Tac3D-SDK 二次开发手册

（Tac3D-AD2）

北京橡木果机器人科技有限公司

Acorn Roboticis

手册版本：v1.3.2

修改时间：2024 年 12 月 26 日

联系方式：support@acornrobotics.com

目录

1 Tac3D 软件基本框架

Tac3D 触觉传感器的软件包含 3 个部分：Tac3D 桌面应用程序（Tac3D-Desktop）、Tac3D 主程序（Tac3D-Core）和 Tac3D 客户端接口（Tac3D-API）。

其中，Tac3D 主程序负责连接 Tac3D 触觉传感器、解算触觉信息和发送触觉信息；Tac3D 桌面应用程序和 Tac3D 客户端接口可接收由 Tac3D 主程序发送的触觉信息，并用于展示、记录或二次开发。Tac3D 主程序与桌面应用程序和客户端接口之间采用 UDP 网络协议传输触觉信息。

因此，在使用 Tac3D 触觉传感器进行二次开发时，Tac3D-API 不能单独使用。而应当先启动 Tac3D 主程序，并将 Tac3D 主程序的网络传输地址和端口设置为调用 Tac3D 客户端接口的应用程序的网络地址和所设置的网络端口。这里，我们将 Tac3D 主程序和客户端接口分离设计的初衷是用户在编写客户端应用程序时可以把 Tac3D 主程序始终运行在后台，测试时只需重启客户端程序而无需重启 Tac3D 主程序。以避免每次启动主程序时的预热等待。

2 Tac3D-SDK 目录结构

解压 Tac3D-SDK-v3.2.1.zip 后得到的 Tac3D-SDK-v3.2.1 文件夹中包含以下子文件夹：

1. **doc 文件夹**

此文件夹中为 Tac3D-SDK 的说明文档

2. **Tac3D-Core 文件夹**

此文件夹中为 Tac3D 主程序。其中“Tac3D-Core/windows-x86_64”文件夹内为 windows 版本的 Tac3D 主程序及其依赖的动态链接库，支持 windows10/11 系统；“Tac3D-Core/linux-x86_64”文件夹内为 linux 版本的 Tac3D 主程序，支持 ubuntu18/20/22 系统。

3. **Tac3D-API 文件夹**

此文件夹中为 Tac3D 客户端接口。其中“Tac3D-API/python”文件夹内为 Tac3D-API 的 python 版本；“Tac3D-API/CPP”文件夹内为 Tac3D-API 的 C++版本。

4. **tools 文件夹**

此文件夹中为 Tac3D 传感器的相机测试工具。由于目前的 Tac3D-Desktop 仅有 windows 版本，在有图形界面的 Linux 系统上可暂时借助此文件夹中的 test_camera.py 脚本实现 Tac3D-Desktop 的相机编号测试功能。

3 启动 Tac3D 主程序

由于 Tac3D 客户端接口只进行触觉信息的接收而不进行解算。因此，在使用 Tac3D 客户端接口时，首先需启动 Tac3D 主程序。

3.1 在 DexHand Pro 上启动

如果您使用的是集成了 Tac3D 传感器的 DexHand Pro 机械手，Tac3D 的主程序将运行在 DexHand Pro 机械手内集成的处理器上。在调用启动服务端函数 dh_start_server 启动 DexHand Pro 机械手时(详见 DexHand Pro 机械手产品手册)，会同步启动 Tac3D 传感器，在启动后请尽量不要将 Tac3D 从机械手上拆除；倘若在启动传感器后因特殊原因从机械手上拆除了传感器，则需要首先停止服务端（dh_stop_server），并再次启动服务端（dh_start_server）才能重新启动 Tac3D 传感器。

3.2 从 Tac3D 桌面应用程序启动

如果您使用的是通过 Type-C 接口直接连接电脑的 Tac3D 触觉传感器，则需要将 Tac3D 的主程序将运行在连接 Tac3D 触觉传感器的电脑上。Tac3D-Desktop 中内置了 Tac3D 主程序，因此可以使用 Tac3D-Desktop 向 Tac3D-API 传输触觉数据。若要使用 Tac3D-Desktop 为 Tac3D-API 提供触觉数据，需要在启动触觉传感器时进行以下设置：

在启动传感器选项页面，将“启用网络传输”设置为“True”；将“接收地址”设置为运行基于 Tac3D-API 的客户端程序的计算机网络地址，如果是本机，则设置为“127.0.0.1”；将“接收端口”设置为客户端程序所设置的数据接收端口号，此端口号的默认值为 9988。

点击“启动传感器”，完成 Tac3D 主程序的启动。

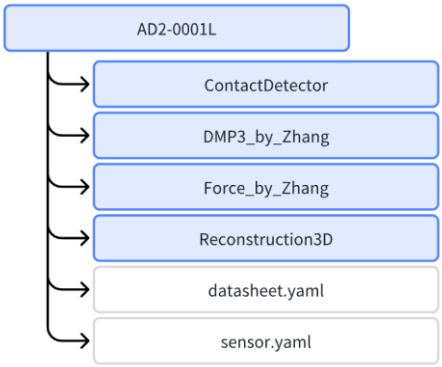
3.3 从命令行启动

如果您使用的是通过 Type-C 接口直接连接电脑的 Tac3D 触觉传感器，则需要将 Tac3D 的主程序将运行在连接 Tac3D 触觉传感器的电脑上。在无图形界面的设备上可以通过命令行启动 Tac3D 触觉传感器的主程序。window 端 Tac3D 触觉传感器的主程序为“Tac3D-Core/windows-x86_64”目录下的“Tac3D.exe”文件。Linux 端 Tac3D 触

觉传感器的主程序为“Tac3D-Core/linux-x86_64”目录下的“Tac3D”文件。

3.3.1 解压配置文件

Tac3D 配置文件，如“AD2-0001L-v3.2.1.tcfg”实际为一个 zip 格式的压缩包。可通过任何解压缩工具将此文件解压，解压后将得到一个文件结构如下的文件夹：



这里，直接包含文件“sensor.yaml”文件的目录为配置文件夹的主目录。

若在此之前，已经在 Tac3D-Desktop 应用程序中导入了配置文件。则可以在 Tac3D-Desktop 的“config”文件夹中找到以传感器序列号命名的已经解压好的配置文件夹。

在使用命令行调用 Tac3D 触觉传感器之前，将解压好的配置文件夹移动到“Tac3D-Core/windows-x86_64/config”或“Tac3D-Core/linux-x86_64/config”目录下。

3.3.2 在 windows 上使用 cmd 命令行启动 Tac3D 主程序

在 windows 命令行中调用 Tac3D.exe 时，命令行的工作目录应位于“Tac3D-Core/windows-x86_64”。启动命令和参数为：

```
Tac3D.exe -c configDir [-d cameraIndex] [-i SDK_ip -p SDK_port] [-h]
```

3.3.3 在 windows 上使用 powershell 启动 Tac3D 主程序

在 windows 的 powershell 中调用 Tac3D.exe 时，命令行的工作目录应位于“Tac3D-Core/windows-x86_64”。启动命令和参数为：

```
./Tac3D.exe -c configDir [-d cameraIndex] [-i SDK_ip -p SDK_port] [-h] [-v]
```

3.3.4 在 windows 上启动 Tac3D 主程序

在 linux 上调用 Tac3D 时，命令行的工作目录应位于“Tac3D-Core/linux-x86_64”。启动命令和参数为：

```
# 首先添加可执行权限
chmod +x Tac3D
# 启动传感器主程序
./Tac3D -c configDir [-d cameraIndex] [-i SDK_ip -p SDK_port]
[-h] [-v]
```

在部分 linux 系统上，调用传感器内的摄像头需要单独的权限，可视情况在必要时使用 sudo 权限启动传感器主程序。

3.3.5 选项和参数说明

选项	参数说明
-c （必要）	有参数，为所启动的 Tac3D 触觉传感器的配置文件夹路径，在 config 目录下以各传感器的序列号命名，如 A1-0001R。需要注意的是，配置文件夹必须是必须包含 sensor.yaml 文件的。调用前应检查所传入的 configDir 目录中是否包含 sensor.yaml 文件的。
-d （可选）	有参数，为 Tac3D 触觉传感器的相机索引号。该索引号与使用 OpenCV 的 cv::VideoCapture()打开 Tac3D 触觉传感器的相机时传入的相机索引号一致。
-i （可选）	有参数，为接收触觉数据的 Tac3D-API 端的计算机 IP 地址。默认值为本地地址 127.0.0.1。 -i 和-p 参数必须同时使用才会生效。
-p （可选）	有参数，为接收触觉数据的 Tac3D-API 端程序使用的 UDP 接收端口。默认为 9988。 -i 和-p 参数必须同时使用才会生效。
-h （可选）	无参数，显示帮助
-v （可选）	无参数，显示软件版本

windows 命令例程（cmd）：

```
Tac3D.exe -c config/AD2-0001L -d 0 -i 127.0.0.1 -p 9988
```

windows 命令例程（powershell）：

```
./Tac3D.exe -c config/AD2-0001L -d 0 -i 127.0.0.1 -p 9988
```

Linux 命令例程：

```
./Tac3D -c config/AD2-0001L -d 0 -i 127.0.0.1 -p 9988
```

上述命名含义为使用 config 目录下的 A1-0001R 文件夹中的配置文件，启动序列号为 AD2-0001L 的 Tac3D 触觉传感器； “-d 0”指 AD2-0001L 的相机的索引号为 0（即在 OpenCV 中使用 cv::VideoCapture(0)可以打开的应当是 A1-0001R 传感器的相机），-d 参数的具体取值以实际 AD2-0001L 相机的索引号为准；“-i 127.0.0.1 -p 9988”指定 SDK 端的 IP 地址和端口分别为 127.0.0.1 和 9988。

4 基于 Tac3D-API 的二次开发

4.1 libTac3D

libTac3D 是 Tac3D-API 的 C++版本，在 Tac3D-API/libTac3D 目录下。libTac3D 以源代码形式提供，src 目录下的 libTac3D.cpp 为源文件，inc 目录下的 libTac3D.hpp 为头文件。Tac3D-API/libTac3D 目录下的 main-example.cpp 为 libTac3D 的使用示例。Tac3D-API/libTac3D 目录下提供了编译示例所需的 CMakeLists.txt 文件。

4.1.1 Windows 平台的环境配置和编译

鉴于科研工作中的机器人控制多基于 Linux 平台，libTac3D 也是主要基于 Linux 平台开发。当前版本的 libTac3D 对 Windows 的适配较弱。若需要在 Windows 平台进行基于 libTac3D 的二次开发，环境配置过程较为繁琐。因此，现阶段不推荐在 Windows 平台进行基于 libTac3D 的开发。但这里仍然对 Windows 平台的环境配置和编译方法进行介绍。

libTac3D 的 Windows 端是从 Linux 端迁移而来，使用 posix 标准 API，因此，在 windwos 系统下编译时需要使用 MinGW 配置 posix 标准 API 的编译环境。

libTac3D 依赖两个第三方库，OpenCV 和 Yaml-cpp。由于这些库在 Windows 平台的主流 Release 版本是用 MSVC 编译的，使用 gcc 编译基于 libTac3D 开发的软件时无法 gcc 正确地链接这些库。因此，需要下载 OpenCV 和 Yaml-cpp 源代码，自行使用基于 posix 标准 API 的 gcc/g++编译库文件。

在完成环境配置后，可通过编译示例测试环境配置是否正确。打开 cmd 命令行，cd

到 Tac3D-API/libTac3D 目录下。通过以下命令编译：

```
md build
cd build
cmake -G "MinGW Makefiles" -DCMAKE_CXX_FLAGS="-std=c++11 -
Wall -Wpedantic" ..
make
```

若编译过程无报错，并在 build 目录下找到 libTac3D-example.exe 文件，则说明环境配置正确且已完成编译。

4.1.2 Linux 平台的环境配置和编译

Linux 平台自带 gcc 编译器，环境配置过程较为简单。推荐在 Linux 平台进行基于 libTac3D 的二次开发。

libTac3D 依赖两个第三库，OpenCV 和 Yaml-cpp。这两个库都可以通过以下命令从 apt 源直接安装：

```
# 更新软件列表:
sudo apt update
# 安装 C++编译和调试所需的环境:
sudo apt-get install build-essential gdb cmake
# 安装 OpenCV 和 yaml-cpp:
sudo apt-get install libopencv-dev libyaml-cpp*
```

在完成环境配置后，可通过编译示例测试环境配置是否正确。在 Tac3D-API/libTac3D 目录下打开新终端。通过以下命令编译：

```
mkdir build
cd build
cmake ..
make
```

若编译过程无报错，并在 build 目录下找到 libTac3D-example 文件，则说明环境配置正确且已完成编译。

4.1.3 libTac3D 的 API 说明

首先,使用 libTac3D 需包含头文件 libTac3D.hpp。libTac3D 使用命名空间 Tac3D。

```
#include "libTac3D.hpp"
```

- **类：Tac3D::Sensor**

Tac3D::Sensor 类是 libTac3D 使用中最主要的一个类。当 Tac3D::Sensor 实例的构造函数调用后，将立即开始在单独的线程中接收 Tac3D-Desktop 发送的触觉信息数据帧。并在收到数据帧后，自动调用用户传入的回调函数处理数据帧。此外，Tac3D::Sensor 类提供了向 Tac3D-Desktop 发送校准信号的接口。

成员函数：

```
Tac3D::Sensor::Sensor(void (*recvCallback, void *param)(Frame &frame), int port, void* callbackParam = NULL)
```

功能：

Tac3D::Sensor 类的构造函数。

参数：

recvCallback：接收 Tac3D 触觉信息数据帧的回调函数。在每次收到一个完整的 Tac3D-Desktop 传来的数据帧后，将自动调用一次该回调函数。回调函数的参数为所接收到的触觉数据帧（见 Tac3D::Frame 类的介绍）。

port：接收 Tac3D-Desktop 传来的数据的 UDP 端口。需要注意此端口应与在 Tac3D-Desktop 中设置的数据接收端口一致。

callbackParam：回调函数的自定义参数，指针类型。这里在 Tac3D:: Sensor 初始化时传给 callbackParam 的参数将在回调函数被调用时作为 recvCallback 函数的第二个参数传给 recvCallback 函数中，以便在回调函数 recvCallback 中可以访问 callbackParam 指针指向的空间、对象或结构体。

成员函数：

```
void Tac3D::Sensor::waitForFrame(void)
```

功能：

阻塞等待接收数据帧，直到接收到第一帧数据帧为止。需注意，此函数的用途通常为等待 Tac3D-Desktop 启动传感器，而非在接收到一帧的数据后等待下一帧数据。若在调用此函数之前已经成功接收到来自 Tac3D-Desktop 的数据，则在此函数中不会进行任何阻塞等待。

参数：

无

返回值：

无

成员函数：

```
void Tac3D::Sensor::calibrate(std::string SN)
```

功能：

向 Tac3D-Desktop 发送一次传感器校准信号，要求设置当前的 3D 变形状态为无变形是的状态以消除“温漂”或“光飘”的影响。相当于在 Tac3D-Desktop 端点击一次“零位校准”按键的效果。

参数：

SN：要求校准的传感器的 SN 码。

返回值：

无

- **类：Tac3D::Frame**

Tac3D::Frame 是触觉信息数据帧类。所接收到的触觉数据帧以 Tac3D::Frame 的形式传递给回调函数。

成员变量：

```
std::string Tac3D::Frame::SN
```

类型：

字符串

说明：

传感器 SN 码：所接收到的数据帧来自的触觉传感器的 SN 码，用于标记触觉数据帧的来源。

成员变量：

```
uint32_t Tac3D::Frame::index
```

类型：

32 位无符号整型

说明：

帧序号：自每一个触觉传感器启动时从 0 开始计数的帧编号。每个触觉传感器独立计数。但由于每个触觉传感器在启动时有一段初期校准采样，触觉数据帧的传输是从初期校准采样结束后开始的。故实际接收到的触觉数据帧的序号并不是从 0 开始的。

成员变量：

```
double Tac3D::Frame::sendTimestamp
```

类型：

64 位浮点数

说明：

数据发送时间戳：自每一个触觉传感器在 Tac3D-Desktop 上启动时从 0 开始计算的触觉数据帧发送时间。该时间由 Tac3D-Desktop 端计算。

成员变量：

```
double Tac3D::Frame::recvTimestamp
```

类型：

64 位浮点数

说明：

数据接收时间戳：自每一个 Tac3D::Sensor 实例初始化时从 0 开始计算的接收到触觉数据帧的时间。该时间由 Tac3D::Sensor 端计算。

成员函数：

```
template <typename T>
T* Tac3D::Frame::get(std::string fieldName)
```

功能：

从 Tac3D::Frame 触觉数据帧中取出所需的数据。注意，使用 Tac3D::Frame::get 函数仅能获得指向所需数据的指针。这些指针所指向的内存空间的数据会在回调函数结束后被新接收到的数据覆盖。因此，在回调函数中务必先将数据完整拷贝到用户自己申请的内存中或创建的变量中再进行后续的操作。

参数：

T：需要获取的数据类型。

fieldName：需要获取的数据名称。

当前版本的 Tac3D-A1 系列传感器提供以下三种触觉信息，对应的数据名称和数据类型如下表：

表 5-1 触觉数据帧中的数据名称和数据类型

触觉信息	数据名称 fieldName	数据类型 T	单位
三维形貌	3D_Positions	cv::Mat	mm
三维变形场	3D_Displacements	cv::Mat	mm

三维分布力	3D_Forces	cv::Mat	N
三维合力	3D_ResultantForce	cv::Mat	N
三维合力矩	3D_ResultantMoment	cv::Mat	N·mm

三维形貌、三维变形场和三维分布力为 400 行 3 列的矩阵。矩阵的 1~400 行分别对应 0~399 号标志点，矩阵的每一列分别对应 x、y、z 方向的分量值。三维合力和三维合力矩为 1 行 3 列的矩阵，三个元素分别为合力/合力矩的 x、y、z 分量。(有关坐标系的定义请参阅《Tac3D-AD2 触觉传感器产品规格书》)。

返回值:

指向所需获取的数据的指针。若 Tac3D::Frame 中未找到对应数据名称的数据则返回 NULL。

4.1.4 libTac3D 例程说明

在 Tac3D-API/libTac3D 目录下的 main-example.cpp 为 libTac3D 的例程。

首先引用头文件

```
#include "libTac3D.hpp"
```

声明用于存储接收到的数据的用户变量

```
cv::Mat P, D, F, Fr, Mr; // 用于存储三维形貌、三维变形场、三维分布力、三维合力、三维合力矩数据的矩阵
int frameIndex;  // 帧序号
double sendTimestamp, recvTimestamp; // 时间戳
std::string SN;  // 传感器 SN
```

编写接收触觉数据帧的回调函数:

```
void Tac3DRecvCallback(Tac3D::Frame &frame, void *param)
{
    cv::Mat *tempMat;
    // float* testParam = (float*)param; // 接收自定义参数（在 Tac3D::Sensor 初始化时传递）
    SN = frame.SN;  // 获得传感器 SN 码，可用于区分触觉信息来源于哪个触觉传感器
    frameIndex = frame.index;  // 获得帧序号
```

```
    sendTimestamp = frame.sendTimestamp;  // 获得发送时间戳
    recvTimestamp = frame.recvTimestamp;  // 获得接收时间戳

    // 使用 frame.get 函数通过数据名称"3D_Positions"获得 cv::Mat 类型的三维形貌的数据指针
    tempMat = frame.get<cv::Mat>("3D_Positions");
    tempMat->copyTo(P); // 务必先将获得的数据拷贝到自己的变量中再使用(注意,OpenCV 的 Mat 中数据段和矩阵头是分开存储的,因此需要使用 copyTo 同时复制矩阵的矩阵头和数据段，而不应当用=符号赋值)

    // 使用 frame.get 函数通过数据名称"3D_Displacements"获得 cv::Mat 类型的三维变形场的数据指针
    tempMat = frame.get<cv::Mat>("3D_Displacements");
    tempMat->copyTo(D); // 务必先将获得的数据拷贝到自己的变量中再使用

    // 使用 frame.get 函数通过数据名称"3D_Forces"获得 cv::Mat 类型的三维分布力的数据指针
    tempMat = frame.get<cv::Mat>("3D_Forces");
    tempMat->copyTo(F); // 务必先将获得的数据拷贝到自己的变量中再使用

    // 使用 frame.get 函数通过数据名称"3D_ResultantForce"获得 cv::Mat 类型的三维合力的数据指针
    tempMat = frame.get<cv::Mat>("3D_ResultantForce");
    tempMat->copyTo(Fr); // 务必先将获得的数据拷贝到自己的变量中再使用

    // 使用 frame.get 函数通过数据名称"3D_ResultantForce"获得 cv::Mat 类型的三维合力的数据指针
    tempMat = frame.get<cv::Mat>("3D_ResultantMoment");
    tempMat->copyTo(Mr); // 务必先将获得的数据拷贝到自己的变量中再使用
}
```

需要注意的是，不应将对机器人系统的控制逻辑或其他耗时的程序放在回调函数中执行。否则可能因触觉数据帧处理不及时导致数据堆积和严重的数据滞后。

在 main 函数中初始化 Tac3D 实例启动数据接收：

```
int main(int argc,char **argv)
{
    float testParam = 100.0;
    Tac3D::Sensor tac3d(Tac3DRecvCallback, 9988, &testParam); // 创建 Sensor 实例，设置回调函数为上面写好的 Tac3DRecvCallback，设置 UDP 接收端口为 9988，设置自定义参数为 testParam
    tac3d.waitForFrame(); // 等待 Tac3D-Desktop 端启动传感器并建立连接

    usleep(1000*1000*5); // 5s
    tac3d.calibrate(SN); // 发送一次校准信号（应确保校准时传感器未与任何物体接触！否则会输出错误的数据！）

// 例程中没有其他需要执行的任务，故使用 while 循环阻止主程序退出
    while (1)
    {
        usleep(1000*1000);
    }
    return 0;
```

4.2 PyTac3D

PyTac3D 实现了基于 Python 的数据采集端。PyTac3D 中不含主程序端功能，因此 PyTac3D 需要搭配主程序（windows 下的 Tac3D.exe 或 linux 下的 Tac3D）或 Tac3D-Desktop 程序使用。

PyTac3D 是 Tac3D-API 的 Python 版本，在 Tac3D-API/PyTac3D 目录下。PyTac3D 以源代码形式提供，PyTac3D.py 为库文件，main-example.cpp 为 PyTac3D 的使用示例。此外，PyTac3D_Displayer.py 提供了一个基于 PyTac3D 和 vedo 开发的简易三维显示界面供开发参考。

4.2.1 环境配置

PyTac3D 开发使用的 Python 版本为 3.7.9。推荐使用与 3.7 接近的 Python 版本。

PyTac3D 依赖的第三方库包括 numpy 和 ruamel.yaml。安装命令如下：

```
pip install numpy ruamel.yaml
```

PyTac3D_Displayer 在以上依赖库的基础上还额外依赖 vedo（2023.4.6 版本）。若

需要使用 PyTac3D_Displayer 则需安装 2023.4.6 版本的 vedo。安装命令如下：

```
pip install vedo==2023.4.6
```

4.2.2 PyTac3D 的 API 说明

首先，导入 PyTac3D。

```
import PyTac3D
```

- **类：PyTac3D.Sensor**

PyTac3D.Sensor 类是 PyTac3D 使用中最主要的一个类。当 PyTac3D.Sensor 实例的初始化后，将立即开始在单独的线程中接收 Tac3D-Desktop 发送的触觉数据帧。用户可通过两种从 PyTac3D.Sensor 对象获取数据帧。一种是与 libTac3D 类似的通过回调函数获取，另一种是通过 PyTac3D.Sensor.getFrame()函数获取。此外，PyTac3D.Sensor 类提供了向 Tac3D-Desktop 发送校准信号的接口。

成员函数：

```
PyTac3D.Sensor.__init__ (self, recvCallback = None, port =
9988, maxQSize = 5, callbackParam = None)
```

PyTac3D.Sensor 类的初始化函数。

参数：

recvCallback：接收 Tac3D 触觉信息数据帧的回调函数。在每次收到一个完整的 Tac3D-Desktop 传来的数据帧后，将自动调用一次该回调函数。回调函数的参数为所接收到的触觉数据帧（数据帧的介绍见 PyTac3D.getFrame()函数的介绍部分）。

port：接收 Tac3D-Desktop 传来的数据的 UDP 端口。需要注意此端口应与在 Tac3D-Desktop 中设置的数据接收端口一致。

maxQSize：最大接收队列长度。在使用 Sensor.getFrame()获取触觉数据帧时，数据帧缓存队列的最大长度。（若使用 recvCallback 方法获取触觉数据帧，则不受此参数的影响）

callbackParam：回调函数的自定义参数。这里在 PyTac3D.Sensor 初始化时传给 callbackParam 的参数将在回调函数被调用时作为回调函数 recvCallback 的第二个参数传给回调函数中，以便在回调函数中可以访问通过 callbackParam 传入的对象。

成员函数：

```
PyTac3D.Sensor.waitForFrame(self)
```

功能：

阻塞等待接收数据帧，直到接收到第一帧数据帧为止。需注意，此函数的用途通常为等待 Tac3D-Desktop 启动传感器，而非在接收到一帧的数据后等待下一帧数据。若在调用此函数之前已经成功接收到来自 Tac3D-Desktop 的数据，则在此函数中不会进行任何阻塞等待。

参数：

无

返回值：

无

成员函数：

```
PyTac3D.Sensor.calibrate(self, SN)
```

功能：

向 Tac3D-Desktop 发送一次传感器校准信号，要求设置当前的 3D 变形状态为无变形是的状态以消除“温漂”或“光飘”的影响。相当于在 Tac3D-Desktop 端点击一次“零位校准”按键的效果。

参数：

SN：要求校准的传感器的 SN 码字符串。

返回值：

无

成员函数：

```
PyTac3D.Sensor.getFrame(self)
```

功能：

获取缓存数据帧队中的数据帧。当缓存数据帧队列不为空时，则返回队列最前端的帧。当缓存数据帧队列为空时则返回 None。

参数：

SN：要求校准的传感器的 SN 码字符串。

返回值：

frame：数据帧，数据类型为字典，其结构如下。

```
{
    "SN": （字符串）传感器 SN 码——所接收到的数据帧来自的触
```

```
        觉传感器的 SN 码，用于标记触觉数据帧的来源
    "index": （整形）帧序号--自每一个触觉传感器启动时从 0
        开始计数的帧编号。每个触觉传感器独立计数。但由于每
        个触觉传感器在启动时有一段初期校准采样，触觉数据帧
        的传输是从初期校准采样结束后开始的。故实际接收到的
        触觉数据帧的序号并不是从 0 开始的。
    "sendTimestamp": （浮点数）数据发送时间戳--自每一个触
        觉传感器在 Tac3D-Desktop 上启动时从 0 开始计算的触觉数
        据帧发送时间。该时间由 Tac3D-Desktop 端计算。
    "recvTimestamp": （浮点数）数据接收时间戳--自每一个
        Sensor 实例初始化时从 0 开始计算的接收到触觉数据帧的时
        间。该时间由 Tac3D::Sensor 端计算。
    "3D_Positions": （numpy.array）三维形貌--是一个形状为
        400 行 3 列的二维数组。数组的每一行分别对应一个标志点，
        数组的每一列分别对应标志点的 x、y、z 坐标。
    "3D_Displacements": （numpy.array）三维位移场--是一个
        形状为 400 行 3 列的二维数组。数组的每一行分别对应一个标
        志点，数组的每一列分别对应标志点的 x、y、z 方向位移。
    "3D_Forces": （numpy.array）三维位移场--是一个形状为 400
        行 3 列的二维数组。数组的每一行分别对应一个标志点，数
        组的每一列分别对应标志点附近区域的 x、y、z 方向受力。
    "3D_ResultantForce": （numpy.array）三维合力--是一个
        形状为 1 行 3 列的二维数组。三个元素分别为合力的 x、y、z 分量。
。
    " 3D_ResultantMoment ": （numpy.array）三维合力矩--是一个形状
为 1
        行 3 列的二维数组。三个元素分别为合力矩的 x、y、z 分量。
}
```

4.2.3 PyTac3D 例程说明

在 Tac3D-API/PyTac3D 目录下的 main-example.py 为 PyTac3D 的例程。

首先导入 PyTac3D，同时导入例程所需的 time 包

```
import PyTac3D
```

```
import time
```

例程中用到的存储获取的数据的变量

```
SN = ''  # 传感器 SN
frameIndex = -1 # 帧序号
sendTimestamp = 0.0 # 时间戳
recvTimestamp = 0.0 # 时间戳

# 用于存储三维形貌、三维变形场、三维分布力、三维合力、三维合力矩数据的矩阵
P, D, F, Fr, Mr = None, None, None, None, None
```

编写接收触觉数据帧的回调函数：

```
def Tac3DRecvCallback(frame, param):
    global SN, frameIndex, sendTimestamp, recvTimestamp, P, D, F
    # 获取 SN
    print(param) # 显示自定义参数

    SN = frame['SN']
    print(SN)

    # 获取帧序号
    frameIndex = frame['index']
    print(frameIndex)

    # 获取时间戳
    sendTimestamp = frame['sendTimestamp']
    recvTimestamp = frame['recvTimestamp']

    # 获取标志点三维形貌
    # P 矩阵（numpy.array）为 400 行 3 列，400 行分别对应 400 个标志点，3 列分别为各标志点的 x，y 和 z 坐标
    P = frame.get('3D_Positions')

    # 获取标志点三维位移场
    # D 矩阵为（numpy.array）400 行 3 列，400 行分别对应 400 个标志点，3 列分别为各标志点的 x，y 和 z 位移
    D = frame.get('3D_Displacements')
```

```
    # 获取三维分布力
    # F 矩阵为（numpy.array）400 行 3 列，400 行分别对应 400 个标志点，
3 列分别为各标志点附近区域所受的 x，y 和 z 方向力
    F = frame.get('3D_Forces')

    # 获得三维合力
    # Fr 矩阵为 1x3 矩阵，3 列分别为 x，y 和 z 方向合力
    Fr = frame.get('3D_ResultantForce')

    # 获得三维合力矩
    # Mr 矩阵为 1x3 矩阵，3 列分别为 x，y 和 z 方向合力矩
        Mr = frame.get('3D_ResultantMoment')
```

需要注意的是，不应将对机器人系统的控制逻辑或其他耗时的程序放在回调函数中执行。否则可能因触觉数据帧处理不及时导致数据堆积和严重的数据滞后。

初始化 Tac3D 实例启动数据接收：

```
# 创建 Sensor 实例，设置回调函数为上面写好的 Tac3DRecvCallback，设置
UDP 接收端口为 9988，数据帧缓存队列最大长度为 5
tac3d = PyTac3D.Sensor(recvCallback=Tac3DRecvCallback,
port=9988, maxQSize=5, callbackParam = 'test param')

# 等待 Tac3D-Desktop 端启动传感器并建立连接
tac3d.waitForFrame()

time.sleep(5) # 5s

# 发送一次校准信号（应确保校准时传感器未与任何物体接触！否则会输出错误的
数据!)
tac3d.calibrate(SN)

time.sleep(5) #5s

# 获取 frame 的另一种方式：通过 getFrame 获取缓存队列中的 frame
frame = tac3d.getFrame()
if not frame is None:
    print(frame['SN'])

# 例程中没有其他需要执行的任务，故使用 while 循环阻止主程序退出
while True:
```

```
    time.sleep(1)
```

5 Tac3D 主程序异常退出错误码与解决方法

错误码	错误信息	解决方法
0xF01	无效的配置文件	1. 请检查配置文件夹的路径是否正确（文件夹中应包含 sensor.yaml 文件） 2. 请使用随传感器附带的原版配置文件 3. 请确保使用的 Tac3D 主程序版本不低于配置文件的版本
0xF02 0xF03 0xF04	网络配置错误	1. 请检查当前计算机是否有可用的网卡和网络端口
0xF05	数据接收网络地址无效	1. 请检查所设置的数据接收网络地址是否正确
0xF06	参数错误	1. 请检查启动 Tac3D 主程序时传入的命令行参数是否正确
0xF07	传感器硬件连接失败	1. 请检查传感器是否连接到电脑，并确认电脑上带有一般摄像头的驱动程序 2. 请检查启动传感器时设定的相机编号（命令中-d 的参数）是否正确
0xF08	传感器与配置文件不匹配	1. 请检查启动传感器时设定的相机编号（命令中-d 的参数）对应的传感器 SN 是否与所使用的配置文件匹配
0xF0A	传感器连接断开	1. 请检查传感器和计算机直接的物理连接是否中断

6 常见问题 FAQ

1. 在 Tac3D-Desktop 打开传感器后，Tac3D-API 端接无法接收到数据

请按照以下原因排查此问题：

① 检查在 Tac3D-Desktop 打开传感器时的网络传输设置是否正确：“网络传输功能”应为“true”状态；“接收地址”应为运行 Tac3D-API 端程序的计算机 IP 地址，若运行 Tac3D-Desktop 和 Tac3D-API 的是同一台计算机，此 IP 可设置为回环地址“127.0.0.1”；“接收端口”应与 Tac3D-API 端程序中指定的数据接收端口一致，此端口默认为 9988。

② 若运行 Tac3D-Desktop 和 Tac3D-API 的不是同一台计算机，请检查两台计算机是否处于同一可互相通信的局域网内，以及所填写的“SDK 端 IP”是否是计算机在局域网内的 IP 地址。

③ 若运行 Tac3D-Desktop 和 Tac3D-API 的不是同一台计算机，请检查运行 SDK 端程序的计算机的防火墙规则是否允许接收所指定的“接收端口”的 UDP 数据。

2. 在 Tac3D 传感器帧率未达到 30 FPS

请按照以下原因排查此问题：

① 请检查计算机的电源设置是否处于省电模式。笔记本电脑在未连接外接电源的情况下会默认降低性能以减小功耗。部分笔记本电脑可尝试将电源线重新拔插一次以切换到非省电模式。

② Tac3D 的触觉信息重建程序的计算开销较大，部分配置不足、发布较早或轻薄型电脑可能无法达到 30 FPS 运行的性能要求。对于电脑性能不足的情况，可尝试将传感器的配置文件夹中“sensor.yaml”文件中第 28 行处（v3.2.1 版本配置文件）“fastMode”选项设置为“true”以降低性能开销。但需注意，更改此设置可能影响 Tac3D 触觉传感器在部分特殊工况下的测量稳定性。

③ 在一台计算机上同时使用两个 Tac3D 触觉传感器时，需将传感器直接连接至电脑主板的 USB 接口上（笔记本电脑自带的 USB 接口或台式机背面主板上的 USB 接口）。若使用 USB 扩展坞或台式机前面板的 USB 接口可能因 USB 带宽不足导致帧率降低。

④ 若使用的是 windows11 操作系统，则可能由于操作系统限制后台软件的资源占用导致传感器帧率降低。此问题的典型表现是，在传感器刚启动时可以达到 30FPS 帧率，但将窗口放在后台一段时间后帧率下降至月 20FPS。针对此问题，可参阅此链接 https://stackoverflow.com/questions/76152777/what-is-this-win11-feature-that-limits-compute-resources，在具有管理员权限的命令行中使用如下命令关闭对 Tac3D.exe 的省电策略：

```
powercfg /powerthrottling disable /path "PATH_TO_TAC3D_EXE"
```

将命令中的 PATH_TO_TAC3D_EXE 替换为 Tac3D.exe 文件的完整路径。若使用 Tac3D-Desktop，对应的 Tac3D.exe 文件位于 Tac3D-Desktop 软件目录的“api”文件夹中。

3. 传感器在大变形时出现卡顿、抽帧问题

请按照以下原因排查此问题：

① 局部变形量过大导致测量失效。Tac3D-AD2 触觉传感器的允许的最大变形量为感知表面边缘处 2mm，感知表面中心处 3mm，同时请避免施加过大的局部载荷，以防止测量失效或传感器损坏。

② 变形速度过快导致测量失效。此问题通常出现在触觉传感器受到冲击载荷的瞬间。

4. 弹性体未接触物体时 3D 位移场和 3D 分布力的噪点很多

请按照以下原因排查此问题：

① 触觉传感器在启动时或在校准时，弹性体存在受力变形。请撤去与弹性体表面接触的物体，确保弹性体处于未受力状态时点击校准按钮或发送校准指令。并确保在校准过程中（约

3 秒钟）弹性体未受力变形。

5. 使用 3D 分布力求和计算的合力不准确

请按照以下原因排查此问题：

① Tac3D 触觉传感器为降低 3D 分布力中的噪声和增强对接触特征的呈现，对分布力进行了一定的降噪和非接触区抑制处理。部分处理方法是基于图形学的而非力学的，其对每个测力点的测力值的微小调整可因求和而呈现系统性偏差。考虑到部分实验中仍然有使用合力信息的需求，Tac3D 触觉传感器提供了单独计算的合力数据。该数据相比于 3D 分布力求和更准确。使用“3D_ResultantForce”数据名称可从触觉数据帧中直接获取该合力数据。不过仍需要注意的是，Tac3D 触觉传感器作为一种测量分布信息的传感器，其所能提供的合力数据的准确性是有限的。这种差异类似于 RGB 相机和光度计在测量光强上的性能差距。在对数值精度要求较高的定量实验中请谨慎使用来自 Tac3D 触觉传感器的合力数据。

6. 通过数据名称“3D_ResultantForce”获得的合力不准确

请按照以下原因排查此问题：

① 施力物体接触到了弹性体上感知表面范围以外的区域。Tac3D 触觉传感器的分布力解算方法假定弹性体所受载荷仅限于感知表面范围之内。使用 Tac3D 触觉传感器测量分布力时，请确保弹性体上仅在感知表面范围之内受到载荷。

② Tac3D 触觉传感器的外壳（除安装连接件的位置）在工作中不应受力。若必须手持 Tac3D 触觉传感器进行演示或测试，应在 Tac3D 触觉传感器上安装附带的连接件，并手持连接件一端。否则，外力造成传感器外壳的额外变形会改变已标定好的传感器结构参数，导致测量零点发生偏移。

③ Tac3D 触觉传感器内部的摄像头会因工作时发热而产生温漂。温漂会使 Tac3D 触觉传感器提供的 Z 方向合力大小系统性地向 Z+方向偏离。在使用 Tac3D 触觉传感器数据前，将传感器打开预热 5 分钟，可大幅减小温漂对合力计算的影响。

⑤ 测量表面内受到大面积均匀分布的沿表面方向的力（如抓取气球、水球或与弹性体表面形状高度接近的凹面）。根据弹性体力学理论论证，Tac3D 触觉传感器所采用的弹性体结构在上述形式的分布力作用下的测量灵敏度较低。此外，上述工况下使用的物体在与传感器的接触中容易接触到弹性体测量表面以外的区域。使用时应避免此类工况。

⑥ 10 N 以上的测力范围未经过标定，此外在过大的载荷下弹性体可表现出较为显著的非线性力学特性。使用时应避免此类工况。

7. 通过数据名称“3D_ResultantMoment”获得的合力矩不准确

此问题暂未解决，标定 Tac3D 触觉传感器的合力矩的理论方法研究尚在进行中。

7 历史版本

软件版本 3.2.1（当前版本）

发布时间：2024.11.01

适用的配置文件版本:3.2.1 及以下(更新内容适用于使用 3.2.1 版本配置文件的传感器)

1. 提升弹性体快速变形时的变形测量稳定性
2. 改善零点漂移问题
3. 增加多种滤波器
4. 传感器与配置文件序列号匹配性检测
5. 增加异常退出错误信息提示

软件版本 3.2.0

发布时间：2024.06.21

适用的配置文件版本:3.2.0 及以下(更新内容适用于使用 3.2.0 版本配置文件的传感器)

1. 提升弹性体快速变形时的变形测量稳定性
2. 增加配置文件版本检查
3. 命令行调用使用-v 参数查看软件版本

软件版本 3.1.3

发布时间：2024.03.30

适用的配置文件版本：3.1.3


Acorn
Robotics