智能机器的软件算法

简介

做这个项目算是开拓视野。当时波士顿动力的机器狗正火。我也对智能机器人挺感兴趣。坚持做了半年，最后小有所成，机器狗蹦跶蹦跶地能走起来了，颇有成就感。这个项目需要计算机系统、自动控制、AI 等多个领域知识，需要有快速自学的能力。偶尔拓展一下领域广度挺有意思，也加强了对基础学科的理解。

如果学校的线性代数、概率论、凸优化等抽象的基础学科的教育，能以做一个机器人作为兴趣辅助，将对学生的学习热情起到多么大的改善啊。若能将其原理简化做成优秀的在线兴趣课，改变应试兴趣班的风气，也是革新之举。(https://brilliant.org/)

其软件算法主要涉及到运动学、动力学、自动控制、嵌入式软件、各种传感器的参数识别校准、计算机视觉、AI 等。

此文为这半年的技术概述，间杂入门知识。内容片段大量来源于网络，仅供个人学习交流。

传感器

IMU

IMU 的减震与隔热

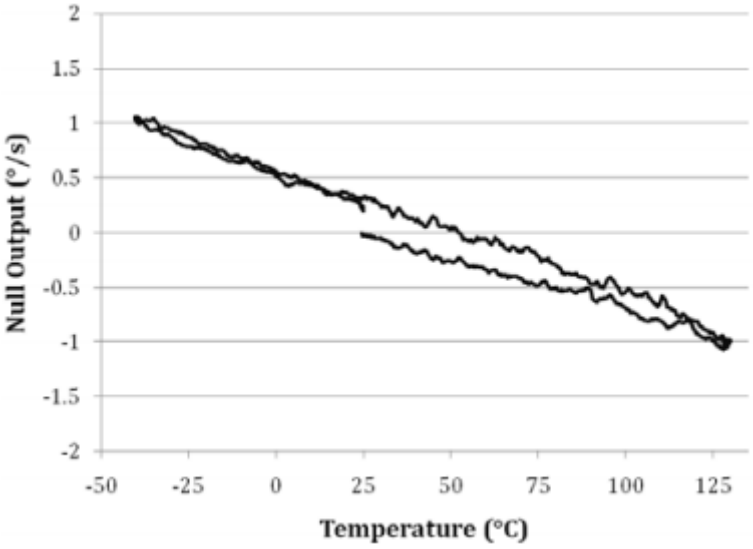
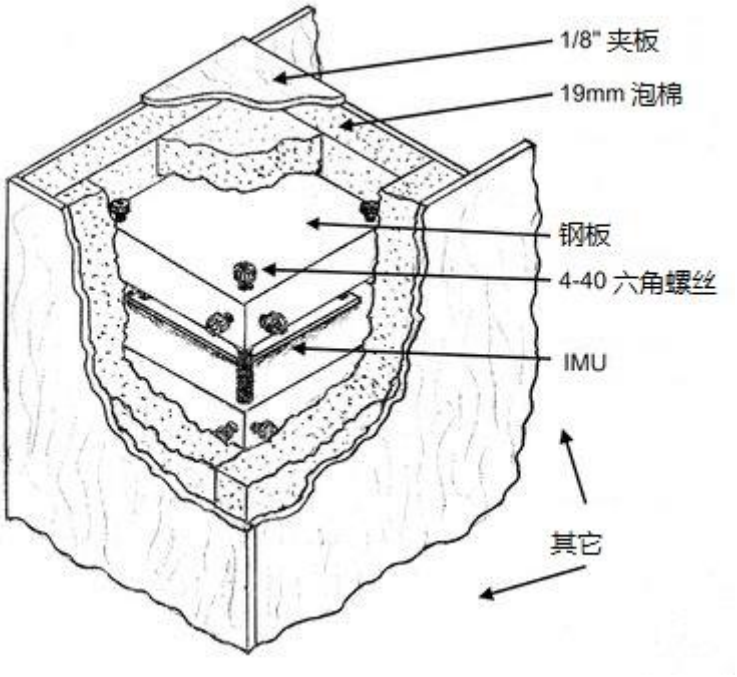


Figure 1. An Uncompensated ADXRS453 Null Bias Output While Being Cycled Over Temperature (–45°C to +130°C)

Manufacturer	Part Number	Sensitivity (°/s/g)	g2 Sensitivity (°/s/g2)	Bias Stability (°/h)
Analog Devices	ADXRS646	0.015	0.0001	8
Melexis	MLX90609	0.1	Not Specified	17
Silicon Sensing	CRG20-01	0.1	0.005	5
VTI	SCR1100-D04	0.1	Not Specified	2.1

机械减震



软件减震

有时候，受大小和重量的限制，不可能随心所欲的设计机械减振结构，或者在振动比较大的时候，机械减振效果不佳。这个时候就需要考虑采用软件滤波。比如滑动平均滤波等。但软件滤波有一个缺点，就是会有延迟，这会影响姿态控制。通常，机械减振和软件滤波都会用到，已到达好的减振效果，但需要根据实际情况在二者之间做一个权衡，以满足设计要求。

IMU 数学模型与标定

https://www.pianshen.com/article/3340535564/

深度摄像头

校准与标定

https://github.com/IntelRealSense/librealsense/blob/master/doc/t265.md

The T265's sensors (including the IMU) are calibrated on the production line, so no further calibration process is required (unlike the IMU on the D435i). Refer to the rs-ar-basic example to see how to get and use sensor intrinsics and extrinsics for the cameras and IMU.

For reference, this is the orientation of each sensor: T265 Sensor extrinsics

Timestamps

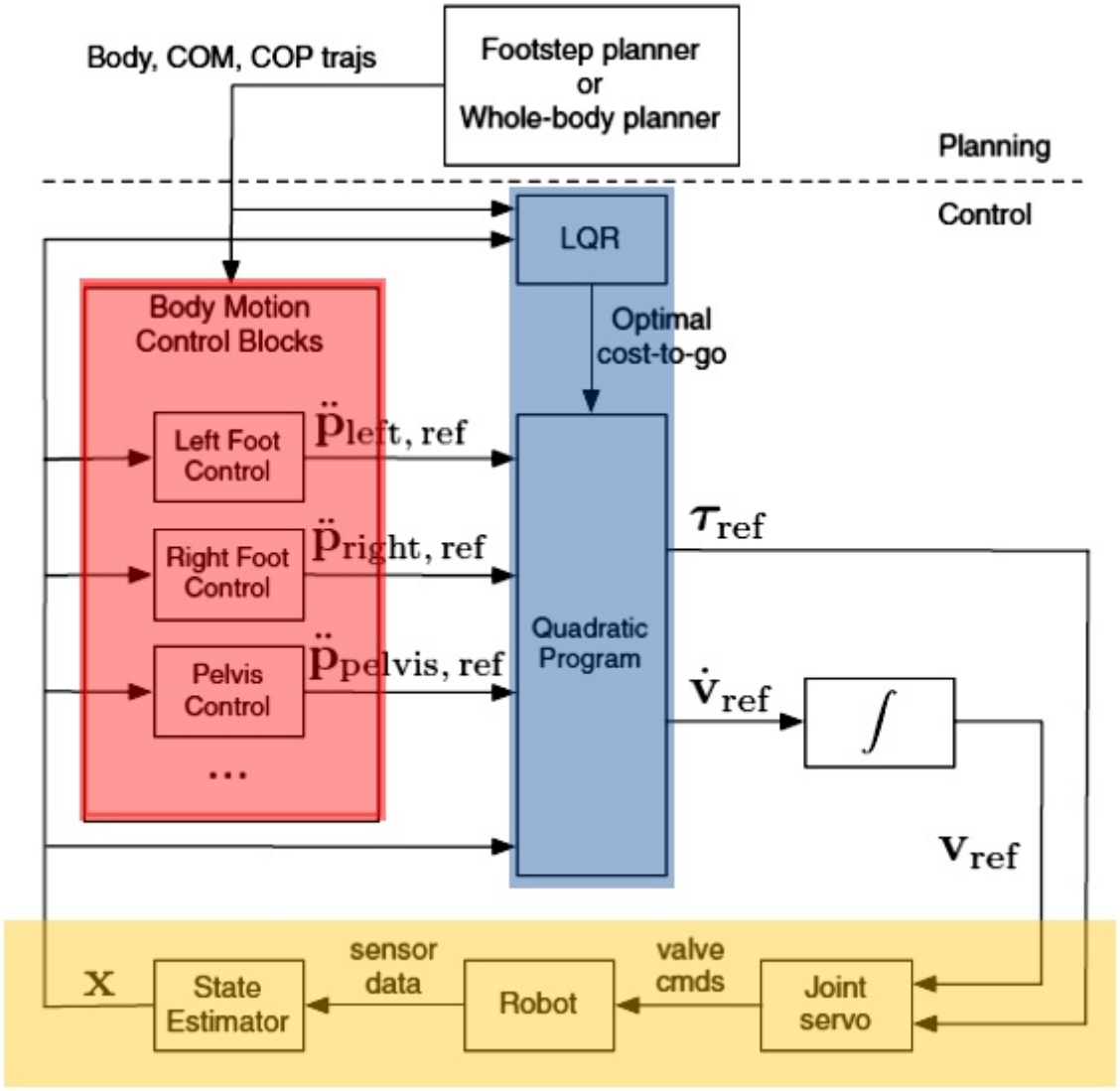
The T265 always reports the time for all sensors in RS2_TIMESTAMP_DOMAIN_GLOBAL_TIME. This uses a time synchronization mechanism between the host computer and the T265 to provide all sensor data timestamps in millisecond accurate host clock time.

激光雷达

联合标定

运动控制

多足机器人的控制涉及运动规划与运动控制两个部分



运动规划

基于贝塞尔曲线的轨迹规划

对于机器人脚部的轨迹规划问题的主要目标是：具有足够的离地间隙，从而保证机器人能跨越适当的障碍，并且要具有理想的摆动腿回缩率（适当的摆动腿回缩率可以提高机器人运动时的稳定性，适当的腿部攻角可以降低冲击能量损失），轨迹曲线通过一定数量且满足机器人运　动中速度和加速度要求的关键点，从而利用贝塞尔曲线生成腿部运动轨迹。摆动相和支 撑　相是分开设计的，支撑相的轨迹是正弦曲线，２个轨迹围绕着单个参考点来设计。这种方法使得机器人拥有更高效的运动规划，但面对复杂的地形仍然不是通用方法。

2.基于 SLIP 的运动规划

基于 SLIP 模型，机器人的步态相位可以分为飞行相和摆动相，并由此可以得到简化

有效的机器人动力学模型，对机器人运动进行有效的规划。

3.基于 CPG 的步态规划

CPG 通过模拟生物的低端神经元，从而生成机器人的步态规划。该方法利用数学方法生 成振荡曲线，将其作为腿部关节的位 置和速度输入，具有一定的自稳定能力，通过振荡曲线还可以方便地调节四足机器人腿与腿之间的相位关系。但该方法仍有很大的局限性，由于依赖已知的振荡器，尽管该方法具有一定的自稳定能力，但在面对复杂的地形时，环境对机器人的扰动很大，CPG 算法将不再适用。而野外的地形往往高度变化较大，针对该问题，Saputra 等提出了可变神经元的神经振荡器，适应性更强。

4.基于 ZMP 的步态规划

零力矩点（Zero Moment Point，ZMP），是针对静态步态稳定足式机器人的通用方法，即在机器人步态规划时，计算机器人的 ZMP（在地面上存在一点 P，使得与地面平行轴方向的、由惯性力 F=ma 与重力 G 所产生的净力矩为 0 的点）。使得 ZMP 始终位于机器人足端与地面的多个接触点所围成的多边形内，这样可以保证机器人具有理想的静平衡状态。基于 ZMP 的稳定方法在双足和四足机器人上均广泛使用，但 ZMP 算法的缺点也很明显，只适用于静态步态，对于复杂的动态步态，ZMP 算法很难应用。

运动控制

柔顺阻抗控制

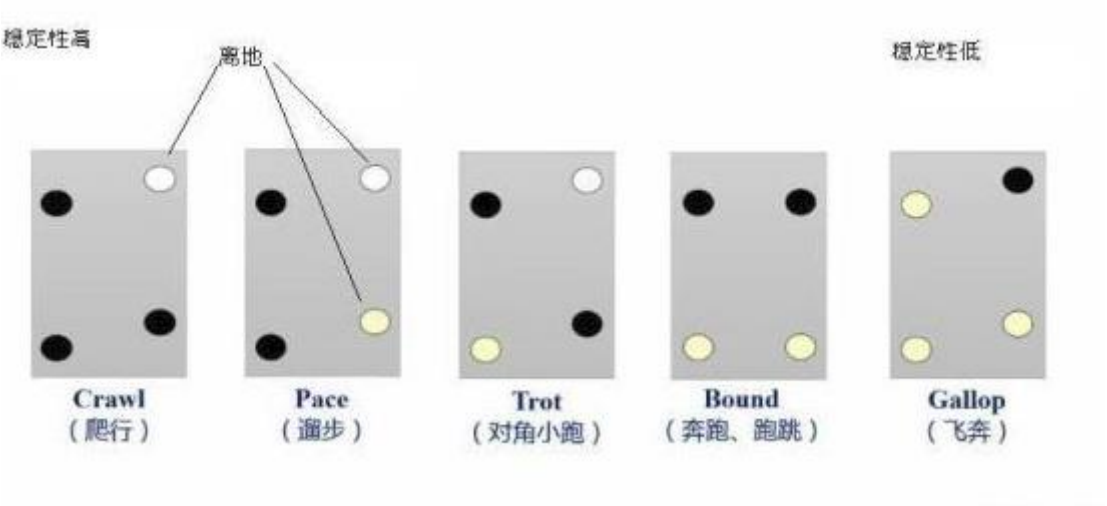
前馈控制

分层操作空间控制

模型预测控制（MPC）

神经网络自适应控制器

步态



摆动腿控制

腿足机器人低速运动情况下每条腿都会在支持相（维持机器人身体高度以及姿态）、摆动相（将腿摆到期望的位置）之间循环交替。通常的摆动腿的控制方法较为简单，《MIT Cheetah 3：Design and Control of a Robust, Dynamic Quadruped Robot》中提出摆动腿的期望目标位置为：

$$\boldsymbol{p}_{step,i} = \boldsymbol{p}_{h,i} + \underbrace{\frac{T_{c_\phi}}{2}\dot{\boldsymbol{p}}_{c,d}}_{\text{Raibert Heuristic}} + \underbrace{\sqrt{\frac{z_0}{\|\boldsymbol{g}\|}}(\dot{\boldsymbol{p}}_c - \dot{\boldsymbol{p}}_{c,d})}_{\text{Capture Point}}$$

对摆动腿末端从初始位置到目标位置进行插值（例如三次样条或者三次贝塞尔），获取每个控制周期的腿末端期望位置和速度，在操作空间进行阻抗控制，后利用关节末端相对于身体坐标系的雅可比矩阵计算该腿的各关节力矩即可。如果解算了动力学模型，此处可以加一个动力学力矩前馈：

$$\boldsymbol{\tau} = J(q)^T[K_p(p_d - p) + K_d(v_d - v)]$$

腿足机器人很多时候都会通过选择落脚点来控制机器人的速度，这一点在《legged robot that balance》中进行了详细的阐释，主要公式为如下，此处可将机器人看作一个弹簧倒立摆模型。

$$L_f = \frac{T_s}{2} v + K_v(v - v_d)$$

支撑腿控制

四足机器人的控制器进行了些调研。主要包括两类：Model-free Controller 以及 Model-based Controller.

Model-Free Controller

Model-Free Controller, 例如 Virtual Model Controller，这类控制器的优点为不依赖准确的动力学模型，建的模型简单。缺点嘛？以后补充。这种控制方法主要分为这个部分：首先，根据规划结果计算出机器人基座（躯体）应受的虚拟力与力矩；其次，根据一些评价标准将基座受到的虚拟力与力矩分配到每一个足端力，该过程通过求解凸优化问题得到；最后，通过雅克比矩阵求得关节力矩。在实际调试中，主要调节各个反馈参数来正确追踪规划好的运动轨迹，工作量并不小。ETH 在 2013 年在 StarETH 四足真机上进行了测试，具体可参考《Control of Dynamic Gaits for a Quadrupedal Robot》。

*Model-Based Controller

Model-Based Control, 顾名思义，对比于上述模型，这种方法建立了四足机器人动力学模型，ETH 称这种方法为 Whole Body Controller. 这种方法的主要思想为构造优化问题，建立躯体、腿部运动约束，以及足端接触约束，在满足约束的情况下求解最小关节力矩或最小足端力。可以参考论文《Dynamic Locomotion and Whole-Body Control for Quadrupedal Robots》。

支撑腿需要维持机器人身体的高度和姿态，即维持机器人的身体平衡，在 Cheetah 3 中有两

种平衡控制器——基于集中质量模型（lumped model）的平衡控制器和凸模型预测控制（Convex Mpc）。先阐述基于集中质量模型的平衡控制器。

集中质量模型即单刚体模型，也即可以忽略腿的作用，将地面对腿部的作用直接等效作用在质心，这种假设在机器人腿部质量相对于总质量占比很小的情况下是合理的（Cheetah 3 与 Mini Cheetah 的 4 条腿总质量都不机器人重量的 10%）。在将机器人看作单刚体模型的情况下，机器人运动方程为：

$$\underbrace{\begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ [\boldsymbol{p}_1 - \boldsymbol{p}_c]\times & \dots & [\boldsymbol{p}_4 - \boldsymbol{p}_c]\times \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{F} = \underbrace{\begin{bmatrix} m(\ddot{\boldsymbol{p}}_c + \boldsymbol{g}) \\ \boldsymbol{I}_G \dot{\boldsymbol{\omega}}_b \end{bmatrix}}_{\boldsymbol{b}},$$

所用到的也是基本的牛顿运动定律，合力=ma，合力矩=惯量*角加速度。其中 F 为作用在四条腿的地面反作用力。

通过 PD 控制律计算期望的质心加速度和角加速度：

$$\begin{bmatrix} \ddot{\boldsymbol{p}}_{c,d} \\ \dot{\boldsymbol{\omega}}_{b,d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_{p,p}(\boldsymbol{p}_{c,d} - \boldsymbol{p}_c) + \boldsymbol{K}_{d,p}(\dot{\boldsymbol{p}}_{c,d} - \dot{\boldsymbol{p}}_c) \\ \boldsymbol{K}_{p,\omega} \log(\boldsymbol{R}_d \boldsymbol{R}^T) + \boldsymbol{K}_{d,\omega}(\boldsymbol{\omega}_{b,d} - \boldsymbol{\omega}) \end{bmatrix}$$

这个 PD 方法详细描述在《High-slope terrain locomotion for torque-controlled quadruped robots》中，其实就是求了一个期望的质心状态

$$\boldsymbol{b}_d = \begin{bmatrix} m(\ddot{\boldsymbol{p}}_{c,d} + \boldsymbol{g}) \\ \boldsymbol{I}_G \, \dot{\boldsymbol{\omega}}_{b,d} \end{bmatrix}.$$

笔者进行复现过程中，发现此处角加速度的 PD 求解的 Kp 项也能用期望、实际角度之差。然后为了求解较理想的地面反作用力，可以将该问题转化为一个二次规划问题：

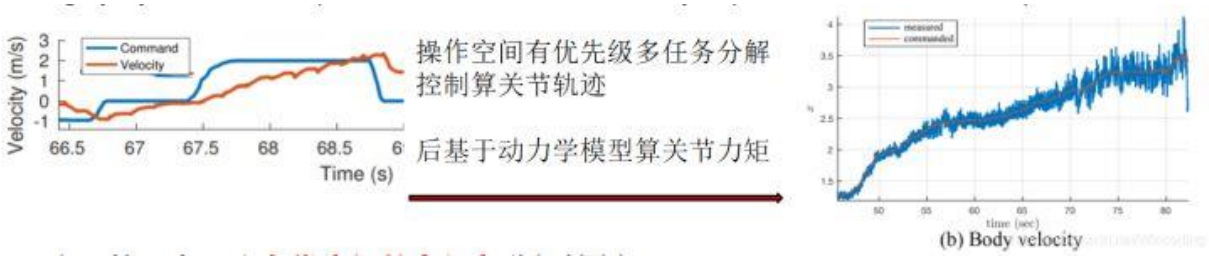
$$\boldsymbol{F}^* = \min_{\boldsymbol{F} \in \mathbb{R}^{12}} \; (\boldsymbol{A}\boldsymbol{F} - \boldsymbol{b}_d)^T \boldsymbol{S} (\boldsymbol{A}\boldsymbol{F} - \boldsymbol{b}_d)$$

其中 S 为正定矩阵，保证了该问题有惟一的全局最小值，此处可以用 Quadprog++求解，求

得地面反作用力后，则期望的腿末端力为其负值，利用雅可比计算关节力矩即可。

这个平衡控制器的模型是一个高度简化的单刚体模型，但在 MIT 实际测试中表现出了不错的效果，在 Cheetah 3 跳跃上桌子的控制方案中，这个模型用来做机器人的落地稳定控制器。

2019 年，MIT 在 Mini Cheetah 上采用了 MPC+WBC 的控制方案，论文《Highly Dynamic Quadruped Locomotion via whole-body impulse control and model predictive control》中称加入 WBC 后 Mini 的最高运动速度由 18 年只采用 MPC 时的 2.45m/s 增加到 19 年的 3.7m/s。



mini 上所用的 MPC 与 Cheetah 3 上的相同，笔者在上一节中也谈了下自己对 Cheetah 3 上的凸模型预测控制的理解。在 cheetah 3 中，MPC 优化出的地面反作用力取负号后左乘雅可比转置得到关节力矩，笔者的理解是此处的关节力矩的计算更多的是反映了静力学层面的关系，其间并没有考虑关节速度、加速度等对关节力矩的影响，即没有考虑动力学。而 mini 上的 WBC 恰好是在 MPC 计算得到地面反作用力的基础上利用动力学模型得到关节的位置、速度、加速度以及力矩这些量，实现更精准的控制。因为机器人是在不断运动的，因此 mini 所用的是机器人浮动基动力学模型，即机器人的基座是运动的，求解难度是大于基座固定不动的工业机器人的。浮动基动力学方程表达形式如下：

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{b}(\mathbf{q},\mathbf{u}) + \mathbf{g}(\mathbf{q}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_{ext}^T\mathbf{F}_{ext}$$

mini 中的 WBC 的核心思想是操作空间有优先级的多任务控制组织方案，关于优先级、多任务这点，举个简单的例子，比如一个机械臂，指定了期望的末端速度，同时我们要求在保证达到期望的末端速度前提下，机械臂的第一、二个关节速度要尽可能接近 0，这便是一个简单的有优先级的多任务控制问题。解决此类问题最常用的方法是零空间组织法（null space projection）。

关于 null space　projection,可以看一下 ETH 的精简机器人学讲义：https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2017/RD_HS2017script.pdf，笔者刚所举的例子在这份讲义的第 44 页有详细的讲解。

mini 的关节位置、速度、加速度求解其实也是用了相同的方法：

$$\Delta \mathbf{q}_i = \Delta \mathbf{q}_{i-1} + \boldsymbol{J}_{i|pre}{}^{\dagger} \left(\mathbf{e}_i - \boldsymbol{J}_i \Delta \mathbf{q}_{i-1} \right), \tag{16}$$

$$\dot{\mathbf{q}}_i^{\mathrm{cmd}} = \dot{\mathbf{q}}_{i-1}^{\mathrm{cmd}} + \boldsymbol{J}_{i|pre}{}^{\dagger} \left(\dot{\mathbf{x}}_i^{\mathrm{des}} - \boldsymbol{J}_i \dot{\mathbf{q}}_{i-1}^{\mathrm{cmd}} \right), \tag{17}$$

$$\ddot{\mathbf{q}}_i^{\mathrm{cmd}} = \ddot{\mathbf{q}}_{i-1}^{\mathrm{cmd}} + \overline{\boldsymbol{J}_{i|pre}^{\mathrm{dyn}}} \left(\ddot{\mathbf{x}}_i^{\mathrm{cmd}} - \dot{\boldsymbol{J}}_i \dot{\mathbf{q}} - \boldsymbol{J}_i \ddot{\mathbf{q}}_{i-1}^{\mathrm{cmd}} \right), \tag{18}$$

where

$$\begin{aligned} \boldsymbol{J}_{i|pre} &= \boldsymbol{J}_i \boldsymbol{N}_{i-1}, \\ \boldsymbol{N}_{i-1} &= \boldsymbol{N}_0 \boldsymbol{N}_{1|0} \cdots \boldsymbol{N}_{i-1|i-2}, \end{aligned} \tag{19}$$

$$\begin{aligned} \boldsymbol{N}_0 &= \boldsymbol{I} - \boldsymbol{J}_c^{\dagger} \boldsymbol{J}_c, \\ \boldsymbol{N}_{i|i-1} &= \boldsymbol{I} - \boldsymbol{J}_{i|i-1}{}^{\dagger} \boldsymbol{J}_{i|i-1}. \end{aligned} \tag{20}$$

Here, $i \geq 1$, and

$$\begin{aligned} \Delta \mathbf{q}_0, \ \dot{\mathbf{q}}_0^{\mathrm{cmd}} &= \mathbf{0}, \\ \ddot{\mathbf{q}}_0^{\mathrm{cmd}} &= \overline{\boldsymbol{J}_c^{\mathrm{dyn}}} (-\boldsymbol{J}_c \dot{\mathbf{q}}). \end{aligned} \tag{21}$$

$\mathbf{e}_i$ is the position error defined by $\mathbf{x}_i^{\mathrm{des}} - \mathbf{x}_i$ and $\ddot{\mathbf{x}}_i^{\mathrm{cmd}}$ is the acceleration command of i-th task defined by

$$\ddot{\mathbf{x}}_i^{\mathrm{cmd}} = \ddot{\mathbf{x}}^{\mathrm{des}} + \boldsymbol{K}_p \left(\mathbf{x}_i^{\mathrm{des}} - \mathbf{x}_i \right) + \boldsymbol{K}_d \left(\dot{\mathbf{x}}^{\mathrm{des}} - \dot{\mathbf{x}} \right), \tag{22}$$

这里位置、速度的任务雅可比在讲义的 43 页有阐述，加速度的任务雅可比在讲义的 73 页也有阐述（主要是基于接触约束导出的 null space projection）然后，便是将计算出的位置、速度、加速度带入动力学方程做一个优化，

$$\min_{\boldsymbol{\delta}_{\mathbf{f}_r},\boldsymbol{\delta}_f} \quad \boldsymbol{\delta}_{\mathbf{f}_r}^\top \boldsymbol{Q}_1 \boldsymbol{\delta}_{\mathbf{f}_r} + \boldsymbol{\delta}_f^\top \boldsymbol{Q}_2 \boldsymbol{\delta}_f \tag{25}$$

s.t.

$$\boldsymbol{S}_f \left(\boldsymbol{A}\ddot{\mathbf{q}} + \mathbf{b} + \mathbf{g}\right) = \boldsymbol{S}_f \boldsymbol{J}_c^\top \mathbf{f}_r \qquad \text{(floating base dyn.)}$$

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}^{\text{cmd}} + \begin{bmatrix} \boldsymbol{\delta}_f \\ \mathbf{0}_{n_j} \end{bmatrix} \qquad \text{(acceleration)}$$

$$\mathbf{f}_r = \mathbf{f}_r^{\text{MPC}} + \boldsymbol{\delta}_{\mathbf{f}_r} \qquad \text{(reaction forces)}$$

$$\boldsymbol{W}\mathbf{f}_r \geq \mathbf{0}, \qquad \text{(contact force constraints)}$$

关于关节位置的计算论文中是说：

$$\mathbf{q}_j^{\text{cmd}} = \mathbf{q}_j + \Delta\mathbf{q}_j .$$

其中 qj 是此刻测得的关节位置，在摆动腿的控制中，大多是从初始位置到末端位置进行插值，然后计算每个控制周期的期望关节位置、速度。我的理解是此处摆动腿便不能再用插值的方式进行控制，而是每个控制周期都会有基于 null space projection 的关节位置、速度指令，需要对其做 PD 控制律，然后再加上一个基于动力学模型的力矩前馈。

论文中优化问题的核心思想是在满足动力学方程约束的情况下，允许机器人基座与期望状态之间存在些许的偏差，这种放松更有利于机器人的高速运动，而后将优化出的地面反作用力、关节加速度以及之前计算】得到的关节位置、速度带到动力学方程里：

$$\begin{bmatrix} \boldsymbol{\tau}_f \\ \boldsymbol{\tau}_j \end{bmatrix} = \boldsymbol{A}\ddot{\mathbf{q}} + \mathbf{b} + \mathbf{g} - \boldsymbol{J}_c^\top \mathbf{f}_r$$

便可以求出支撑腿、摆动腿的力矩，(摆动腿没有接触力 fr 这一项) 求出的这个力矩会做为一个前馈，然后加上关节位置、速度的 PD 控制律，对机器人的支撑腿、摆动腿进行控制。按论文中所述，这种控制方案让 mini 跑的更快了。

机电设计

为实现快速奔跑并提高能量利用效率，MIT Cheetah 的设计者主要做了三方面的重要工作：

1）机械结构设计；

2）执行机构设计；

3）控制器设计。

个驱动器都分布在髋关节，对于膝关节的 pitch 则通过连杆或者滚珠丝杠传递下去，有益于减小腿部惯量并提升动态运动性能。

但此设计也存在一些弊端，即串联式的关节设计使得电机分布在旋转关节的两侧，因此两电机间的电气连接线在设计上将有所考究。

在当今主流的关节机器人的驱动器设计中，有中空式走线和非中空式走线两种方案，目前大多数机器人都会采用前者。因为作为后者的非中空式走线的线路会直接暴露在关节外侧，一方面影响机器人设计的整体性和紧凑性，另外一方面将带来因为勾拽而导致电气连接线脱落的安全隐患。而中空式走线将会避免这些问题（推测 Spotmini 也采用中空式走线的方式），但同时会在结构设计上带来更大的复杂性，并且需要扩大驱动器的径向和轴向尺寸，致使整体设计的紧凑度降低，躯体惯量增大，给后续动态运动能力带来负面影响。

同时，无论是非中空式走线还是当今主流的中空式走线，在相应关节需要旋转大量次数的工况下，都会带来相应的线材磨损和疲劳损伤，最终影响作为商品的可靠性能。在机器人的科研应用中，几乎已经认为中空式走线的方案是“完美”的，因为科研类机器人几乎不考虑疲劳寿命，但作为商业产品性质的 AlienGo，想追求更加极致的产品可靠性，是其硬件设计优化上永远的目标。

自主导航

点云数据处理方法概述

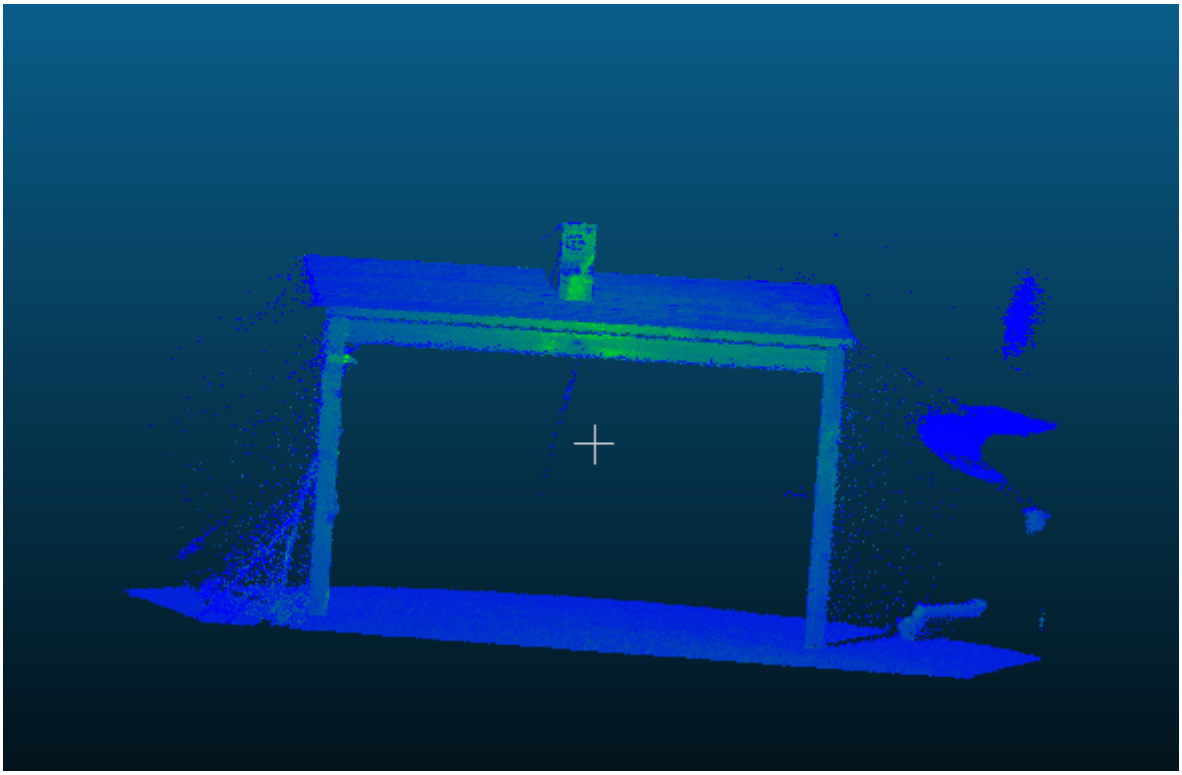
ICP 点云配准就是我们非常熟悉的点云处理算法之一。实际上点云数据在形状检测和分类、立体视觉、运动恢复结构、多视图重建中都有广泛的使用。点云的存储、压缩、渲染等问题也是研究的热点。随着点云采集设备的普及、双目立体视觉技术、VR 和 AR 的发展，点云数据处理技术正成为最有前景的技术之一。PCL 是三维点云数据处理领域必备的工具和基本技能，这篇文章也将粗略介绍。

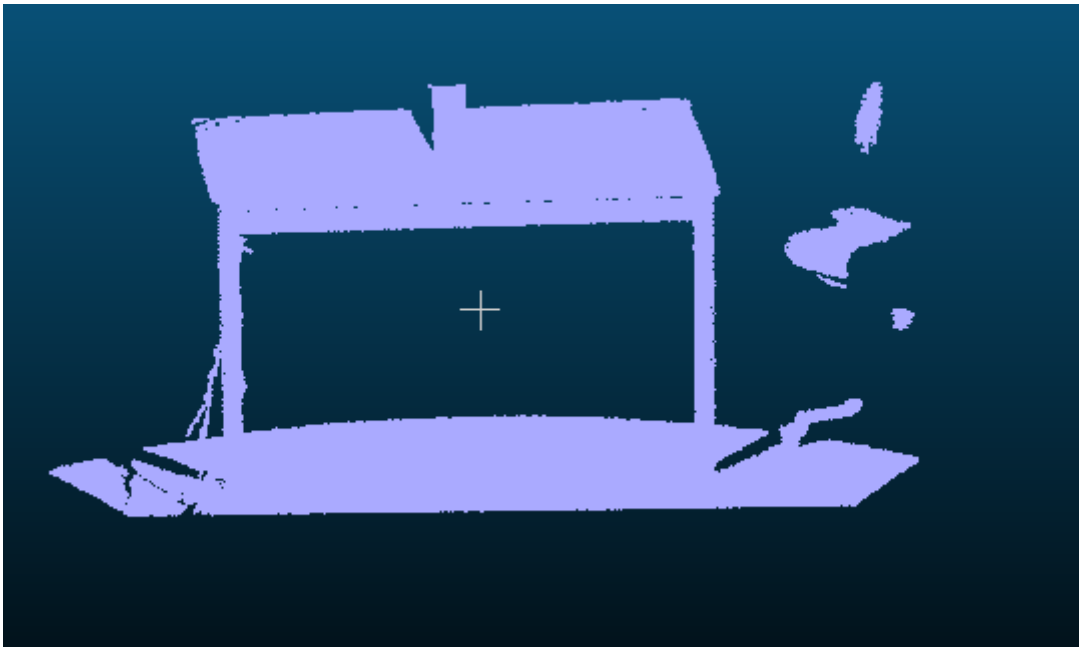
三维点云数据处理技术

1. 点云滤波（数据预处理）

点云滤波，顾名思义，就是滤掉噪声。原始采集的点云数据往往包含大量散列点、孤立点，比如下图为滤波前后的点云效果对比。

点云滤波的主要方法有：双边滤波、高斯滤波、条件滤波、直通滤波、随机采样一致滤波、VoxelGrid 滤波等，这些算法都被封装在了 PCL 点云库中。





2. 点云关键点

我们都知道在二维图像上，有 Harris、SIFT、SURF、KAZE 这样的关键点提取算法，这种特征点的思想可以推广到三维空间。从技术上来说，关键点的数量相比于原始点云或图像的数据量减小很多，与局部特征描述子结合在一起，组成关键点描述子常用来形成原始数据的表示，而且不失代表性和描述性，从而加快了后续的识别，追踪等对数据的处理了速度，故而，关键点技术成为在 2D 和 3D 信息处理中非常关键的技术。

常见的三维点云关键点提取算法有一下几种：ISS3D、Harris3D、NARF、SIFT3D

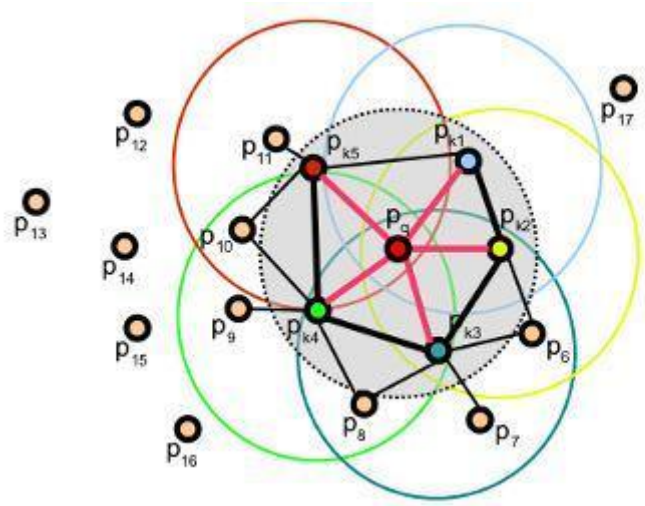
这些算法在 PCL 库中都有实现，其中 NARF 算法是博主见过用的比较多的。

3. 特征和特征描述

如果要对一个三维点云进行描述，光有点云的位置是不够的，常常需要计算一些额外的参数，比如法线方向、曲率、文理特征等等。如同图像的特征一样，我们需要使用类似的方式来描述三维点云的特征。

常用的特征描述算法有：法线和曲率计算、特征值分析、PFH、FPFH、3D Shape Context、Spin Image 等。

PFH：点特征直方图描述子，FPFH：跨苏点特征直方图描述子，FPFH 是 PFH 的简化形式。这里不提供具体描述了，具体细节去谷歌吧。



4. 点云配准

点云配准的概念也可以类比于二维图像中的配准，只不过二维图像配准获取得到的是 x，y，alpha，beta 等放射变化参数，二三维点云配准可以模拟三维点云的移动和对其，也就是会获得一个旋转矩阵和一个平移向量，通常表达为一个 4×3 的矩阵，其中 3×3 是旋转矩阵，1*3 是平移向量。严格说来是 6 个参数，因为旋转矩阵也可以通过罗格里德斯变换转变成 1*3 的旋转向量。

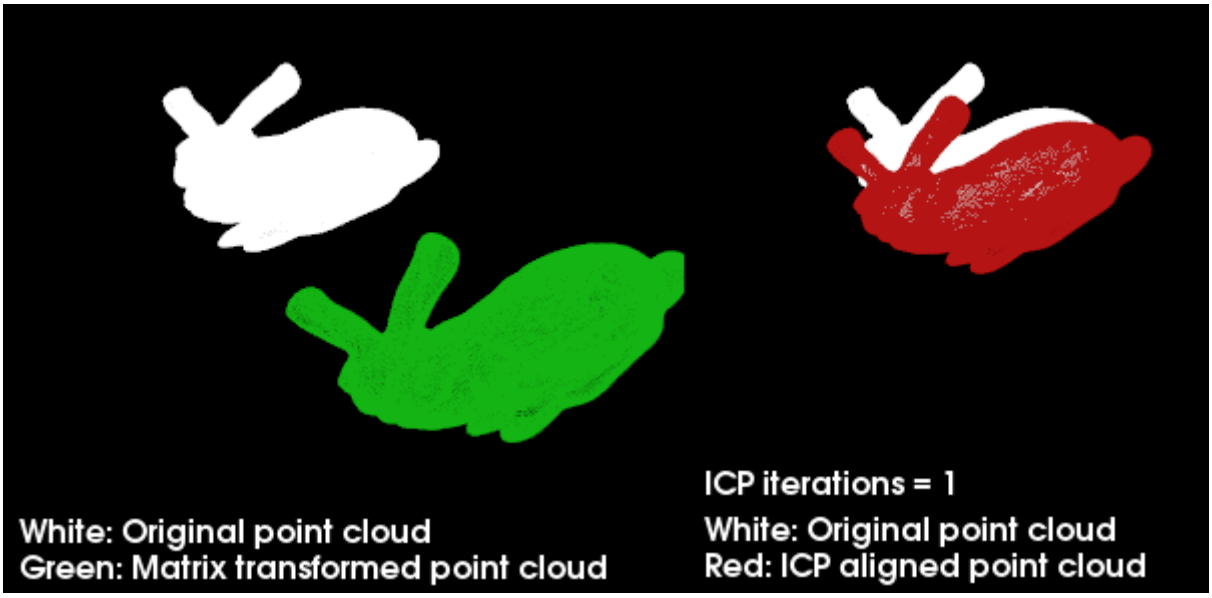
常用的点云配准算法有两种：正太分布变换和著名的 ICP 点云配准，此外还有许多其它算法，列举如下：

ICP：稳健 ICP、point to plane ICP、point to line ICP、MBICP、GICP

NDT 3D、Multil-Layer NDT

FPCS、KFPSC、SAC-IA

Line Segment Matching、ICL



5. 点云分割与分类

点云的分割与分类也算是一个大 Topic 了，这里因为多了一维就和二维图像比多了许多问题，点云分割又分为区域提取、线面提取、语义分割与聚类等。同样是分割问题，点云分割涉及面太广，确实是三言两语说不清楚的。只有从字面意思去理解了，遇到具体问题再具体归类。一般说来，点云

分割是目标识别的基础。

分割：区域声场、Ransac 线面提取、NDT-RANSAC、K-Means、Normalize Cut、3D Hough Transform(线面提取)、连通分析

分类：基于点的分类，基于分割的分类，监督分类与非监督分类

6. SLAM 图优化

SLAM 又是大 Topic，SLAM 技术中，在图像前端主要获取点云数据，而在后端优化主要就是依靠图优化工具。而 SLAM 技术近年来的发展也已经改变了这种技术策略。在过去的经典策略中，为了求解 LandMark 和 Location，将它转化为一个稀疏图的优化，常常使用 g2o 工具来进行图优化。下面是一些常用的工具和方法。

g2o、LUM、ELCH、Toro、SPA

SLAM 方法：ICP、MBICP、IDC、likehood Field、 Cross Correlation、NDT

7. 目标识别检索

这是点云数据处理中一个偏应用层面的问题，简单说来就是 Hausdorff 距离常被用来进行深度图的目标识别和检索，现在很多三维人脸识别都是用这种技术来做的。

8. 变化检测

当无序点云在连续变化中，八叉树算法常常被用于检测变化，这种算法需要和关键点提取技术结合起来，八叉树算法也算是经典中的经典了。

9. 三维重建

我们获取到的点云数据都是一个个孤立的点，如何从一个个孤立的点得到整个曲面呢，这就是三维重建的 topic。

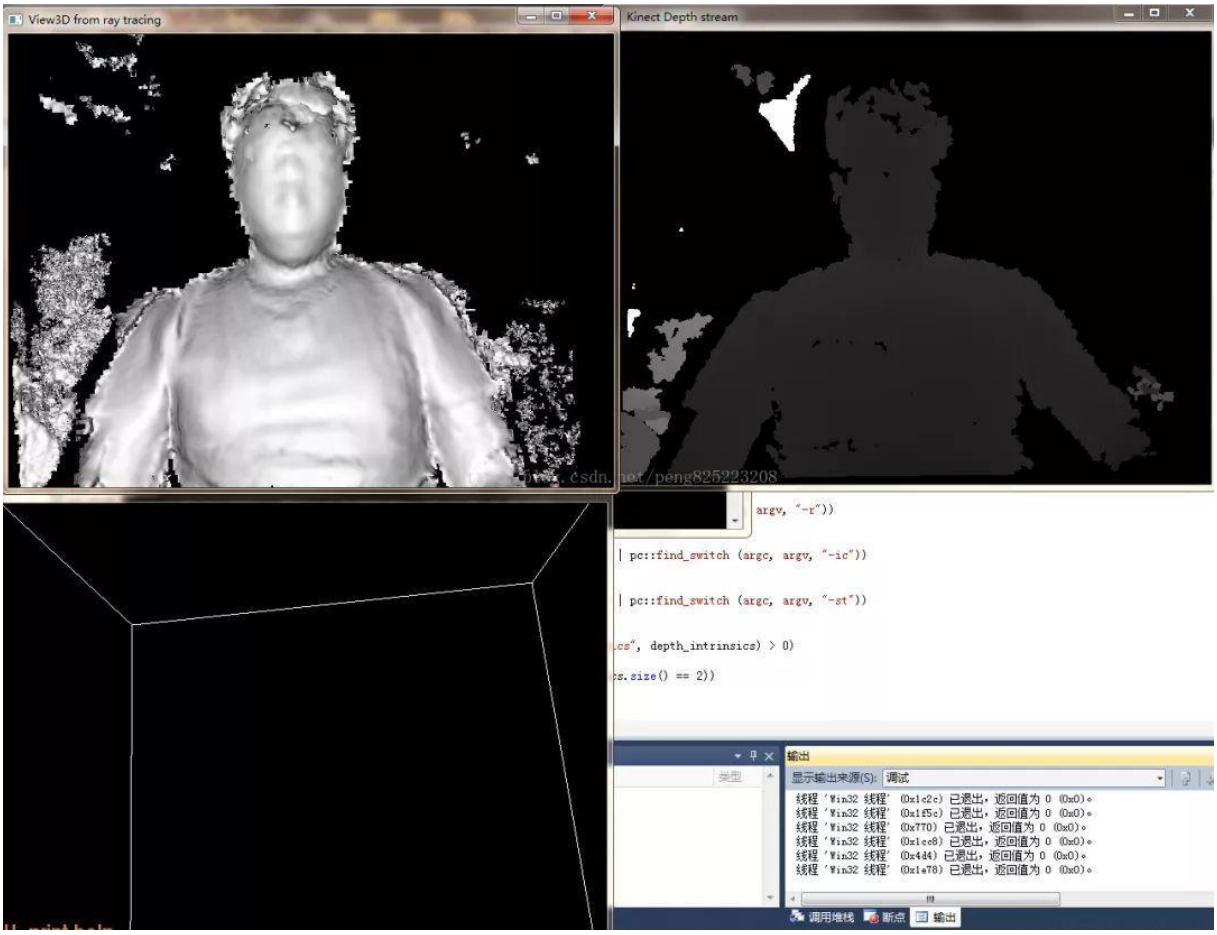
在玩 kinectFusion 时候，如果我们不懂，会发现曲面渐渐变平缓，这就是重建算法不断迭代的效果。我们采集到的点云是充满噪声和孤立点的，三维重建算法为了重构出曲面，常常要应对这种噪声，获得看上去很舒服的曲面。

常用的三维重建算法和技术有：

泊松重建、Delauary triangulatoins

表面重建，人体重建，建筑物重建，输入重建

实时重建：重建纸杯或者龙作物 4D 生长台式，人体姿势识别，表情识别



10. 点云数据管理

点云压缩，点云索引（KDtree、Octree），点云 LOD（金字塔），海量点云的渲染

PCL 库简介

点云数据处理中，不仅涉及前段数据的输入，中间数据和处理，还涉及到后端点云的渲染显示，如果这些函数都要我们亲自来实现，那么开发效率必然受到极大影响。在点云数据处理领域，有一个不可或缺的助手：PCL (Point Cloud Library)。PCL 在点云数据处理中的地位犹如 OpenCV 在图像处理领域的地位，如果你接触三维点云数据处理，那么 PCL 将大大简化你的开发。

位姿估计

ICP：例如 $X = \{ x_1 , x_2 , x_3 , \ldots , x_n \}$ X=\{x_{1},x_{2},x_{3},...,x_{n}\} X={x1,x2,x3,...,xn}

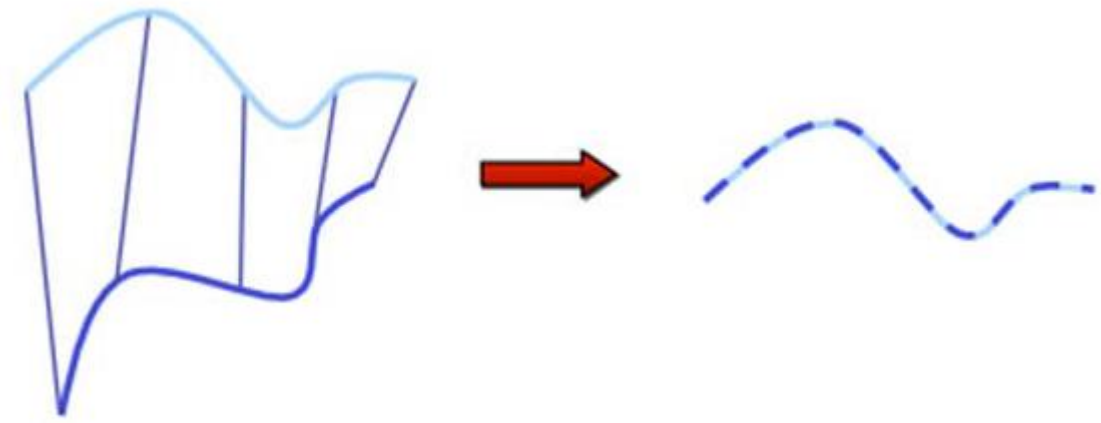
是我雷达传感器得到的 n 个点，而 $P = \{ p_1 , p_2 , \ldots , p_n \}$ P=\{p_{1},p_{2}, ..., p_{n}\}

P={p1,p2,...,pn}是地图上的 n 个点，如何将他们进行对应求解？

$E (R , t) = \Sigma_{i = 1}^{n} \| x_i - R p_i - t \|_2$ E(R,t)=\Sigma_{i=1}^{n}{|x_{i}-Rp_{i}-t|^{2}}

E(R,t)=Σi=1n|xi−Rpi−t|2

用上面这个公式进行优化（其实是一个优化问题），但它有一个闭式解，参考《SLAM 十四讲》7.9 的 ICP。



雷达 SLAM

1.初始化：雷达第一次的 scan；

2.位姿跟踪：ICP；(个人觉得并不全是 ICP，例如 LOAM 中就不是这样的)

3.地图优化：占据栅格地图；

Cartographer 专题讲解一】萌妹子带你读 Cartoprapher 论文

https://www.bilibili.com/video/av412871551/

视觉 SLAM

1.初始化：（求解本质矩阵 E E E,三角化等）；

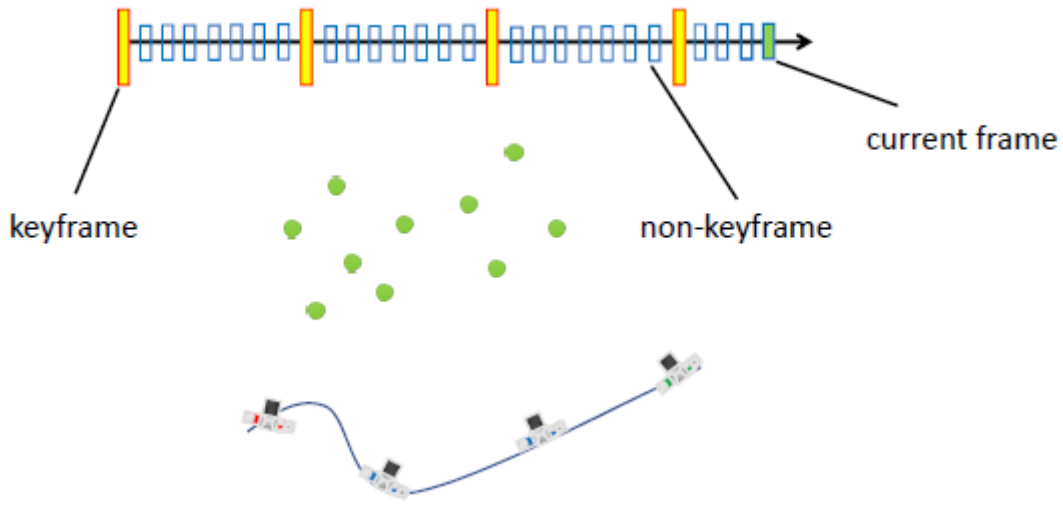
2.位姿跟踪：1）特征点跟踪；2）位姿的 BA；

3.地图优化：1）三角化，BA；2）闭环，位姿图；

Visual SLAM

实时性要求：只对视频中的某几个 keyframe 进行优化，而不是所有帧都使用。

Local BA 的概念：相邻的若干帧拿进来做 BA，而不是所有的帧拿进来做 BA。



Re-localization

- Motion blurs

运动太快或者环境还暗，tracking 会丢失。

- Moving objects

走到一群人（一个食堂人很多），会丢失

- Large occlusion

大范围的遮挡，也会 loss tracking

- Sudden fast motion

突然的急速运动，剧烈抖动，初始化不好

- Sudden illumination change

隧道中出来，相机过曝

在这些问题中，相机如何重定位？re-localization。

Localization

先将地图建立好，然后再进行定位。

优点在于，这个 mapping 的结果得到的地图的结果会很好（因为你可以花很多时间去调整），还可以花很多时间去 mapping（因为没有实时性要求了）。

drift 和 loop closure

首先要将 loop 检测出来。（image search）

检测出来之后进行图优化（global SfM）。

基于 AI 的导航

视觉算法

采用 Intel 的 Movidus 算法库。

待续。

供应链与质量管控

供应链与质量的管控应该是从研发样机到量产产品中最为重要的环节。这里包括供应链供货产品质量一致性管控、装配过程一致性管控及机器人整机出厂性能一致性管控等等。而这几类的管控需要大量成熟、细致的技术文案作为基础，需要大量产业技术工人的严格执行作为支撑，更需要上游零部件供应商的鼎立相助作为配合，因此就不仅仅是要求研发人员能够对相关技术”吃得通透“而产出详细完整的技术文档，而是对公司整体的运营、管理与执行能力的综合考验。

入门资料

A. 力学：

修理论力学课，或者自学推荐中科大杨维纮（hong）教授的《力学》

B. 具体类型机器人动力学：

B1 机械臂类，就是连杆多刚体机构，包括足式

B1.1 最简单的：机器人运动/动力学，可以看斯坦福 JJCraig 教授的《Introduction to Robotics》前面几章

B1.2 比较完整的：多刚体动力学，上一条是它的子集。推荐书籍 Featherstone R.《Rigid body dynamics algorithms》

B1.3 最完整的：Screw Theory，将力学各种量用群论描述分析，逻辑性非常强，体系完整。上一条是它的子集。但是比较难，除非你对数学很有爱，否则不用很早学它

B2 无人机类，请移步有哪些值得一读的无人机 / 四旋翼方面的论文？ - Liu Top 的回答

C. 控制理论

C1 传统控制理论 虽然叫传统，但是绝对不陈旧，实用性和反馈的思想都让它很值得认真学。推荐书籍《Feedback Control of Dynamic Systems》Franklin G

C2 现代控制理论 基于状态空间理论建立的学科，这个思想非常重要，要深入学。建议选课

C3 控制理论的其他分支 有很多，找对你有用的学。个人觉得最有用的是这些：

C3.1 最优控制 建议在学过凸优化后再来学它，理解会容易许多。推荐看一个叫 Emo Todorov 的教授写的东西，此人搞机器人但数学功底极其扎实，写的教程很清楚透彻。

C3.2 非线性控制 要求不高的话会反馈线性化就够了，推荐书籍 李春文《多变量非线性控制的逆系统方法》

C3.3 鲁棒控制 推荐书籍：周克敏《ESSENTIALS OF ROBUST CONTROL》有中文版貌

似，周克敏教授是鲁棒控制领域的大佬

D. 优化

D.1 凸优化 这是最基本又最重要的。推荐书籍：《Convex Optimization》Stephen Boyd，斯坦福的教材

D.2 轨迹优化 最优控制和增强学习都是在讲轨迹优化。

E. 机器学习

E.1 增强学习 其实是一种离散时间最优控制，这是机器学习学科离控制最近的。推荐书籍：《Reinforcement Learning: An Introduction》，R.S.Sutton，MIT 教授。