Livox Quick Start Demo End!¥n");
  return 0;
}
```

赤字の部分

```
  SetLivoxLidarPclDataType(handle, kLivoxLidarCartesianCoordinateHighData, PclDataTypeCallback, nullptr);
//  SetLivoxLidarPclDataType(handle, kLivoxLidarCartesianCoordinateLowData,PclDataTypeCallback, nullptr);
//  SetLivoxLidarPclDataType(handle, kLivoxLidarSphericalCoordinateData,PclDataTypeCallback, nullptr);
```

のコメントを変更すると、取り込む Data Type の変更ができる。

MATLAB　MEX プログラムの更新

カレントフォルダに下記の C プログラムを保存し、

```
>> mex sharedmemv5.cpp
```

でコンパイルする。このプログラムでは、シリアライズされた、Point Cloud と imu データを MATLAB 行列として取り込む。

¥Livox-SDK2-1.2.4¥samples¥livox_lidar_quick_startv3¥sharedmemv5.cpp

```
#include"mex.h"
#include <stdio.h>
#include <windows.h>
#include <stdint.h>
#define _USE_MATH_DEFINES
#include <math.h>
LPCTSTR lpName = "(c)2024 www.gerox.com v5";
int SHMEMSIZE = 1+8+8+(3+3)*4+(4*3+1+1)*96;
static int initialized = 0;
static uint8_t *shmem;
static HANDLE hFile;

union _Data2Union {
    int16_t i16;
    uint16_t u16;
    uint8_t u8[2];
};

union _Data4Union {
    float f;
    int32_t i32;
    uint32_t u32;
    uint8_t u8[4];
};

 union _Data8Union {
    uint32_t u32;
    uint64_t u64;
    uint8_t u8[8];
};

union _Data8Union Uint64Bytes;
union _Data4Union FloatBytes;
union _Data4Union Int32Bytes;
union _Data4Union Uint32Bytes;
```

```
union _Data2Union Int16Bytes;
union _Data2Union Uint16Bytes;

void cLeanup(void) {
 if(initialized != 0) {
  mexPrintf("MEX-file is safely terminated.¥n");
  UnmapViewOfFile(shmem);
  CloseHandle(hFile);
  initialized = 0;
 }
}
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray * prhs[]) {
 int i;
 if(nrhs != 0) {
           mexErrMsgTxt("zero input must be required for shared memory.");
           return;
 }
 if(initialized == 0) {
  mexPrintf("Shared Memory Access for MATLAB by Gerox(c) 2024¥n");
  mexPrintf("http://www.gerox.com Build %s¥n",__DATE__);
  hFile = CreateFileMapping(INVALID_HANDLE_VALUE,NULL,PAGE_READWRITE,0,SHMEMSIZE,lpName);
  if(hFile == NULL) {
   mexErrMsgTxt("Could not open file mapping object¥n");
   return;
  }
  shmem = (uint8_t *)MapViewOfFile(hFile,FILE_MAP_ALL_ACCESS,0,0,SHMEMSIZE);
  initialized = 1;
  mexMakeMemoryPersistent(shmem);
  mexAtExit(cLeanup);
 }
 if(nlhs == 1) {
  plhs[0] = mxCreateDoubleMatrix(5,96 + 2,mxREAL);
  double *data = mxGetPr(plhs[0]);
  data[0] = (double) shmem[0];//  DataType
  switch ((int)data[0]) {
   case 0:
    Uint64Bytes.u8[0] = (uint8_t) shmem[1];
    Uint64Bytes.u8[1] = (uint8_t) shmem[2];
    Uint64Bytes.u8[2] = (uint8_t) shmem[3];
    Uint64Bytes.u8[3] = (uint8_t) shmem[4];
    Uint64Bytes.u8[4] = (uint8_t) shmem[5];
    Uint64Bytes.u8[5] = (uint8_t) shmem[6];
    Uint64Bytes.u8[6] = (uint8_t) shmem[7];
    Uint64Bytes.u8[7] = (uint8_t) shmem[8];
    data[1] = (double)Uint64Bytes.u64;
    Uint64Bytes.u8[0] = (uint8_t) shmem[9];
    Uint64Bytes.u8[1] = (uint8_t) shmem[10];
    Uint64Bytes.u8[2] = (uint8_t) shmem[11];
    Uint64Bytes.u8[3] = (uint8_t) shmem[12];
    Uint64Bytes.u8[4] = (uint8_t) shmem[13];
    Uint64Bytes.u8[5] = (uint8_t) shmem[14];
    Uint64Bytes.u8[6] = (uint8_t) shmem[15];
    Uint64Bytes.u8[7] = (uint8_t) shmem[16];
    data[8] = (double)Uint64Bytes.u64;
    FloatBytes.u8[0] = (uint8_t) shmem[17];
    FloatBytes.u8[1] = (uint8_t) shmem[18];
    FloatBytes.u8[2] = (uint8_t) shmem[19];
    FloatBytes.u8[3] = (uint8_t) shmem[20];
    data[2] = (double)FloatBytes.f;
    FloatBytes.u8[0] = (uint8_t) shmem[21];
    FloatBytes.u8[1] = (uint8_t) shmem[22];
    FloatBytes.u8[2] = (uint8_t) shmem[23];
    FloatBytes.u8[3] = (uint8_t) shmem[24];
    data[3] = (double)FloatBytes.f;
    FloatBytes.u8[0] = (uint8_t) shmem[25];
    FloatBytes.u8[1] = (uint8_t) shmem[26];
    FloatBytes.u8[2] = (uint8_t) shmem[27];
    FloatBytes.u8[3] = (uint8_t) shmem[28];
    data[4] = (double)FloatBytes.f;
    FloatBytes.u8[0] = (uint8_t) shmem[29];
    FloatBytes.u8[1] = (uint8_t) shmem[30];
    FloatBytes.u8[2] = (uint8_t) shmem[31];
    FloatBytes.u8[3] = (uint8_t) shmem[32];
    data[5] = (double)FloatBytes.f;
    FloatBytes.u8[0] = (uint8_t) shmem[33];
    FloatBytes.u8[1] = (uint8_t) shmem[34];
    FloatBytes.u8[2] = (uint8_t) shmem[35];
    FloatBytes.u8[3] = (uint8_t) shmem[36];
    data[6] = (double)FloatBytes.f;
    FloatBytes.u8[0] = (uint8_t) shmem[37];
```

```
    FloatBytes.u8[1] = (uint8_t) shmem[38];
    FloatBytes.u8[2] = (uint8_t) shmem[39];
    FloatBytes.u8[3] = (uint8_t) shmem[40];
    data[7] = (double)FloatBytes.f;
   break;
   case 1:
    Uint64Bytes.u8[0] = (uint8_t) shmem[1];
    Uint64Bytes.u8[1] = (uint8_t) shmem[2];
    Uint64Bytes.u8[2] = (uint8_t) shmem[3];
    Uint64Bytes.u8[3] = (uint8_t) shmem[4];
    Uint64Bytes.u8[4] = (uint8_t) shmem[5];
    Uint64Bytes.u8[5] = (uint8_t) shmem[6];
    Uint64Bytes.u8[6] = (uint8_t) shmem[7];
    Uint64Bytes.u8[7] = (uint8_t) shmem[8];
    data[1] = (double)Uint64Bytes.u64;
    Uint64Bytes.u8[0] = (uint8_t) shmem[9];
    Uint64Bytes.u8[1] = (uint8_t) shmem[10];
    Uint64Bytes.u8[2] = (uint8_t) shmem[11];
    Uint64Bytes.u8[3] = (uint8_t) shmem[12];
    Uint64Bytes.u8[4] = (uint8_t) shmem[13];
    Uint64Bytes.u8[5] = (uint8_t) shmem[14];
    Uint64Bytes.u8[6] = (uint8_t) shmem[15];
    Uint64Bytes.u8[7] = (uint8_t) shmem[16];
    data[8] = (double)Uint64Bytes.u64;
    for (i = 0;i < 96;i++) {
     int ind = 41 + i * 14;
     Int32Bytes.u8[0] = (uint8_t) shmem[ind    ];
     Int32Bytes.u8[1] = (uint8_t) shmem[ind + 1];
     Int32Bytes.u8[2] = (uint8_t) shmem[ind + 2];
     Int32Bytes.u8[3] = (uint8_t) shmem[ind + 3];
     data[10 + i * 5    ] = (double)(Int32Bytes.i32);


     Int32Bytes.u8[0] = (uint8_t) shmem[ind + 4];
     Int32Bytes.u8[1] = (uint8_t) shmem[ind + 5];
     Int32Bytes.u8[2] = (uint8_t) shmem[ind + 6];
     Int32Bytes.u8[3] = (uint8_t) shmem[ind + 7];
     data[10 + i * 5 + 1] = (double)(Int32Bytes.i32);
     Int32Bytes.u8[0] = (uint8_t) shmem[ind + 8];
     Int32Bytes.u8[1] = (uint8_t) shmem[ind + 9];
     Int32Bytes.u8[2] = (uint8_t) shmem[ind +10];
     Int32Bytes.u8[3] = (uint8_t) shmem[ind +11];
     data[10 + i * 5 + 2] = (double)(Int32Bytes.i32);
     data[10 + i * 5 + 3] =  (double) shmem[ind +12];
     data[10 + i * 5 + 4] =  (double) shmem[ind +13];
    }
   break;
   case 2:
    Uint64Bytes.u8[0] = (uint8_t) shmem[1];
    Uint64Bytes.u8[1] = (uint8_t) shmem[2];
    Uint64Bytes.u8[2] = (uint8_t) shmem[3];
    Uint64Bytes.u8[3] = (uint8_t) shmem[4];
    Uint64Bytes.u8[4] = (uint8_t) shmem[5];
    Uint64Bytes.u8[5] = (uint8_t) shmem[6];
    Uint64Bytes.u8[6] = (uint8_t) shmem[7];
    Uint64Bytes.u8[7] = (uint8_t) shmem[8];
    data[1] = (double)Uint64Bytes.u64;
    Uint64Bytes.u8[0] = (uint8_t) shmem[9];
    Uint64Bytes.u8[1] = (uint8_t) shmem[10];
    Uint64Bytes.u8[2] = (uint8_t) shmem[11];
    Uint64Bytes.u8[3] = (uint8_t) shmem[12];
    Uint64Bytes.u8[4] = (uint8_t) shmem[13];
    Uint64Bytes.u8[5] = (uint8_t) shmem[14];
    Uint64Bytes.u8[6] = (uint8_t) shmem[15];
    Uint64Bytes.u8[7] = (uint8_t) shmem[16];
    data[8] = (double)Uint64Bytes.u64;
    for (i = 0;i < 96;i++) {
     int ind = 41 + i * 8;
     Int16Bytes.u8[0] = (uint8_t) shmem[ind    ];
     Int16Bytes.u8[1] = (uint8_t) shmem[ind + 1];
     data[10 + i * 5    ] = (double)(Int16Bytes.i16);
     Int16Bytes.u8[0] = (uint8_t) shmem[ind + 2];
     Int16Bytes.u8[1] = (uint8_t) shmem[ind + 3];
     data[10 + i * 5 + 1] = (double)(Int16Bytes.i16);
     Int16Bytes.u8[0] = (uint8_t) shmem[ind + 4];
     Int16Bytes.u8[1] = (uint8_t) shmem[ind + 5];
     data[10 + i * 5 + 2] = (double)(Int16Bytes.i16);
     data[10 + i * 5 + 3] =  (double) shmem[ind + 6];
     data[10 + i * 5 + 4] =  (double) shmem[ind + 7];
    }
```

```
   break;
   case 3:
    Uint64Bytes.u8[0] = (uint8_t) shmem[1];
    Uint64Bytes.u8[1] = (uint8_t) shmem[2];
    Uint64Bytes.u8[2] = (uint8_t) shmem[3];
    Uint64Bytes.u8[3] = (uint8_t) shmem[4];
    Uint64Bytes.u8[4] = (uint8_t) shmem[5];
    Uint64Bytes.u8[5] = (uint8_t) shmem[6];
    Uint64Bytes.u8[6] = (uint8_t) shmem[7];
    Uint64Bytes.u8[7] = (uint8_t) shmem[8];
    data[1] = (double)Uint64Bytes.u64;
    Uint64Bytes.u8[0] = (uint8_t) shmem[9];
    Uint64Bytes.u8[1] = (uint8_t) shmem[10];
    Uint64Bytes.u8[2] = (uint8_t) shmem[11];
    Uint64Bytes.u8[3] = (uint8_t) shmem[12];
    Uint64Bytes.u8[4] = (uint8_t) shmem[13];
    Uint64Bytes.u8[5] = (uint8_t) shmem[14];
    Uint64Bytes.u8[6] = (uint8_t) shmem[15];
    Uint64Bytes.u8[7] = (uint8_t) shmem[16];
    data[8] = (double)Uint64Bytes.u64;
    for (i = 0;i < 96;i++) {
     int ind = 41 + i * 10;
     Uint32Bytes.u8[0] = (uint8_t) shmem[ind    ];
     Uint32Bytes.u8[1] = (uint8_t) shmem[ind + 1];
     Uint32Bytes.u8[2] = (uint8_t) shmem[ind + 2];
     Uint32Bytes.u8[3] = (uint8_t) shmem[ind + 3];
     data[10 + i * 5    ] = (double)(Uint32Bytes.u32);
     Uint16Bytes.u8[0] = (uint8_t) shmem[ind + 4];
     Uint16Bytes.u8[1] = (uint8_t) shmem[ind + 5];
     data[10 + i * 5 + 1] = (double)(Uint16Bytes.u16);
     Uint16Bytes.u8[0] = (uint8_t) shmem[ind + 6];
     Uint16Bytes.u8[1] = (uint8_t) shmem[ind + 7];
     data[10 + i * 5 + 2] = (double)(Uint16Bytes.u16);
     data[10 + i * 5 + 3] =  (double) shmem[ind + 8];
     data[10 + i * 5 + 4] =  (double) shmem[ind + 9];
    }
   break;
   default:
   mexPrintf("**************unknown data format*******¥n");
  }
  return;
 }
 mexPrintf("------------------------------¥n");
 mexPrintf("Usage: sharedmemv5             ¥n");
 mexPrintf("data=sharedmemv5;              ¥n");
 mexPrintf("Shared Memory Access for MATLAB by Gerox(c) 2024¥n");
 mexPrintf("http://www.gerox.com Build %s¥n",__DATE__);
 mexPrintf("------------------------------¥n");
 mexErrMsgTxt("one output must be required.");
}
```

MATLAB からのデータ取り込みスクリプト

IMU データと、Point Cloud データは、非同期で CallBack が呼び出される。呼び出される毎に Callback して得られたデータを取り込むには、処理速度が不足し、取り込み落ちが発生する。そのため、num 回分まとめて、Point Cloud データを取り込み終わってから、処理しデータを出力する oneshot 関数を実装した。Point Cloud データを取り込んでいる間、非同期で IMU データも取り込まれるため、出力は、２つの変数に分けて出力するようにしている。oneshot 関数内での一回のサンプルで Point Cloud データは、96*num 点のデータが取り込めるが、反射がない場合、(0,0,0)の座標を返すため、意味のない座標データは、削除し、データ数の削減と、処理速度の向上を図っている。IMU センサの座標系は、z 軸出力から、NED 座標系であると考えられるが、Livox のファームウェアがどのように実装されているかは、実際にデータを出力しチェックする必要がある。また、例えば、ICM-40609 のジャイロの出力は、deg/sec であるのに対し、Livox のファームウェアでは、rad/sec に変換されて出力されているため、符号や単位などは、変更されている可能性がある。

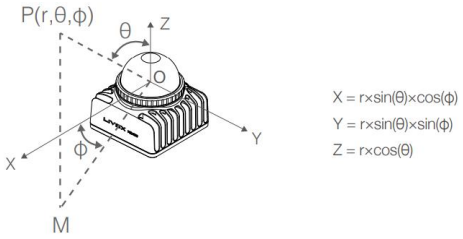
LiDAR 座標系と ROS の座標系ルール（ENU 座標系(x-east, y-north, z-up)と単位）に変換している。そこでここでは、処理速度の負荷を考慮し、共有メモリ内での単位変換、座標系変換は行わず、oneshot 関数内で実装した。Oneshot 関数内では、Data Type により自動判別するため、変更の必要はないが、異なるデータタイプのデータを取り込みたい場合には、main.cpp を書き換えて、リコンパイルし実行する必要がある。

	main.cpp 共有メモリ	MATLAB 行列 （oneshot 関数の出力）
Data Type 0	Gyro_x[rad/s] Gyro_y[rad/s] Gyro_z[rad/s] Acc_x[g] Acc_y[g] Acc_z[g]	Gyro_x[rad/s] Gyro_y[rad/s] Gyro_z[rad/s] -Acc_x[m/s^2] -Acc_y[m/s^2] -Acc_z[m/s^2]
Data Type 1	X[mm] Y[mm] Z[mm] Reflectivity Tag	X[m] Y[m] Z[m] Reflectivity Tag
Data Type 2	X[cm] Y[cm] Z[cm] Reflectivity Tag	X[m] Y[m] Z[m] Reflectivity Tag
Data Type 3	Depth[mm] Theta[0.01 deg] Phi[0.01 deg] Reflectivity Tag	X[m] Y[m] Z[m] Reflectivity Tag

oneshot.m

```
function [lidar,imu]=oneshot(num)
pupdate = 0;
imupdate = 0;
pclcnt = 0;
imu=[];
lidar=[];
loopcount=100000000;
for i=1:loopcount
    data=sharedmemv5;
    datatype = data(1,1);
    if(datatype == 0)
        if(data(4,2) ~= imupdate)
            imupdate = data(4,2);
            % order gyro acc
            imu=[imu,[data(3:5,1);data(1:3,2);imupdate]];
        end
        continue;
    else
        if(data(2,1) == pupdate)
            continue
        end
        pclcnt = pclcnt + 1;
        pupdate = data(2,1);
        d96=data(:,3:end);
        % minimum detection range 10cm, more than 10[cm]->100[mm]
        % maximum detection range 100m, less than
        % 100[m]->10000[cm]->100000[mm]
        if(datatype == 3)
            d96idx=(d96(1,:) > 100) & (d96(1,:) < 100000);
            lidar=[lidar,[d96(:,d96idx);ones(1,sum(d96idx))*pupdate]];
            if(pclcnt >= num)
                % unit from gyro[rad/s] and acc[g] to gyro[rad/s] and acc[m/s^2]
                imu(4:6,:)= - 9.8 * imu(4:6,:);
                r = lidar(1,:) / 1000;% unit [m]
                % from 0.01[deg] to [rad]
                theta=lidar(2,:) * 19 / 108862;% * pi / 180 /100 [rad]
                phi=lidar(3,:) * 19 / 108862;% * pi / 180 /100 [rad]
                lidar(1,:)=r.*sin(theta).*cos(phi);
                lidar(2,:)=r.*sin(theta).*sin(phi);
                lidar(3,:)=r.*cos(theta);
                break;
            end
        elseif(datatype == 2)
            d96idx=(max(abs(d96(1:3,:))) > 10) & (max(abs(d96(1:3,:))) < 10000);
            lidar=[lidar,[d96(:,d96idx);ones(1,sum(d96idx))*pupdate]];
            if(pclcnt >= num)
                % unit from gyro[rad/s] and acc[g] to gyro[rad/s] and acc[m/s^2]
                imu(4:6,:)= - 9.8 * imu(4:6,:);
                % from [cm] to [m]
                lidar(1:3,:) = lidar(1:3,:) / 100;% unit [m]
                break;
            end
        elseif(datatype == 1)
            d96idx=(max(abs(d96(1:3,:))) > 100) & (max(abs(d96(1:3,:))) < 100000);
            lidar=[lidar,[d96(:,d96idx);ones(1,sum(d96idx))*pupdate]];
            if(pclcnt >= num)
                % unit from gyro[rad/s] and acc[g] to gyro[rad/s] and acc[m/s^2]
                imu(4:6,:)= - 9.8 * imu(4:6,:);
                % from [mm] to [m]
                lidar(1:3,:) = lidar(1:3,:) / 1000;% unit [m]
                break;
            end
        end
    end
end
if(i == loopcount)
    fprintf('WARNING: time out. Please increase loopcount.');
end
end
```

測定範囲は、10[cm]-100[m]でその範囲外データは削除している。Data Type 1,2 は、単位変換だけであるが、極座標系で出力される Data type 3 は、



上記の式により IMU データ、Point Cloud データを計算している。

Developer Command Prompt で

¥Livox-SDK2-1.2.5¥build¥samples¥livox_lidar_quick_startv5¥Release の

>> livox_lidar_quick_startv5.exe

を実行しながら行う。

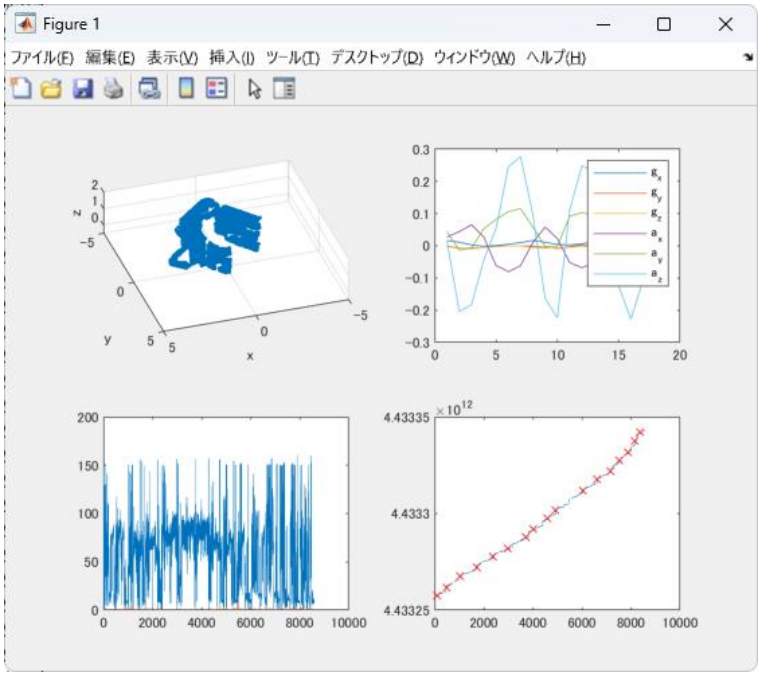
MATLAB 上で

>>untitled8.m

を動かせば下図が出てくる。

¥Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_startv5¥untitled8.m

```
clear all;close all
num=150;
figure
[onelidar,imu] = oneshot(num);
subplot(2,2,1);
p1=plot3(onelidar(1,:),onelidar(2,:),onelidar(3,:),'.');axis equal
xlabel('x');ylabel('y');zlabel('z');
axis([-5,5,-5,5,-0.5,2]);grid on;
view(162,32)
subplot(2,2,2);p2=plot((imu(1:6,:)+[0;0;0;0;0;9.8])');
legend('g_x','g_y','g_z','a_x','a_y','a_z');
for i=1:100
    tic;
    [onelidar,imu] = oneshot(num);
    toc
    set(p1,'XData',onelidar(1,:),'YData',onelidar(2,:),'ZData',onelidar(3,:));
    subplot(2,2,2);plot((imu(1:6,:)+[0;0;0;0;0;9.8])');
    legend('g_x','g_y','g_z','a_x','a_y','a_z');
    subplot(2,2,3);plot(onelidar(4:5,:)')
    imuind=[];
    for j=1:length(imu(7,:))
        [~,ind]=min(abs(onelidar(6,:)-imu(7,j)));
        imuind=[imuind;ind];
    end
    subplot(2,2,4);
    plot(1:length(onelidar(6,:)),onelidar(6,:),imuind,imu(7,:),'rx')
    drawnow;
end
```



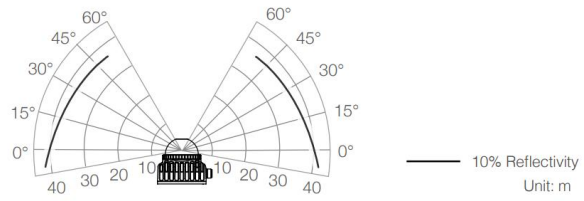
左上が Point Cloud データ、右上が、IMU データの横軸がサンプル回数、縦軸が、3 軸ジャイロ、加速度を表している。なお、LiDAR データは、ENU 座標系として扱うことができる。ENU 座標系の加速度の z 軸成分は、重力加速度-9.8[m/s^2]がオフセットとして入ってしまうため、データ表示では、z 軸成分に 9.8 を加算して 0 付近で見れるようにしている。左下は、横軸が Point Cloud のサンプル点数と縦軸で青が反射強度、赤がタグデータを表している。右下の右肩上がりの直線のグラフは、横軸がサンプル点数、縦軸が経過時間（単位 ns）を表している。青線が Point Cloud が取り込まれた時間で、赤✕が IMU データが取り込まれた時間を表している。

pcshow 関数（要 Computer Vision Toolbox）

Lidar Toolbox でも使用している pcshow 関数で Point Cloud データの反射強度を活用し、点群に色を追加してみた。ちなみに、MOVIE=1 とすると、動画 untitled8pcl.avi として保存される。

PointCloud の反射強度について

> **Target reflectivity**: expressed by a number from 0 to 255. 0 to 150 corresponds to the reflectivity within the range of 0 to 100% in the Lambertian reflection model. 151 to 255 corresponds to the reflectivity of target objects with retroflection properties. When the target is less than 2 m from the Mid-360, it may result in a large reflectivity error. The data should only be used to distinguish whether the target is total reflective or diffuse reflective.



Effective Detection Range inside FOV of the Mid-360

Tag 情報の活用

Tag は検出点の他の追加情報を示す。点群は 8 ビットの符号なし整数としてマークされており、ビットに応じて複数の領域に分けられている。各領域は、雨、霧、ほこり、類似物間の接着点群など、検出点の属性を表している。その中で、Confidence レベルは検出点の信頼性を示しており、通常、正常な点は 0（高い信頼性）である。低い信頼性は、検出点が対応する属性に大きく影響を受け、検出結果の信頼性が低いことを示している。必要に応じて、この情報に基づいて点群をフィルタリングすることができる。

Tags: Indicates additional information about the detected points. Point cloud tags are 8-bit unsigned integers, which are divided into several groups. Each group indicates one property of the detected point, including rain, fog, dust, and dragging noise between objects, etc. The confidence level indicates the credibility of the detected points. "0" stands for normal point cloud data (high confidence level); low confidence level indicates that the detected point is highly affected by the corresponding type of noise, and as a result the detection result has low credibility. Filter the point cloud based on the tag information.

The format of the tag is as shown below:

Bit[7-6]	Bit[5-4]	Bit[3-2]	Bit[1-0]
Reserved	Detected points property: Other 0: High confidence level (normal points) 1: Moderate confidence level 2: Low confidence level 3: Reserved	Detected points property: Atmospheric particles like rain, fog, and dust. 0: High confidence level (normal points) 1: Moderate confidence level 2: Low confidence level 3: Reserved	Detected points property: Dragging noise between objects: 0: High confidence level (normal points) 1: Moderate confidence level 2: Low confidence level 3: Reserved

Bit	Description
bit 6~7	Reserved
bit 4~5	Properties of Detection Points: other 0: High confidence (normal point) 1: Medium confidence 2: Low confidence 3: Reserved
bit 2~3	Properties of Detection Points: Rain, fog, dust and other tiny particles 0: High confidence (normal point) 1: Medium confidence 2: Low confidence 3: Reserved
bit 0~1	Properties of Detection Points: Glue point cloud between adjacent objects 0: High confidence (normal point) 1: Medium confidence 2: Low confidence 3: Reserved

例えば、スクリプトで、Confidence レベルの低い点を除去するには、

```
ind=logical(ones(size(othrMid))-(othrLow+rainLow+glueLow));
```

また、雨と、ほこりだけを除去するには、

```
ind=logical(ones(size(othrMid))-(rainLow+glueLow)-(rainMid+glueMid));
```

と変更すれば対応できる。

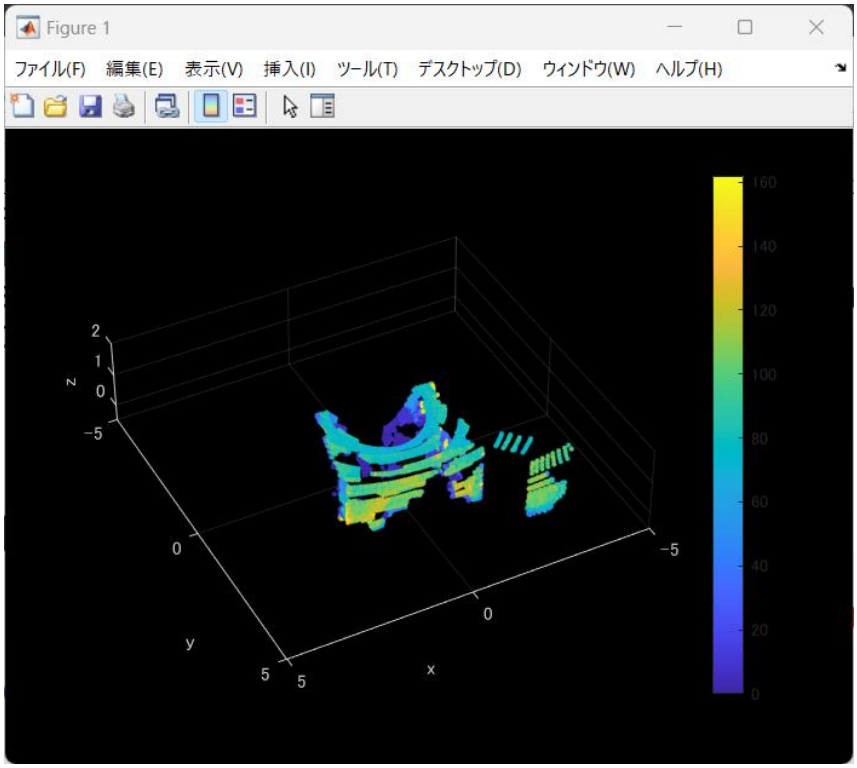
Untitled8pc1 を MATLAB 上で実行すれば下の図が出てくる。Untitled8pc1 の中身を書き換えることでいろいろ除去可能。

¥Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_startv5¥untitled8pcl.m

```
clear all;close all
num=150;
[onelidar,imu] = oneshot(num);
p1=pcshow(onelidar(1:3,:)',uint8(onelidar(4,:))','MarkerSize',50);
xlabel('x');ylabel('y');zlabel('z');
axis([-5,5,-5,5,-0.5,2]);
view(162,83)
view(-37.5,30.0)
colorbar;
loops = 10000;
MOVIE=0;
if MOVIE
    M(loops) = struct('cdata',[],'colormap',[]);
end
for i=1:loops
    tic;
    [onelidar,imu] = oneshot(num);
    toc
    othrMid=bitand(onelidar(5,:),2^4)==2^4;
    othrLow=bitand(onelidar(5,:),2^5)==2^5;
    rainMid=bitand(onelidar(5,:),2^2)==2^2;
    rainLow=bitand(onelidar(5,:),2^3)==2^3;
    glueMid=bitand(onelidar(5,:),2^0)==2^0;
    glueLow=bitand(onelidar(5,:),2^1)==2^1;
    ind=logical(ones(size(othrMid)));
    %  ind=logical(othrMid+othrLow);
    % ind=logical(rainMid+rainLow);
    % ind=logical(rainMid);
    %  ind=logical(glueMid+glueLow);
```

```
    %  ind=logical(othrMid+rainMid+glueMid);
    %  ind=logical(othrLow+rainLow+glueLow);
    %  ind=logical(othrMid+rainMid+glueMid+othrLow+rainLow+glueLow);
    %  ind=logical(ones(size(othrMid))-(rainLow+rainMid));
    %  ind=logical(ones(size(othrMid))-(othrLow+rainLow+glueLow));
    % ind=logical(ones(size(othrMid))-(othrLow+rainLow+glueLow)-(othrMid+rainMid+glueMid));
    onelidar=onelidar(:,ind);
    p1.Children.XData=onelidar(1,:);
    p1.Children.YData=onelidar(2,:);
    p1.Children.ZData=onelidar(3,:);
    p1.Children.CData=uint8(onelidar(4,:))';
    drawnow;
    if MOVIE
        M(i)=getframe(gcf);
    end
end

if MOVIE
    v = VideoWriter("untitled8pcl.avi","Indexed AVI");
    v.VideoCompressionMethod
    open(v)
    writeVideo(v,M)
    close(v)
end
```

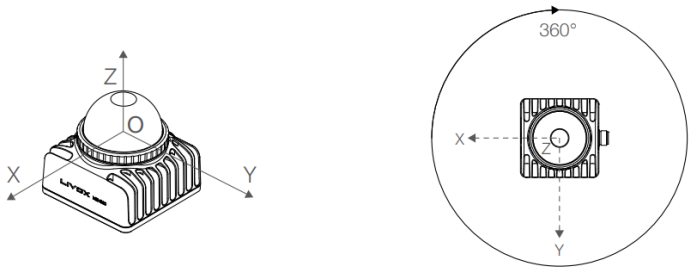


IMU センサの活用

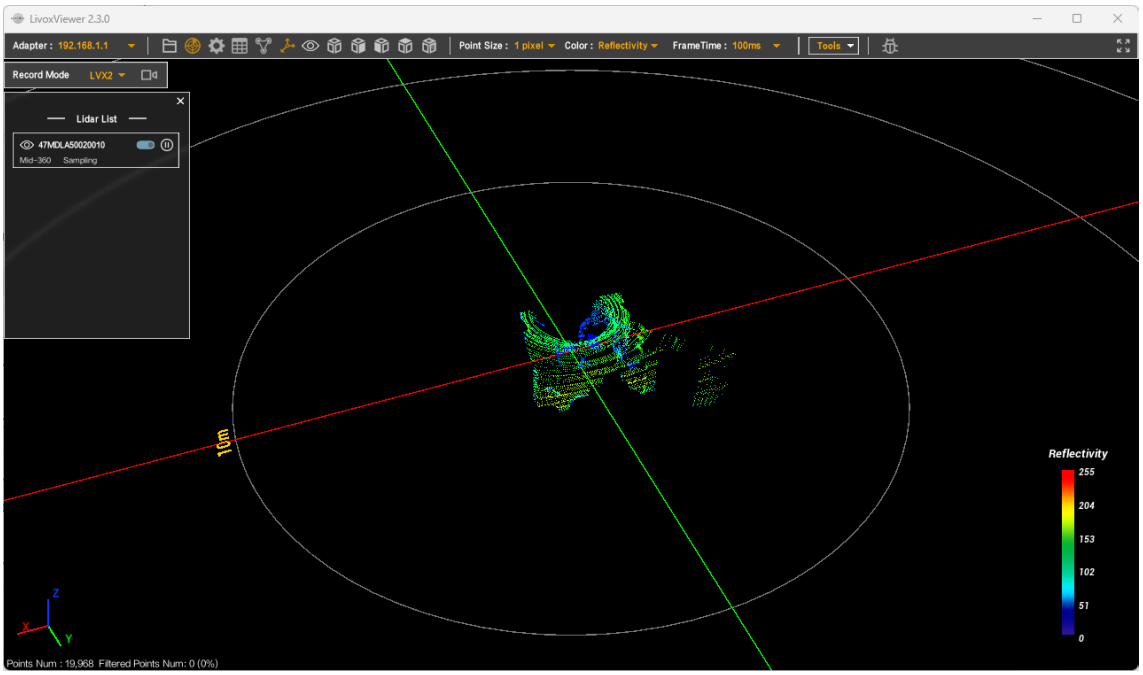
Mid-360 の Point Cloud データの z 軸は、上向きであるが、Mid-360 の 3 軸加速度の生出力は、＋1[g]前後で出力される。（つまり、重力加速度を表す z 軸の正方向は、下向き）となっている。そのため、加速度センサとジャイロセンサの回転による符号には注意を要する。なお、oneshot 関数は、ENU 座標系への変換実装したものを使ってテストした。

Mid-360 の LiDAR 座標系と IMU 座標系の確認

Livox View2 を使って **MATLAB** のデータと比較・確認する。



Mid-360 Coordinates



Livox View2 でキャプチャした点群データを表示したものである。比較のために座標軸を合わせて表示している。今度は、**IMU** センサも実際に動かして符号と軸を確認する。

imubasictest.m

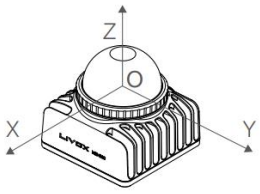
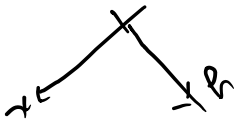
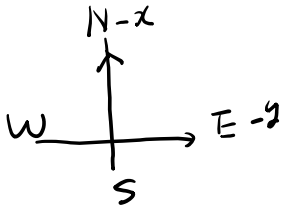
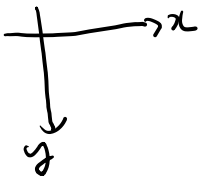
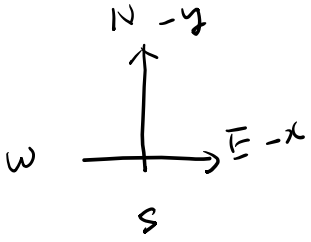
```
clear all;close all
num=1500;
while(1)
  [onelidar,imu] = oneshot(num);
  subplot(6,1,1);plot(imu(1,:));title('g_x');ylim([-2,2]);
  subplot(6,1,2);plot(imu(2,:));title('g_y');ylim([-2,2]);
  subplot(6,1,3);plot(imu(3,:));title('g_z');ylim([-2,2]);
  subplot(6,1,4);plot(imu(4,:));title('a_x');ylim([-10.5,10.5]);
  subplot(6,1,5);plot(imu(5,:));title('a_y');ylim([-10.5,10.5]);
  subplot(6,1,6);plot(imu(6,:));title('a_z');ylim([-10.5,10.5]);
  if(max(abs(imu(1,:)))>2)break;end
  if(max(abs(imu(2,:)))>2)break;end
  if(max(abs(imu(3,:)))>2)break;end
  drawnow;
end
```

NED 座標系と、**ENU** 座標系の違いは、

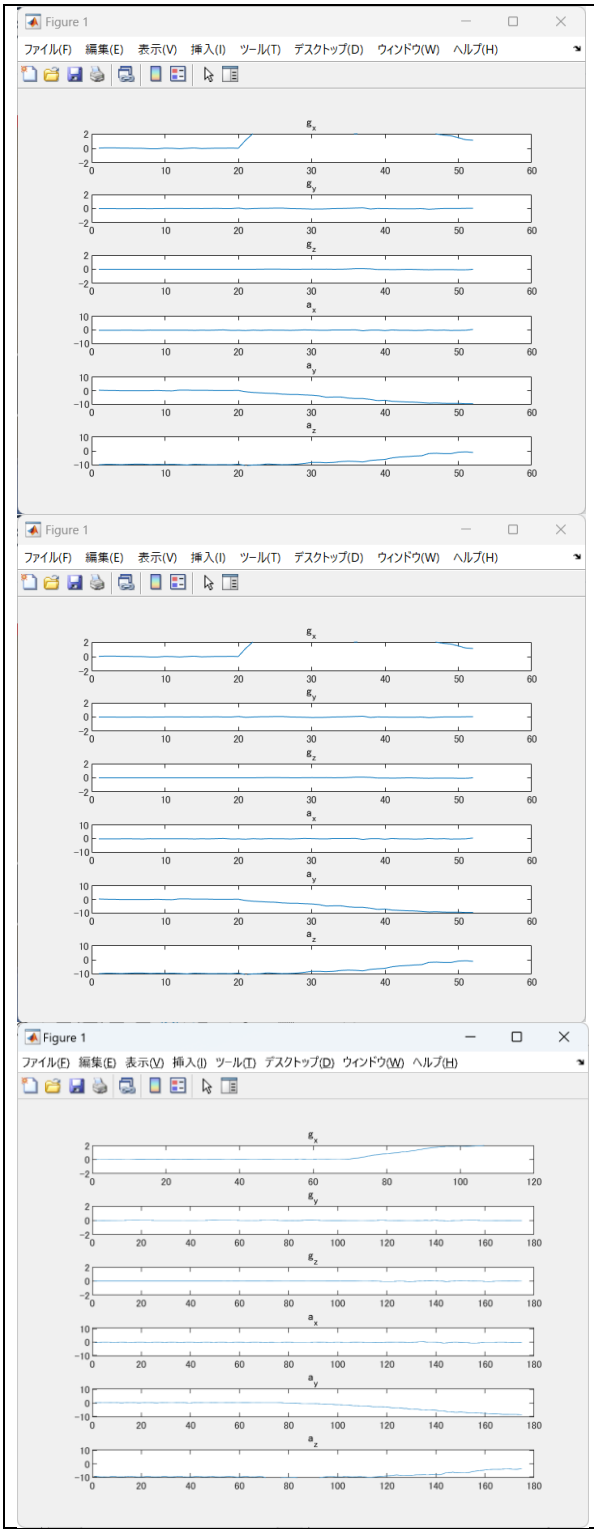
・Z 軸の向きの違い
・基準となる進行方向の違い
がある。

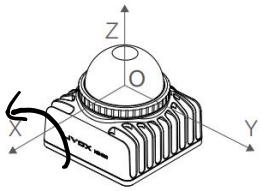
以下にそのイメージを示す。この場合、Z 軸の向きは、Lidar 座標系と ENU 座標系で同じであるが、進行方向（ここでは、Mid-360 をどの向きに設置するかにもよるが・・・）

ここでは、Lidar 座標系の X 軸方向が前方とした場合、ENU 座標系の y 軸は、左方向が正となるが、NED 座標系の y 軸は、右方向が正となる。つまり、どの座標系を取るかにより、3 軸ジャイロ、3 軸加速度の出力の符号が変わる。

LiDAR 座標系	NED 座標系	ENU 座標系
 Mid-360 Co 	 	 

x 軸周りに右ねじ正回転

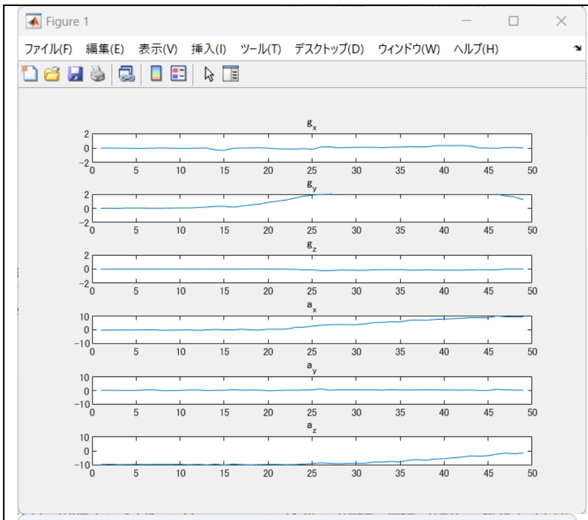


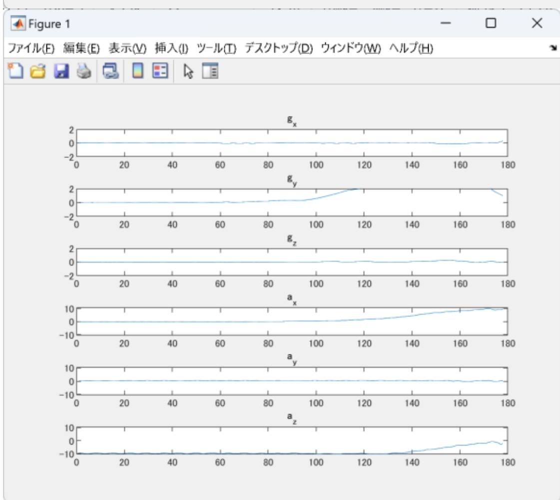


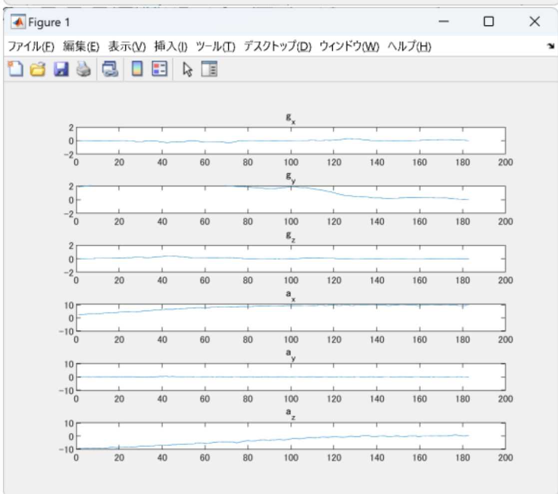
gx-> + OK
gy-> 0 OK
gz-> 0 OK

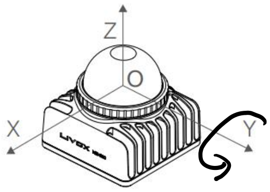
ax-> 0 OK
ay-> 0 to -9.8 OK
az-> -9.8 to 0 OK

y 軸周りに右ねじ正回転





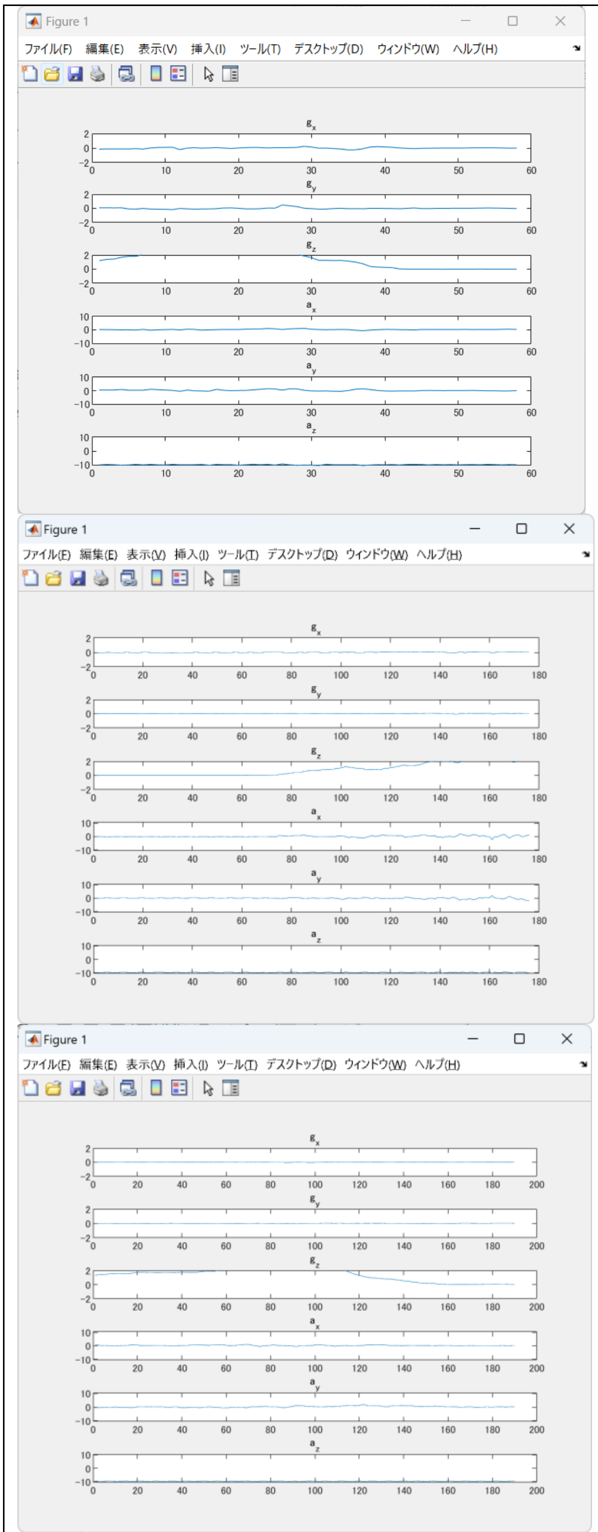
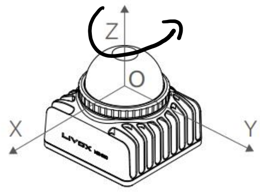





```
gx-> 0 OK
gy-> + OK
gz-> 0 OK

ax-> 0 to 9.8 OK
ay-> 0         OK
az-> -9.8 to 0 OK
```

z 軸（上向き）周りに右ねじ正回転

	 gx-> 0 OK gy-> 0 OK gz-> + OK ax-> 0 OK ay-> 0 OK az-> -9.8 OK

IMU の出力は、ENU 座標系として、LiDAR 座標系と一致している。

Mid-360 で使用している IMU センサは、ICM40609 であり、マニュアルによると、

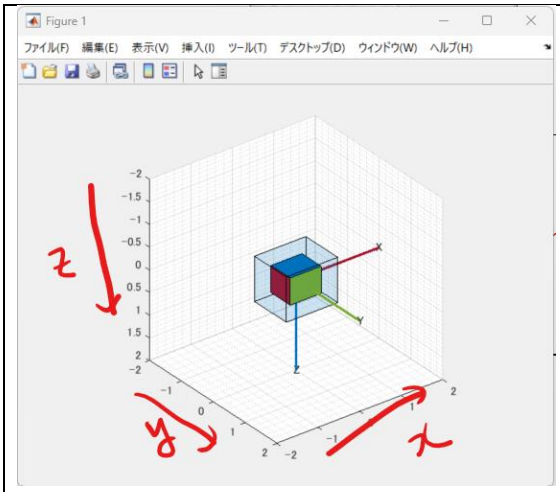
- Gyro Noise: **4.5mdps/√Hz**
- Gyro Offset Stability TC: **±10mdps/C**
- Gyro Sensitivity Error: **±0.5%**
- Gyro Sensitivity/temp: **±0.045%/C**
- Accel Noise: **100μg/√Hz**
- Accel Offset Stability TC: **±0.15mg/C**
- Accel Sensitivity Error: **±0.5%**
- Accel Sensitivity/temp: **±0.007%/C**
- Gyro + Accel Combo current: **0.77mA**
- Extended Accel Full Scale Range: **32g**
- Improved ODR Latency: **32KHz**

と書いてある。ここからカルマンフィルタを使うためのパラメータ計算を行う。

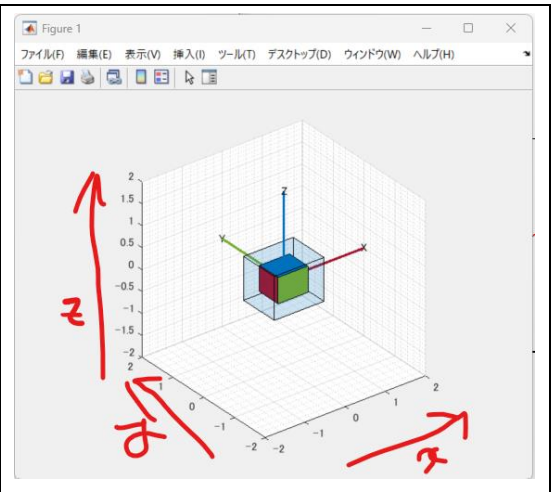
Gyro noise	1 [mdeg/s] = 0.001×pi/180 [rad/s] なので 4.5[mdps/sqrt(Hz)]=4.5×0.001×pi/180 [rad/s/sqrt(Hz)]	0.0045*pi/180[rad/s/sqrt(Hz)]
Accel noise	1 [μg] = 10^-6[m/s^2] なので 100 [μg/sqrt(Hz)] = 100×10^-6[m/s^2/sqrt(Hz)]	0.0001 [m/s^2/sqrt(Hz)]

poseplot 関数による IMU の可視化(要 Navigation Toolbox)

ここでは、IMU センサの座標系を確認するために、Navigation Toolbox の poseplot 関数を使って確認してみる。ちなみに、poseplot 関数のデフォルトの座標系での表示は、North East Down(NED)系である。それに対し、実装した oneshot 関数の出力は、East North Up（ENU）系であるので注意する。ちなみに、poseplot 関数は、オプションにより、NED 座標系と、ENU 座標系に対応している。図を見るとわかるように、z 軸はもちろんであるが、y 軸の向きも変わり表示される。



NED(デフォルト)



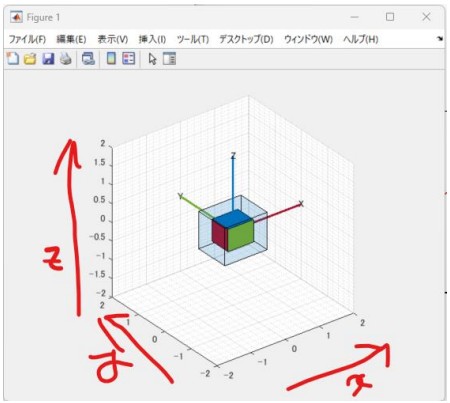
ENU

姿勢角度を計算する imufilter 関数は、デフォルトでは、NED 座標系で計算しているので注意を要する。（Quaternion と言っても、座標系取り方によって値が変わるので注意）で行う。最初、加速度、ジャイロセンサは、静止状態で計測し、その値を使い、ジャイロは、オフセットキャンセル、加速度は、重力加速度 9.8[m/s]でノーマライズした。

¥Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_startv5¥untitled9a.m

```
clear all;close all
num=150;
GyroscopeNoiseMPU9250 = 3.0462e-06; % GyroscopeNoise (variance) in units of rad/s
AccelerometerNoiseMPU9250 = 0.0061; % AccelerometerNoise (variance) in units of m/s^2
GyroscopeNoiseICM40609 = 0.0045 * pi / 180; % GyroscopeNoise (variance) in units of rad/s
AccelerometerNoiseICM40609 = 0.0001; % AccelerometerNoise (variance) in units of m/s^2

FUSE = imufilter('SampleRate',200,'ReferenceFrame', 'ENU');
FUSE.GyroscopeNoise=GyroscopeNoiseICM40609*10000;
FUSE.AccelerometerNoise=AccelerometerNoiseICM40609;
FUSE.GyroscopeDriftNoise = 3.0462e-13;      %(rad/s)^2
FUSE.LinearAccelerationNoise = 0.0096236;       %(m/s^2)^2
[~,imu] = oneshot(num);
imu0=mean(imu(1:3,:),2);
acc0=mean(9.8./sqrt(sum(imu(4:6,:).^2)))
% requires R2021b の Sensor Fusion and Tracking Toolbox
pp = poseplot('ENU');
xlabel('x');ylabel('y');zlabel('z');
for i=1:10000
    tic;
    [~,imu] = oneshot(num);
    toc
    gyro=1.25*(imu(1:3,:)-imu0)';
    accel=(imu(4:6,:)*acc0)';
    rotators = FUSE(accel,gyro);
    for ii=1:size(imu(1,:))
      set(pp,"Orientation",rotators(ii))
    drawnow limitrate
end
end
```



コラム　ROS の座標系と NED、ENU 座標系

3 次元座標を扱う場合、センサの種類や向き、配置により、座標系が異なる場合があるので、細心の注意が必要である。ちなみに、ROS では、

https://www.ros.org/reps/rep-0103.html

に座標系を定義している。

Coordinate Frame Conventions
All coordinate frames should follow these conventions.
Chirality
All systems are right handed. This means they comply with the right hand rule [4].

Axis Orientation
In relation to a body the standard is:
- x forward
- y left
- z up

For short-range Cartesian representations of geographic locations, use the east north up [5] (ENU) convention:
- X east
- Y north
- Z up

To avoid precision problems with large float32 values, it is recommended to choose a nearby origin such as your system's starting position.
Suffix Frames
In the case of cameras, there is often a second frame defined with a "_optical" suffix. This uses a slightly different convention:
- z forward
- x right
- y down

For outdoor systems where it is desirable to work under the north east down [6] (NED) convention, define an appropriately transformed secondary frame with the "_ned" suffix:
- X north
- Y east
- Z down

Rotation Representation
There are many ways to represent rotations. The preferred order is listed below, along with rationale.
1. **quaternion**
   Compact representation
   No singularities
2. **rotation matrix**
   No singularities
3. fixed axis roll, pitch, yaw about X, Y, Z axes respectively
   No ambiguity on order
   Used for angular velocities
4. euler angles yaw, pitch, and roll about Z, Y, X axes respectively
   Euler angles are generally discouraged due to having 24 'valid' conventions with different domains using different conventions by default.

By the right hand rule, the yaw component of orientation increases as the child frame rotates counter-clockwise, and for geographic poses, yaw is zero when pointing east.

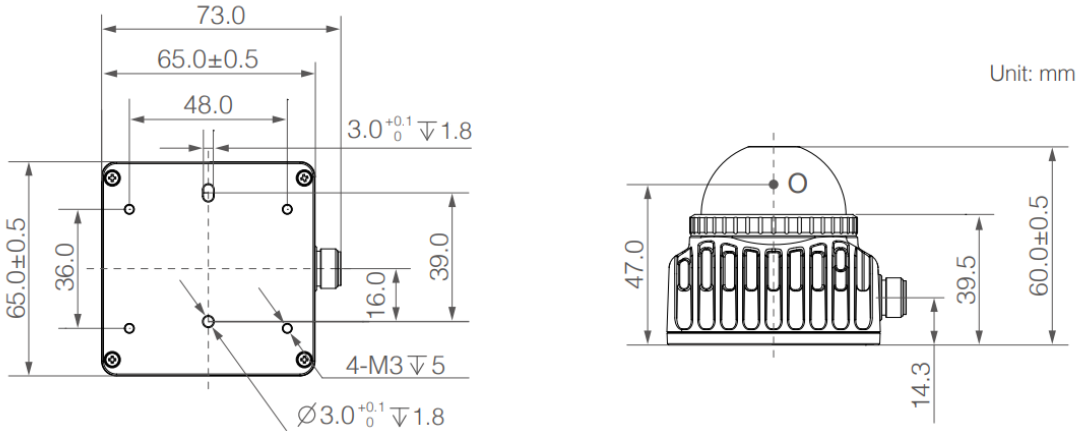
まとめると、

ROS で使う座標系	カメラで使う座標系 frame defined with a "_optical" suffix	屋外環境で使う座標系 frame with the "_ned" suffix
x forward y left z up	z forward x right y down	x forward y left z down
the east north up [5] (ENU) X east Y north Z up		the north east down [6] (NED) X north Y east Z down

絶対方位が入った場合、ENU と NED では、x 軸、y 軸の入れ替えや、軸の符号が変わる可能性があるので、注意を要する。今回の例では、地磁気センサはないので、その必要はない。

Mid-360 と IMU のセンサフュージョン

ドキュメントによれば、Lidar と IMU センサの原点が異なっているので、注意する。



The position of the IMU chip in the point cloud coordinates is x=11.0 mm, y=23.29 mm, z=-44.12 mm.

この例では、IMU の姿勢情報を活用し、Point Cloud データの座標変換を行い、MID-360 の姿勢によらず、初期姿勢のままで Point Cloud データを表示するようにしている。

¥Livox-SDK2-1.2.5¥samples¥livox_lidar_quick_startv5¥untitled9b.m

```
clear all;close all
num=150;
GyroscopeNoiseICM40609 = 0.0045*pi/180; % GyroscopeNoise (variance) in units of rad/s
AccelerometerNoiseICM40609 = 0.0001; % AccelerometerNoise (variance) in units of m/s^2

FUSE = imufilter('SampleRate',200,'ReferenceFrame', 'ENU');
FUSE.GyroscopeNoise=GyroscopeNoiseICM40609;
FUSE.AccelerometerNoise=AccelerometerNoiseICM40609;
FUSE.GyroscopeDriftNoise = 3.0462e-13;      %(rad/s)^2
FUSE.LinearAccelerationNoise = 0.0096236;       %(m/s^2)^2

[onelidar0,imu] = oneshot(num);
imu0=median(imu(1:3,:),2);
acc0=median(9.8./sqrt(sum(imu(4:6,:).^2)));
% requires R2021b の Sensor Fusion and Tracking Toolbox

figure;% default is NED frame
subplot(2,4,1);pp = poseplot('ENU');
xlabel('x');ylabel('y');zlabel('z');
subplot(2,4,2);pp1 = poseplot('ENU');
xlabel('x');ylabel('y');zlabel('z');
view([0,90]);
subplot(2,4,3);pp2 = poseplot('ENU');
xlabel('x');ylabel('y');zlabel('z');
view([90,0]);
subplot(2,4,4);pp3 = poseplot('ENU');
xlabel('x');ylabel('y');zlabel('z');
view([0,0]);

subplot(2,4,5);p11=plot3(onelidar0(1,:),onelidar0(2,:),onelidar0(3,:),'.');
axis equal;axis([-5,5,-5,5,-5,5]);grid on;
xlabel('x');ylabel('y');zlabel('z');
subplot(2,4,6);p12=plot3(onelidar0(1,:),onelidar0(2,:),onelidar0(3,:),'.');
axis equal;axis([-5,5,-5,5,-5,5]);grid on;
xlabel('x');ylabel('y');zlabel('z');view([0,90]);
subplot(2,4,7);p13=plot3(onelidar0(1,:),onelidar0(2,:),onelidar0(3,:),'.');
axis equal;axis([-5,5,-5,5,-5,5]);grid on;
xlabel('x');ylabel('y');zlabel('z');view([90,0]);
subplot(2,4,8);p14=plot3(onelidar0(1,:),onelidar0(2,:),onelidar0(3,:),'.');
axis equal;axis([-5,5,-5,5,-5,5]);grid on;
```

```
xlabel('x');ylabel('y');zlabel('z');view([0,0]);
for i=1:10000
    tic;
    [onelidar,imu] = oneshot(num);
    toc
    gyro=(imu(1:3,:)-imu0)';
    accel=(imu(4:6,:)*acc0)';
    rotators = FUSE(accel,gyro);
    for ii=1:size(imu(1,:))
        set(pp,"Orientation",rotators(ii))
        set(pp1,"Orientation",rotators(ii))
        set(pp2,"Orientation",rotators(ii))
        set(pp3,"Orientation",rotators(ii))
        drawnow limitrate
    end
    rotators1 = FUSE(mean(accel),mean(gyro));
    R=quat2rotm(rotators1);
if 0
    eul=quat2eul(rotators1,'ZYX');
    R=eul2rotm(eul,'ZYX');
end
if 0
    axang=rotm2axang(quat2rotm(rotators1));
    R=axang2rotm(axang);
end
    % change from NED to ENU coordinate for rotation axis
    onelidar0=R*[onelidar(1:3,:)-[11;23.39;-44.12]/1000];
    set(p11,'XData',onelidar0(1,:),'YData',onelidar0(2,:),'ZData',onelidar0(3,:));
    set(p12,'XData',onelidar0(1,:),'YData',onelidar0(2,:),'ZData',onelidar0(3,:));
    set(p13,'XData',onelidar0(1,:),'YData',onelidar0(2,:),'ZData',onelidar0(3,:));
    set(p14,'XData',onelidar0(1,:),'YData',onelidar0(2,:),'ZData',onelidar0(3,:));
end
```

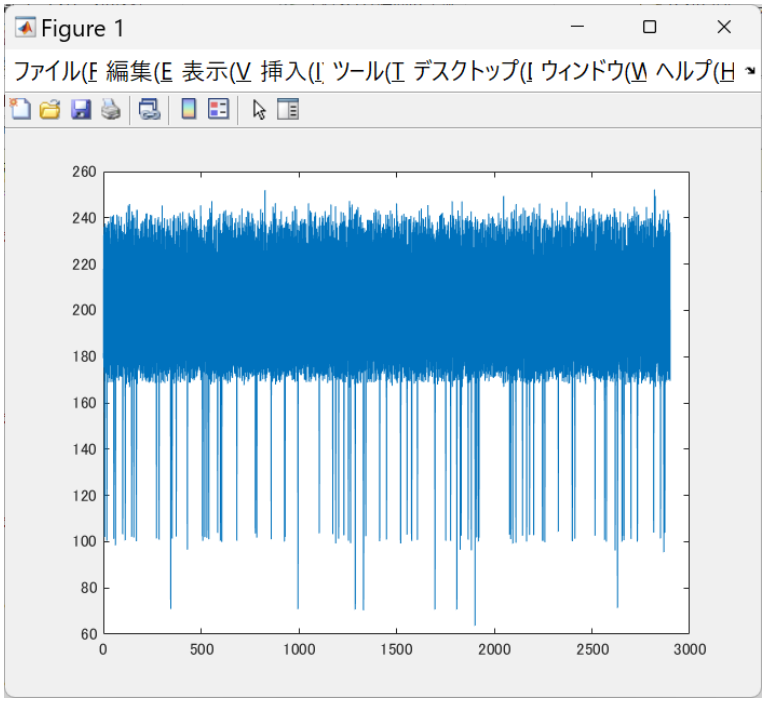
Mid-360 のセンサ回転により生じる振動

```
>> tic;[lidar,imu]=oneshot(25000);toc,save imu imu lidar
経過時間は 28.819904 秒です。
```

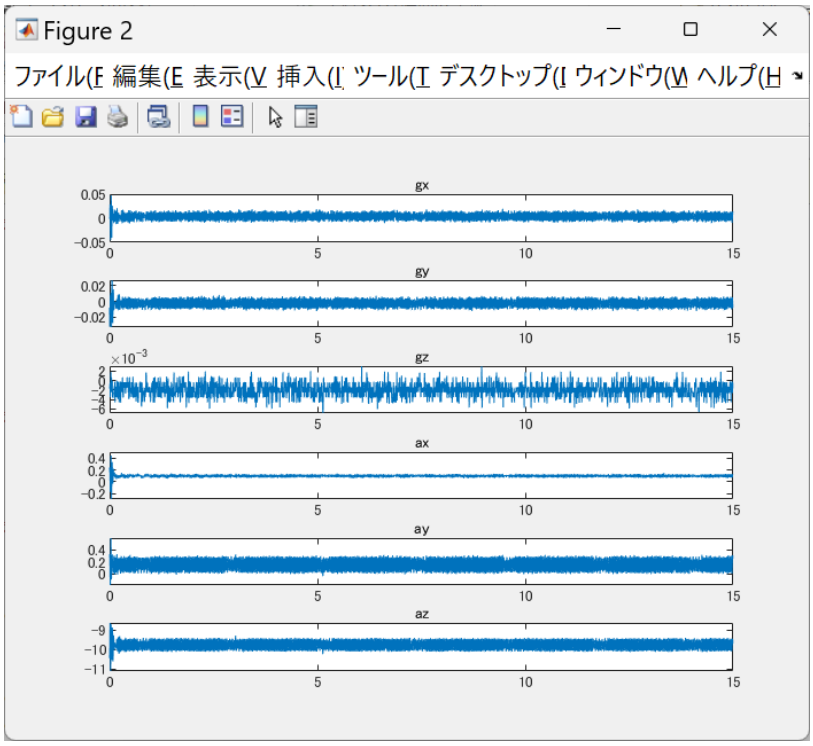
でサンプルしたデータを示す。

IMU のサンプル周波数は、

```
plot(1./diff(imu(7,:)/10^9))
```



であり、200Hz でサンプルできている。



周波数解析をするために、取り込み時間データを基準に IMU データを 200Hz でリサンプリングし、それを FFT 処理し周波数特性を見てみる。

imubasictest3.m

```
%tic;[lidar,imu]=oneshot(25000);toc,save imu imu lidar
%tic;[lidar,imu]=oneshot(25000);toc,save imux imu lidar
%tic;[lidar,imu]=oneshot(25000);toc,save imuy imu lidar
```

```
%tic;[lidar,imu]=oneshot(25000);toc,save imuz imu lidar
close all
load imu
plot(1./diff(imu(7,:)/10^9))
t= (imu(7,:)-imu(7,1))/10^9;
tt=t(1):1/200:t(end);
gx=interp1(t,imu(1,:),tt,"spline");
gy=interp1(t,imu(2,:),tt,"spline");
gz=interp1(t,imu(3,:),tt,"spline");
ax=interp1(t,imu(4,:),tt,"spline");
ay=interp1(t,imu(5,:),tt,"spline");
az=interp1(t,imu(6,:),tt,"spline");
figure
subplot(6,1,1);plot(tt,gx);title('gx');
subplot(6,1,2);plot(tt,gy);title('gy');
subplot(6,1,3);plot(tt,gz);title('gz');
subplot(6,1,4);plot(tt,ax);title('ax');
subplot(6,1,5);plot(tt,ay);title('ay');
subplot(6,1,6);plot(tt,az);title('az');
figure
fftgx=fft(gx);
fftgy=fft(gy);
fftgz=fft(gz);
fftax=fft(ax);
fftay=fft(ay);
fftaz=fft(az);
f=1./tt;
f=tt/tt(end)*200;
num=length(gx)/2;
subplot(6,1,1);plot(f(2:end/2),abs(fftgx(2:end/2))/num,'linewidth',3);title('gx');ylim([0,0.01]);
subplot(6,1,2);plot(f(2:end/2),abs(fftgy(2:end/2))/num,'linewidth',3);title('gy');ylim([0,0.01])
subplot(6,1,3);plot(f(2:end/2),abs(fftgz(2:end/2))/num,'linewidth',3);title('gz');ylim([0,0.01])
subplot(6,1,4);plot(f(2:end/2),abs(fftax(2:end/2))/num,'linewidth',3);title('ax');ylim([0,0.4])
subplot(6,1,5);plot(f(2:end/2),abs(fftay(2:end/2))/num,'linewidth',3);title('ay');ylim([0,0.4])
subplot(6,1,6);plot(f(2:end/2),abs(fftaz(2:end/2))/num,'linewidth',3);title('az');ylim([0,0.4])
pause
for i=1:6
subplot(6,1,i);xlim([27,30])
end
figure
plot3(lidar(1,:),lidar(2,:),lidar(3,:),'.');
```

約 28Hz の gx,gy,ay,az に回転による振動の影響の周波数が見られる。