Configuration Space（上）

文档作者：李拥祺

注： 本文大部分内容来源于这本书：

> Principles of Robot Motion: Theory, Algorithms, and Implementations[M]. MIT Press, 2005.

由于内容较多，我将笔记分为上下两个部分

---

0. 概述

- 构型空间（Configuration Space, C-Space）是运动规划的理论基础之一；
- 不同的机器人由于自身状态（形状、运动能力等）的不同，同时工作环境也大都不同，如果针对这些机器人各自做运动规划将是一件非常 “累”的事情，人类有一个特别强大的能力就是**抽象**， `C-Space` 就是为了从各种各样的实际机器人中抽离出来，在其中设计的算法可以方便扩展到不同类型的机器人上；
- 如果要给一个机器人做运动规划，首先必须知道机器人自身的位置，更具体点，必须知道机器人自身每一个点的位置，不然就存在撞毁的可能性；
- 思考以下几个问题：
  - 需要多少信息才能描述机器人自身每一个点的位置？
  - 如何去表示这些信息？
  - 这些信息包含了哪些数学性质？
  - 环境中的障碍物如何考虑？

---

1. 什么是机器人构型？

1.1 一些基本概念

- 机器人系统的**构型** (用符号 q 表示)就是对机器人自身每一个点的位置的完整描述。
- 构型空间（C-Space，用符号 $\mathcal{Q}$ 表示）就是该机器人系统所有可能构型构成的集合，每一个构型就相当于`C-Space`中一个点。
- 机器人系统的自由度（DOF）就是构型空间的维度，即用来描述一个构型所需要的最小参数个数。
- **工作空间** （WorkSpace）：
  - 一般指机器人所处的二维或三维欧几里得环境空间；
  - 针对一些特定机器人时有更加具体的描述，如机械臂的工作空间指的是其末端执行器（end effector）所能够到达的工作点的集合。

1.2 一个例子：平面二连杆

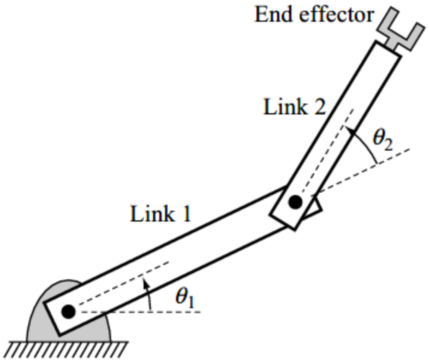


Figure 3.1 The angles θ_1 and θ_2 specify the configuration of the two-joint robot.

- 平面二连杆的自由度是2，我们可以用 `Link1` 的旋转角度 θ_1 和 `Link2` 的旋转角度 θ_2 来确定其构型，简言之就是这两个参数确定了之后，二连杆上所有点的位置也随之确定；

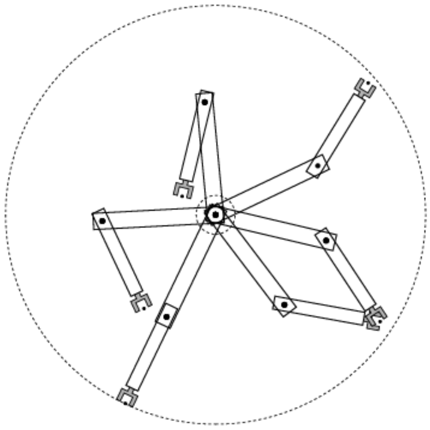


Figure 3.3 The workspace for this two-joint manipulator is an annulus, a disk with a smaller disk removed from it. Note that all points in the interior of the annulus are reachable with a right-arm configuration and a left-arm configuration.

- 平面二连杆的工作空间是其末端执行器所能到达的区域，如上图所示是一个圆环。**从图中可以看出，在工作空间中的点，二连杆都有两种方式来到达，换言之就是工作空间中的点对应两组参数** (θ_1, θ_2)，所以只用末端执行器所在的位置点来描述其构型是不行的（无法确定二连杆上所有点的位置），从而这个圆环也不是其构型空间。

Lyq

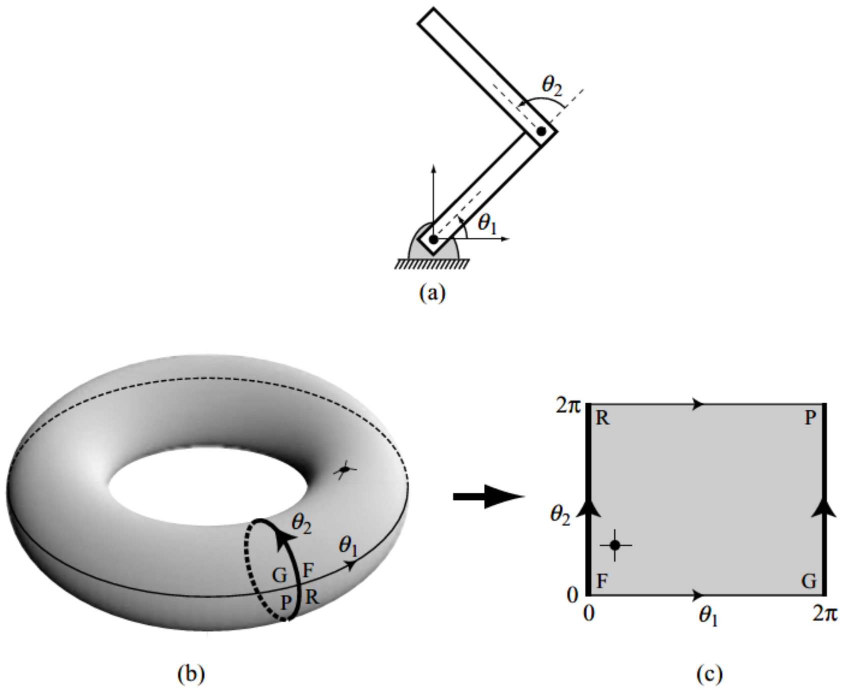


Figure 3.2 (a) A two-joint manipulator. (b) The configuration of the robot is represented as a point on the toral configuration space. (c) The torus can be cut and flattened onto the plane. This planar representation has “wraparound” features where the edge FR is connected to GP, etc.

- 两个`link`的旋转角度θ_i，（$i=1,2$）的取值分别都对应**单位圆** S^1 上的点，这个是比较好理解的，因为取值范围都是 $[0, 2\pi)$；
- 由前面可知这两个旋转角度构成了这个二连杆的`C-Space`（二维圆环），即 $T^2 = S^1 \times S^1$，如果可视化的话，就是一个甜甜圈外壳的样子；这个是比较好理解的，想象一下 θ_1 取某一个值的时候，θ_2 任然可以在 $[0, 2\pi)$ 内取任意值，是不是就如上图中(b)所示。
- 区分两个概念：`embedding dimension` 和 `intrinsic dimension`
  - 在上图中，二维圆环的`embedding dimension`是3，但是其`intrinsic dimension` 是2；
  - 直观理解就是这个二维圆环是嵌入在三维空间中的，但是通过两个参数就可以完全描述这个二维圆环上的任意点；
  - 同理，$S^1 \in \mathbb{R}^2$，但是其只对应一个参数；
- **思考：** S^1 是连通的，但是 $[0, 2\pi)$ 却存在不连通，为什么?

---

2. 障碍物与构型空间

2.1 解释

- 根据前面对机器人构型的理解，我们可以将路径规划问题定义为**寻找一个连续函数（映射）** c

  $c : [0,1] \rightarrow \mathcal{Q} \;\; where \;\; c(0) = q_{start}, c(1) = q_{goal} \;\; and \;\; c(s) \in \mathcal{Q}_{free}, \forall s \in [0,1]$

  在这条路径上不存在一种构型使得机器人与障碍物发生碰撞。
- 我们在构型空间中定义机器人 $R(q)$表示机器人构型集合与障碍物 $\mathcal{WO}_i$的交集为**构型空间障碍物** $\mathcal{QO}_i$

  $\mathcal{QO}_i = \{q \in \mathcal{Q} | R(q) \bigcap \mathcal{WO}_i \neq \emptyset\}$
- 定义机器人那些没有与障碍物发生碰撞的构型组成的集合为**自由构型空间（自由空间）** $\mathcal{Q}_{free}$

  $\mathcal{Q}_{free} = \mathcal{Q} \setminus (\bigcup_i \mathcal{QO}_i)$，其中 $\setminus$ 表示补集（差集），$A \setminus B$表示属于集合A但是与集合B没有相交
- 定义一条**自由路径**：$c : [0,1] \rightarrow \mathcal{Q}_{free}$；

Lyq

- 路径：$c:[0,1]\rightarrow cl(\mathcal{Q}_{free})$ 被称为**半自由路径**，其中 $cl(\mathcal{Q}_{free})$ 表示 $\mathcal{Q}_{free}$ 的闭包。
- 自由路径与半自由路径的**区别**在于机器人是否被允许与障碍物的边界接触。
- 书中一般假设自由空间 $\mathcal{Q}_{free}$ 是开集合，简单理解 $cl(\mathcal{Q}_{free})$ 就是与之相对应的闭集合，即在开集合的基础上加上边界点。

2.2 例子

2.2.1 圆形移动机器人

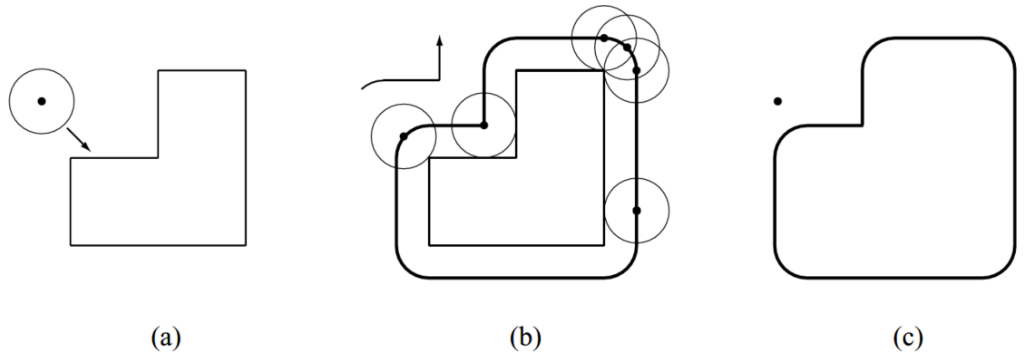


Figure 3.4 (a) The circular mobile robot approaches the workspace obstacle. (b) By sliding the mobile robot around the obstacle and keeping track of the curve traced out by the reference point, we construct the configuration space obstacle. (c) Motion planning for the robot in the workspace representation in (a) has been transformed into motion planning for a point robot in the configuration space.

- 图 (a) 表示一个环境中存在一个圆形移动机器人与一个多边形障碍物；
- 图 (b) 表示如何将工作空间中的障碍物转换为构型空间中的障碍物：
  - 在圆形移动机器人上选取一个参考点，图中是选择圆心；
  - 让圆形移动机器人绕障碍物一周，将参考点构成的曲线作为构型空间障碍物；
- 图 (c) 表示与图 (a) 中所示的工作空间**等价**的构型空间。

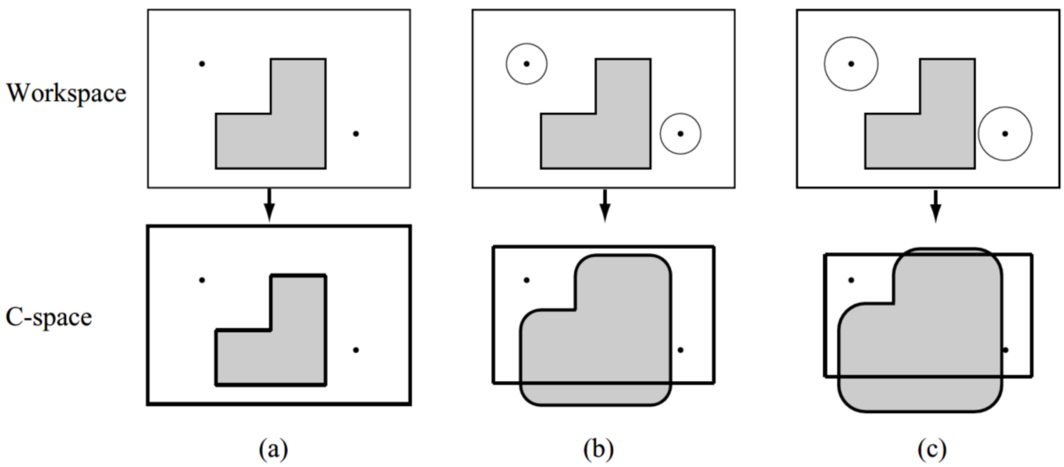
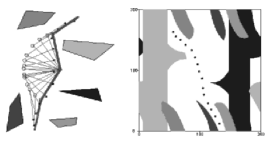
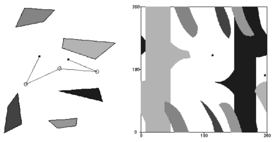
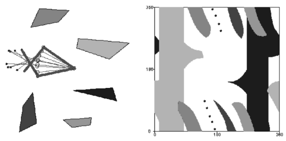


Figure 3.5 The top row shows the workspace and the bottom row shows the configuration space for (a) a point mobile robot, (b) a circular mobile robot, and (c) a larger circular mobile robot.

- 上图表示三个不同半径的圆形移动机器人在同一个环境下尝试去找到一条路径从起始点到终止点；
- 图 (a)、(b)和 (c) 表达了圆形移动机器人的半径对转换的构型空间的影响：
  - 没有半径的圆形移动机器人的工作空间和构型空间一样；

- 半径过大的圆形移动机器人的自由构型空间是被断开的，而且起始构型与目标构型刚好分别处于被断开的两个部分中，所以对于图 (c) 中的这种情况，不存在可行的路径。

2.2.2 平面二连杆

情形1	情形2	情形3
 **Figure 3.6** (Left) A path for the two-joint manipulator through its workspace, where the start and goal configurations are darkened. (Right) The path in the configuration space.	 **Figure 3.7** (Left) The workspace for a two-joint manipulator where the start and goal configurations are shown. (Right) The configuration space shows there is no free path between the start and goal configurations.	 **Figure 3.8** (Left) The workspace for a two-joint manipulator where the start and goal configurations are darkened. (Right) The path shows the wraparound of the planar representation of the configuration space.

- 上面这三张图分别表示了平面二连杆的三种不同情形下的构型空间**网格图**：
  - 情形1 中起始构型与目标构型之间是可以达到的；
  - 情形2 中起始构型与目标构型之间是不可达的；
  - 情形3 中比较特殊，出现了**环绕**，可以简单理解为平面二连杆的两个旋转角度超出了 2π
- 这里需要注意的是，我们在上图中看到的都是**网格**构型空间，这个**网格构型空间**并不能完全表示实际 `C-Space`，因此我们并不能完全根据上图得到可行的路径。
- 为了解决这个问题，我们可以在每一个网格点**膨胀**机器人，检测其是否是安全的（即不碰到障碍物），如果在这个点是安全的，那么到下一个点之间的路径也是安全的，这种方法过于**保守**，可能会导致找不到合适的路径。
- 我们也可以通过提高网格的**分辨率**来解决这个问题。

---

3. 构型空间的维度

3.1 理解

- 假设有一个机器人，它是平面上可以任意移动的一个点，我们可以在平面内固定一个坐标系，那么这个机器人的构型可以表示为：

  $q=(x,y)\in\mathcal{Q}=\mathbb{R}^2$，我们可以认为这个机器人具有**两个自由度**，它的`C-Space`是二维的。
- 假设有一个系统，它是由三个可以自由移动的**点状**机器人构成，描述这个系统的构型就是描述这三个点状机器人，从上一条我们可以知道需要六个坐标来描述，即 x_A,y_A,x_B,y_B,x_C,y_C，我们可以认为这个**系统**具有六个自由度，它的`C-Space` $\mathcal{Q}=\mathbb{R}^6$。
- 实际生活中的机器人常常被建模成一组由关节（`Joint`）连接的刚体，或者直接是一个单一刚体，如大多数移动机器人，而不是一些可以各自独立运动的点。
- 假设**平面上**一个可以平移和旋转的刚体机器人上固定了三个不同的点 A,B,C：
  - A 点位置 (x_A,y_A)；
  - 由于机器人是一个刚体，B 点的位置满足约束：$d(A,B)=\sqrt{(x_A-x_B)^2+(y_A-y_B)^2}$，所以 B 点的位置只能是以 A 点为圆心，$d(A,B)$ 为半径的一个圆上，它具有的自由度只是一个从 A 到 B 的角度值 θ；
  - C 点需要满足两个约束：$d(A,C)$和 $d(B,C)$，即 C 点位置已经确定了。
  - 刚体上每一个点的位置都可以通过 (x_A,x_B,θ) 来确定，即这个机器人的构型可以通过这三个量来确定，因此这个机器人的自由度为 3，它的构型空间为 $\mathbb{R}^2\times S^1$。
- 一个机器人的自由度如果远超任务需要，就称为**超冗余**（hyper-redundant）。

3.2 完整约束与非完整约束

- 这一部分可以去参考资料 2；
- 完整约束（holonomic constraint）
  - 满足形式 $g(q,t)$ 的约束，或者 $g(q,\dot{q},t)$ 通过积分可以化为 $g(q,t)$ 形式的约束。（q 指的是构型）
  - 每一个独立的完整约束都会**减少**这个系统在 `C-Space` 上的一个自由度，即如果一个系统可以用 n 个坐标来描述，同时具有独立的 m 个完整约束，那么这个系统的 `C-Space` 的维度是 $n-m$。
- 非完整约束（nonholonomic constraint）
  - 满足形式 $g(q,\dot{q},t)$ 的约束，且不可通过积分化为 $g(q,t)$ 形式。
  - **非完整约束并不会减少构型空间的自由度。**
  - 这一部分的具体讨论在书的第十二章，这里就不展开讨论了，先有一个概念就行了。
- 通俗理解就是，非完整约束只能约束你不能**直线**（理解为直接的意思）达到某些构型，却不能约束你**曲线**（理解为不那么直接）到达这些构型，而完整约束就是直接宣告不可能到达这些构型。
- 判断一个约束可积不可积用到的数学工具是 `Lie bracket of vector fields` 和 `Frobenius theorem (differential topology)`，这里先不展开讨论，等碰到或者需要用到的时候再说。

3.3 一些例子

3.3.1 空间中的刚体

- 计算自由度的过程：
  - 在刚体上选择三个不共线的点 A,B,C；
  - 首先确定好点 A 的位置 (x_A,y_A,z_A)，然后根据点 A 到点 B 的距离约束，确定点 B 位于以点 A 为中心的球体表面，最后根据点 C 到点 A,B 的距离约束，确定点 C 位于一个平面圆上，因此这个刚体的自由度为 $9-3=6$，即可以用 9 个坐标来描述，但是有三个完整约束。
  - 这是符合我们的认知的，确定点 A 需要三个自由度，确定球心已定的球面上一点只需要两个自由度（联想一下经纬度），确定圆心已定的圆上一点只需要一个自由度。
- 如果我们给这个空间刚体添加三个完整约束如 $z_A=z_B=z_C=0$，那么其自由度就变为了 3。

3.3.2 平面二连杆

- 我们以坐标减去约束的方法来计算一下平面二连杆的自由度：
  - 两个 `Link` 分别可以看成是空间刚体，则有 $6\times 2=12$ 个自由度；
  - 每个 `Link` 添加 3 个约束使得回到平面；
  - 给 `Joint1` 添加 2 个约束使得固定；
  - 一旦 `Link1` 的旋转角度确定，就相当于给 `Joint2` 添加了 2 个约束；
  - 平面二连杆的自由度为：$12-(3*2+2+2)=2$

3.3.3 开链关节机器人（串联机构）

- 将每一个关节的自由度相加得到整个机器人的自由度；
- 几种不同 `Joint` 的自由度：
  - `revolute` 关节自由度为 1，一般记为 `R`；
  - `prismatic` 关节自由度为 2，一般记为 `P`；
  - `spherical` 关节自由度为 3，也被称为 `ball-and-socket`；
- 我们前面提到的平面二连杆也被称为 RR 或 $2R$ 机器人，表示其两个关节都是旋转关节。

3.3.4 闭链机器人（并联机构）

- 假设有一个并联机构有 k 个 `Link`，如何计算其自由度：
  - 将其中一个固定在地面，则还剩 $k-1$ 个 `Link` 可以移动，设每一个可以移动的连杆的自由度为 N （空间机构取6，平面机构取3）；
  - 每一个 `Joint` 都具有 $N-f_i$ 个约束，其中 f_i 为关节自由度；
  - 该机构的自由度 $M=N(k-1)-\sum_{i=1}^{n}(N-f_i)=N(k-n-1)+\sum_{i=1}^{n}f_i$
- 例子

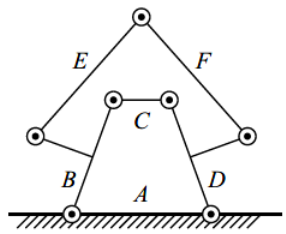
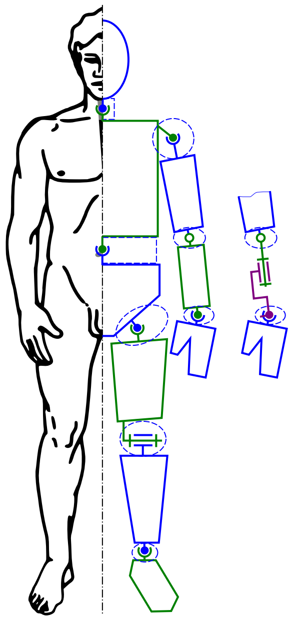


Figure 3.9 A planar mechanism with six links (A through F), seven revolute joints, and one degree of freedom.

该机构的自由度为：$3*(6-7-1)+7*1=1$

3.3.5 手臂



手臂的自由度计算：$6*(4-3-1)+(3+3+1)=7$

参考资料

1. https://www.zhihu.com/question/60108896 （构型空间的知乎回答）
2. https://zhuanlan.zhihu.com/p/48662038 （完整约束与非完整约束的知乎回答）
3. https://zh.wikipedia.org/wiki/%E6%B5%81%E5%BD%A2 （维基百科-流形）
4.