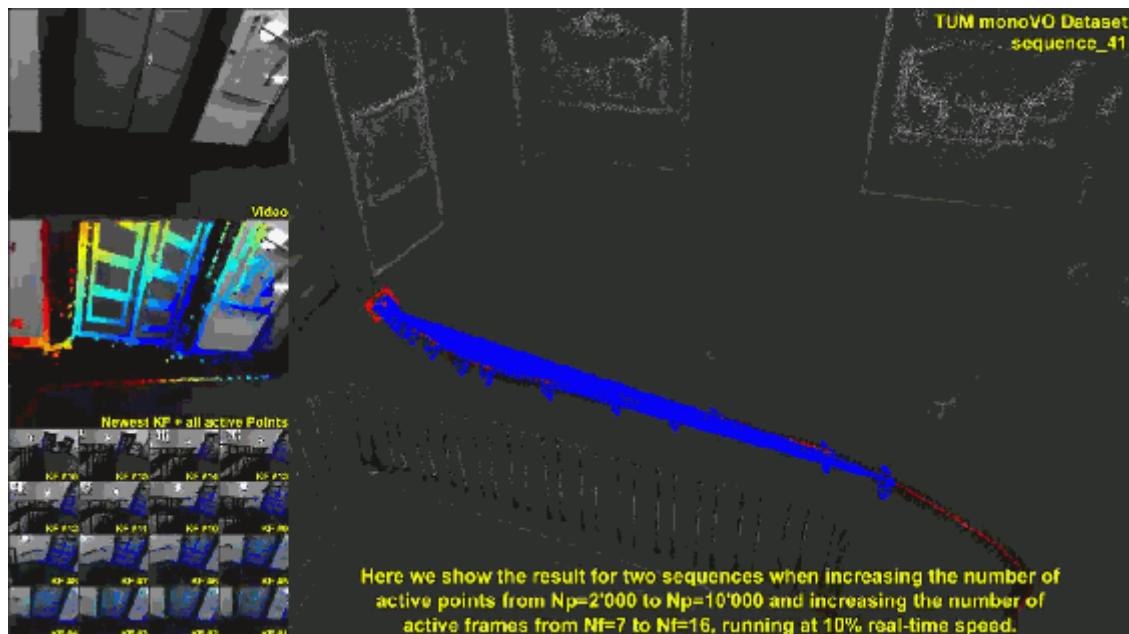


graph-SLAM 算法原理

slam算法是一个基于几何的视觉算法，相对于基于学习的深度学习方法，有其在空间上面感知和定位的优势

这里，我们以机器人的运动定位问题为切入点，讲解garph-slam算法所解决的问题、如何从理想情况一步一步深入，继而解决现实中的综合问题

1. 机器人运动定位问题



现实情景：

- 机器人面对的环境是完全未知的
- 机器人的初始位置已知
- 机器人搭配内部传感器，可以判断机器人的运动距离和方向
- 机器人搭配外部传感器，可以判断环境中的地标位置（地标数量和具体位置全部未知）

那么我们如何使得机器人在未知环境的运动过程中，通过不断感知地标位置来判断自己的具体位置和绘制地图呢？

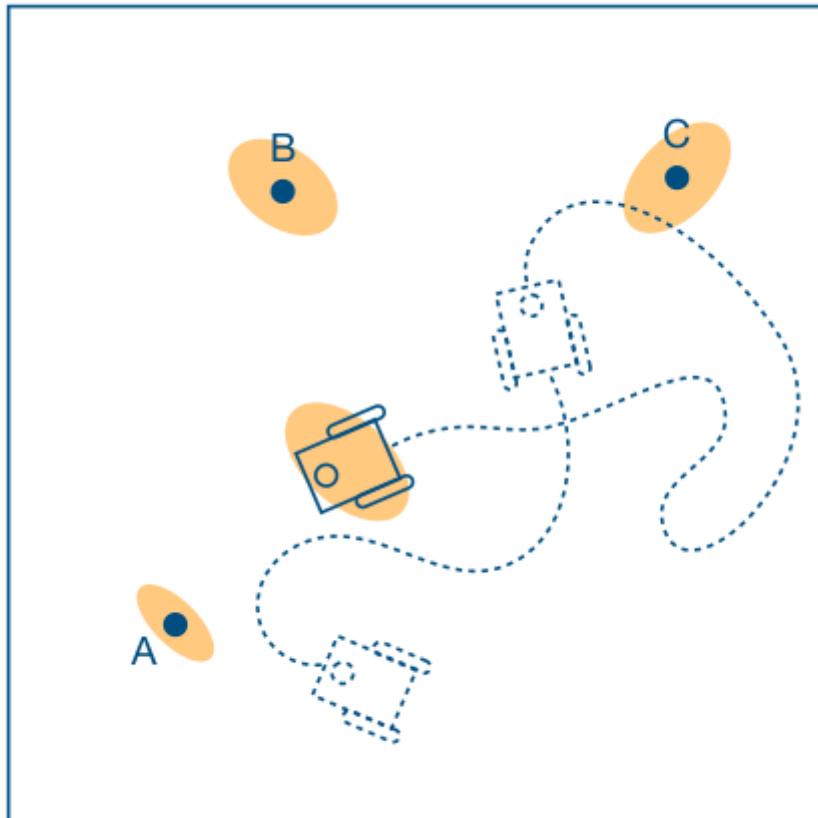
(其实这个问题本质是一个问题，绘制地图也就是综合处理机器人过往的位置信息和感知的地标位置得到的信息)

那么，问题就转换为： **如何通过不断感知自己和地标的相对位置，得到自己的过往和当下的全部位置信息+坐标位置信息**

2. slam算法的基本思想

1. 机器人运动过程中，不断感知地标位置
2. 位姿推测同样存在着误差，且误差随着运动步数的增加而随之增大
3. 机器人在运动过程中每次观测到的地地标数量可能是不同的

4. 通过机器人的观测数据所推测的地地标位置的误差是测量误差与运动误差的累积(combination)
 5. 机器人的位姿误差会随着运动过程不断增加
 6. 机器人在运动过程中可能再次观测到之前遇到过的地标
7. 基于运动过程中较早阶段所获得的地地标位置信息(位置误差较小),机器人可以减少后续阶段中所推导的位姿误差,并同步减少相关地地标位置的计算误差



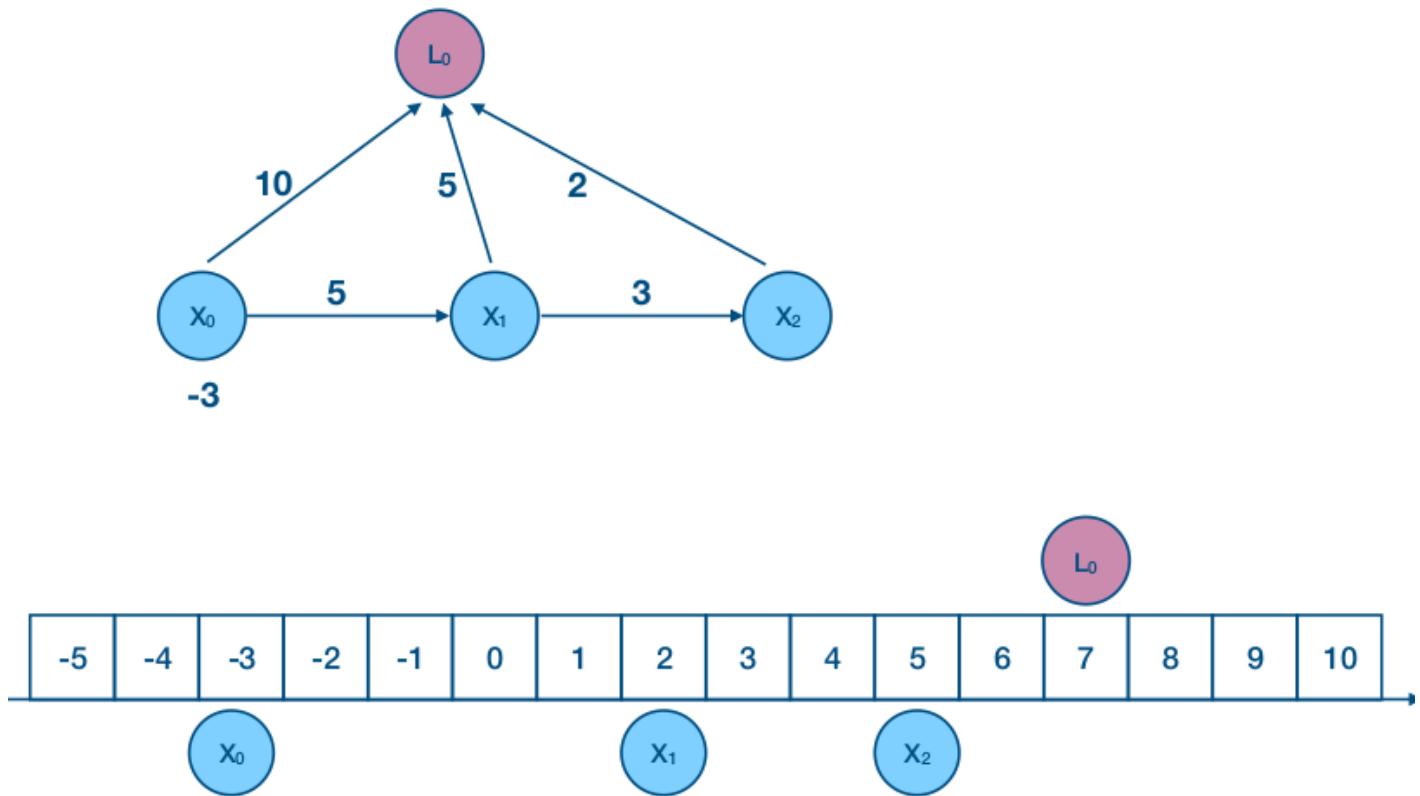
在机器人的运动过程中，存在三种约束：

- 初始位置约束
- 机器人运动相对位置的约束
- 机器人相对于坐标的约束

3. SLAM算法的示例实现

3.1 一维理想情况

机器人的运动过程和感知地标的可视化图如下:



机器人初始位置已知，运动了两次，分别感知了三次地标

我们的目的：计算出机器人的三次位置坐标和坐标的位置

现在我们填写两个矩阵，注意下面的填写过程：

在初始位置向X1运动过程中， $X_1 - X_0 = 5$,左侧矩阵表示位置之间的关系（1表示相加，-1表示纵轴元素减去横轴元素），右侧矩阵表示左侧矩阵计算的结果：

	X_0	X_1	X_2	L_0
X_0	1	-1		
X_1	-1	1		
X_2				
L_0				

 Ω

	X_0	X_1	X_2	L_0
X_0	-5			
X_1	5			
X_2				
L_0				

 ξ

下面的过程都是由于位置转换获取的：

	X_0	X_1	X_2	L_0
X_0	1	-1		
X_1	-1	1+1	-1	
X_2		-1	1	
L_0				

Ω

	-5	X_0
	5-3	X_1
	3	X_2
		L_0

ξ

接下来就是添加地标的位臵約束了，原理和位臵轉換相同：

	X_0	X_1	X_2	L_0
X_0	1+1	-1		-1
X_1	-1	1+1	-1	
X_2		-1	1	
L_0	-1			1

Ω

	-5-10	X_0
	5-3	X_1
	3	X_2
	10	L_0

ξ

	X_0	X_1	X_2	L_0
X_0	1+1	-1		-1
X_1	-1	1+1+1	-1	-1
X_2		-1	1	
L_0	-1	-1		1+1

Ω

	-5-10	X_0
	5-3-5	X_1
	3	X_2
	10+5	L_0

ξ

$$\begin{array}{c}
 \begin{array}{cccc} X_0 & X_1 & X_2 & L_0 \end{array} \\
 \begin{array}{|c|c|c|c|} \hline X_0 & 1+1 & -1 & & -1 \\ \hline X_1 & -1 & 1+1+1 & -1 & -1 \\ \hline X_2 & & -1 & 1+1 & -1 \\ \hline L_0 & -1 & -1 & -1 & 1+1+1 \\ \hline \end{array}
 \end{array}
 \quad
 \begin{array}{c}
 \begin{array}{c} -5-10 \\ 5-3-5 \\ 3-2 \\ 10+5 +2 \end{array} \\
 \begin{array}{c} X_0 \\ X_1 \\ X_2 \\ L_0 \end{array}
 \end{array}$$

Ω ξ

最后添加初始位置的约束，即在X0的位置加1：

$$\begin{array}{c}
 \begin{array}{cccc} X_0 & X_1 & X_2 & L_0 \end{array} \\
 \begin{array}{|c|c|c|c|} \hline X_0 & 1+1+1 & -1 & & -1 \\ \hline X_1 & -1 & 1+1+1 & -1 & -1 \\ \hline X_2 & & -1 & 1+1 & -1 \\ \hline L_0 & -1 & -1 & -1 & 1+1+1 \\ \hline \end{array}
 \end{array}
 \quad
 \begin{array}{c}
 \begin{array}{c} -5-10 \\ -3 \\ 5-3-5 \\ 3-2 \\ 10+5 +2 \end{array} \\
 \begin{array}{c} X_0 \\ X_1 \\ X_2 \\ L_0 \end{array}
 \end{array}$$

Ω ξ

最后，两个矩阵的计算结果如图所示：

$$\begin{array}{c}
 \begin{array}{cccc} X_0 & X_1 & X_2 & L_0 \end{array} \\
 \begin{array}{|c|c|c|c|} \hline X_0 & 3 & -1 & 0 & -1 \\ \hline X_1 & -1 & 3 & -1 & -1 \\ \hline X_2 & 0 & -1 & 2 & -1 \\ \hline L_0 & -1 & -1 & -1 & 3 \\ \hline \end{array}
 \end{array}
 \quad
 \begin{array}{c}
 \begin{array}{c} -18 \\ -3 \\ 1 \\ 17 \end{array} \\
 \begin{array}{c} X_0 \\ X_1 \\ X_2 \\ L_0 \end{array}
 \end{array}$$

Ω ξ

位置矩阵即为左侧的位置关系矩阵的逆和右侧的距离矩阵的乘积：

$$\begin{array}{c}
 \Omega \\
 \downarrow \\
 \Omega^{-1} \quad \bullet \quad \xi = \mu
 \end{array}$$

Ω
 Ω^{-1} \bullet ξ $=$ μ

3	-1	0	-1
-1	3	-1	-1
0	-1	2	-1
-1	-1	-1	3

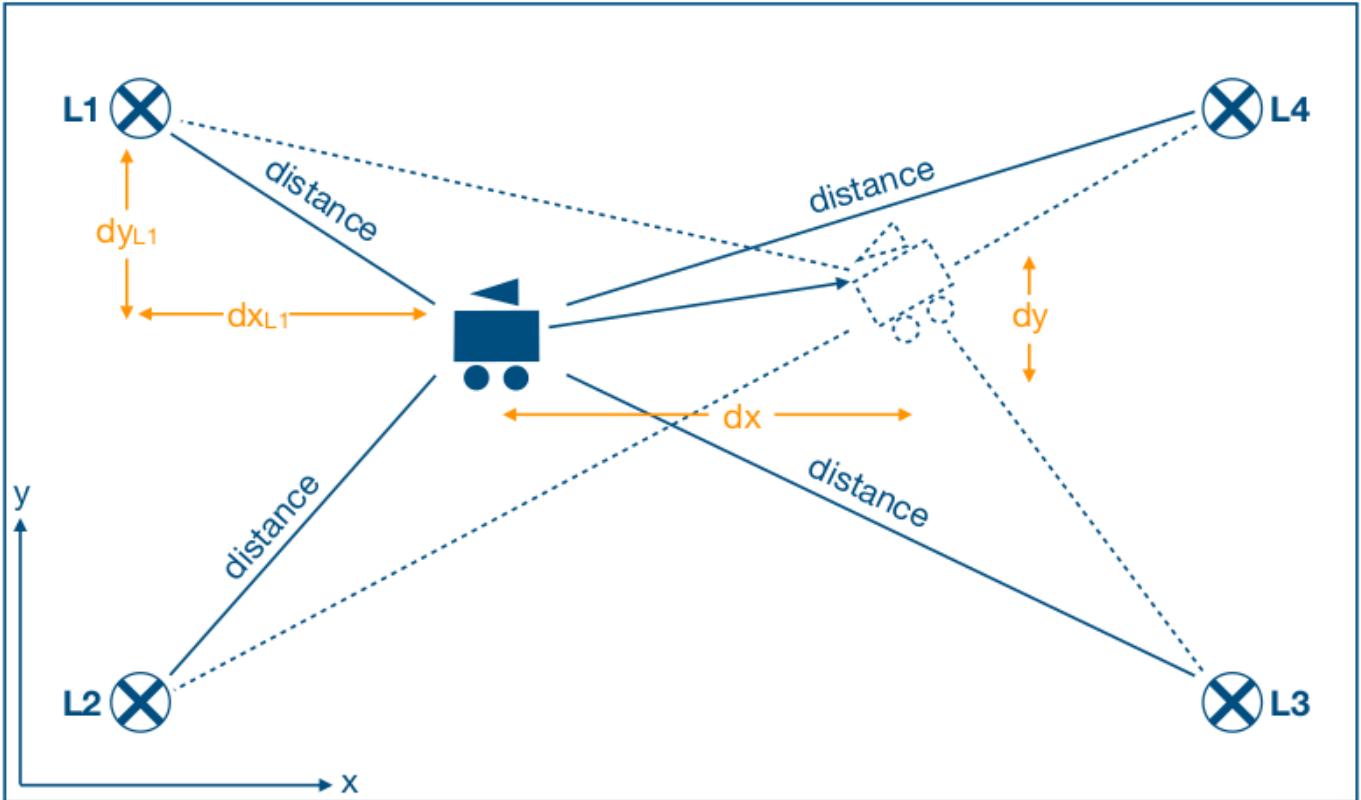
1	1	1	1
1	1.625	1.5	1.375
1	1.5	2	1.5
1	1.375	1.5	1.625

-18
-3
1
17

-3
2
5
7

对于机器人，即使中间的感知情况不存在，对于最终的结果依旧没有影响，但是，如果存在感知误差，就可能会出现位置信息的判断误差；如果添加地标，即在矩阵添加对应的行或者列就可以

3.2 二维理想情况



如果按照上述的一维情况，可能导致巨型矩阵的出现，那么我们需要在就可能保证过往位置和感知信息的前提下，实现信息的压缩，如下图：



Online SLAM算法只保留最近一次运动位置和地标信息来缩减矩阵规模并加快运算效率

[代码实现](#)

以上

参考资料：

[从零开始一起学习SLAM | 为什么要学SLAM？](#)