# Line representation

## Reference

1. Plucker Coordinates for Lines in the Space.
2. Structure-From-Motion Using Lines: Representation, Triangulation and Bundle Adjustment. 本文深度参考这篇论文
3. 多视图几何.
4. <https://zhuanlan.zhihu.com/p/65674067> 关于线特征很好的解读.

## 直线的自由度

3D空间中的直线有4个自由度，通常认为一条线由两个点或者面组成，所以自由度有2x3=6个，但是由于直线绕着自身的方向轴旋转和移动的话，直线还是这个直线，所以真实自由度为6-2=4个。

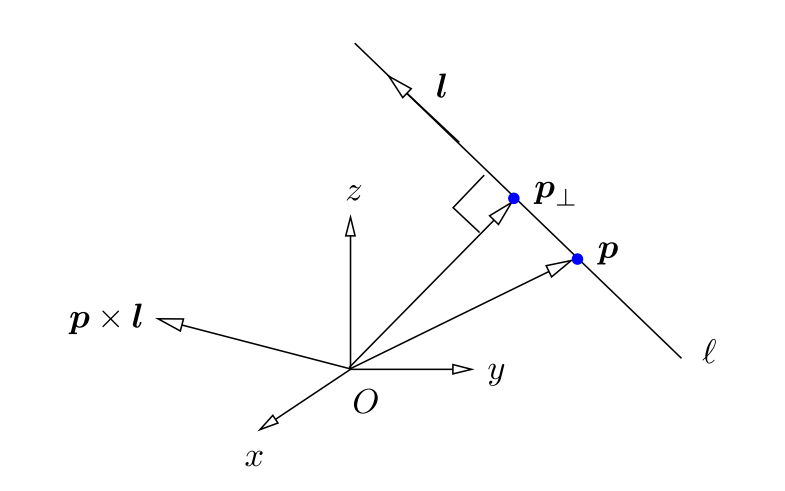
## Plucker坐标系

Plucker坐标系是一种比较重要的表示方法，后面很多的表示方法也能和该方法进行互换

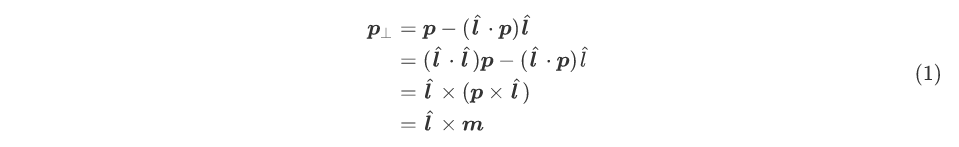
Plucker坐标系使用两个向量表示，，其中：

* 表示直线的方向向量，模长为1；
* , 为直线上的一点，可以看到这部分也表示线与原点组成平面的法线；
* 表示原点到直线的距离；

所以，这个直线的表示形式基本上是方向向量和法向量的表示方法，如下图：



另一点，表示了原点到直线的距离，公式推导如下：



表示把向量映射到直线方向上的长度，也就是上图看到的的长度；所以的模长就是远点到直线的最短距离，因为与垂直，因此：

### 

### 两种特殊的情况

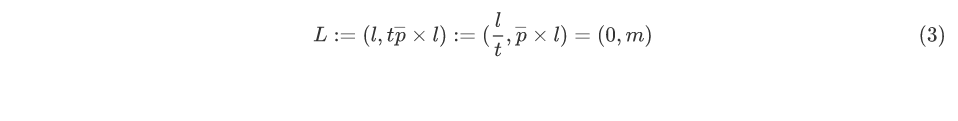
自然而然的想到，如果直线过原点怎么表示？如果直线是无穷远点怎么表示？

#### 过原点的直线

因为直线过原点，那么就把p设为原点，则，跟它的实际意义也相符；

#### 过无穷远点的直线

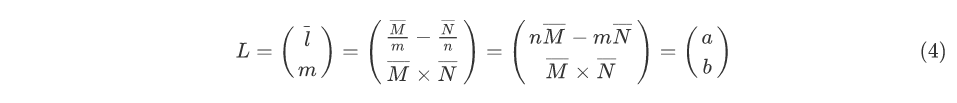
直线过无穷点时，那么就把p设为无穷远点，有，则坐标可以写作：



### 在空间中的表示方法

#### 使用两个点进行表示

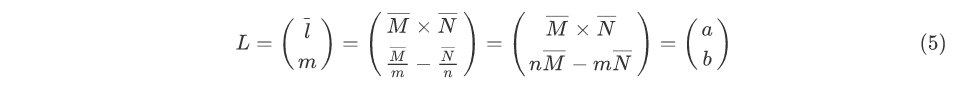
给定两个3D空间中的点，有：，其中一方面使得直线变为其次坐标的表示，另一方面表示该点的逆深度，所以根据上面的Plucker坐标的表示，有：



这里简单分析一下自由度，由上可知，Plucker的表示下一条直线有6个参数表示，是5维齐次坐标系下的齐次表示，同时，因为Plucker坐标中的分别表示线的方向以及线和原点组成的法向量，因此有约束的约束，所以最终Plucker表示下的直线就有4个自由度，和空间直线的自由度一致。

#### 使用两个平面的交表示

这部分属于空间中点和线的对偶形式，给定3D空间中的两个平面，有：，这里就不表示逆深度了，而是表示截距，同样按照Plucker坐标的表示，有：

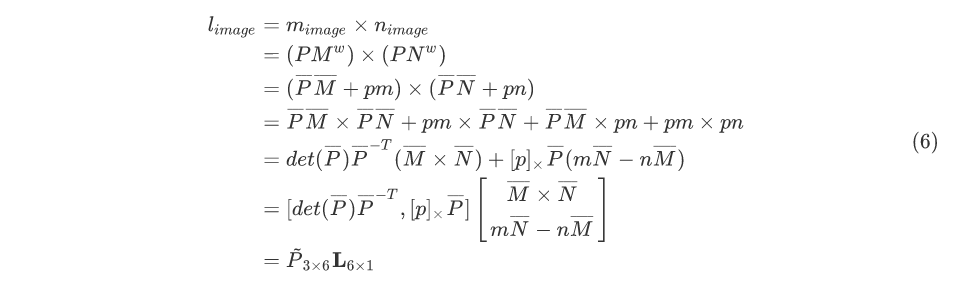


简单的说一下上个公式中的b是怎么得来的，假设点在直线上，则有：

所以可以看到垂直于直线上一点，同时也垂直于直线的方向向量，因此就是直线与原点构成平面的法向量。

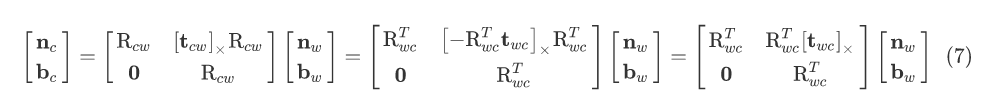
## 投影模型

对于投影矩阵，那么经过投影之后的点而言，有：



对于由两个面相交而得到的直线表示来说，把直线表示替换掉为面表示就可以了（对偶赛高）。

根据上面的推导很容易扩展出世界坐标系到相机坐标系的转移矩阵有：



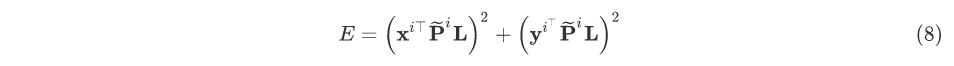
简单说一下上述公式：第一行由公式（6）得到，把其中的用替换就可以了；第二行也很好理解，就是简单的把直线的方向进行了旋转。后面之所以把整个公式变换为camera->world主要是因为这是位姿的表示方法。其中的一步推导用到了一个性质，该性质在李群中比较常见，即：当时，有。

## 三角化

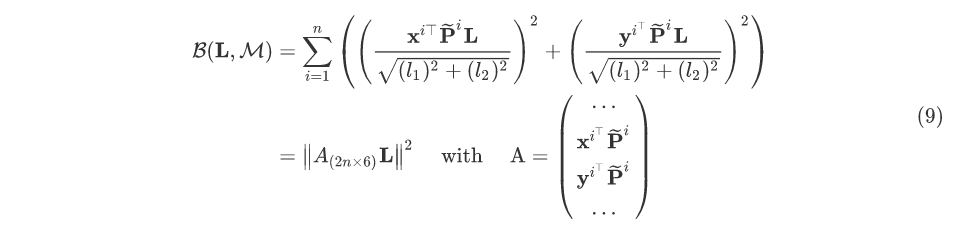
这里主要涉及两个方法：第一个方法是解耦的方法，即先获取表达式的初值，再对初值进行正交补偿；第二个方法是耦合的来解，即**使用迭代的方法得到一个即满足优化函数，又能满足正交要求的解**。下面简单的说明一下这两个方法。

### 获取直线表示的初值

这部分比较简单，优化的目标函数就是投影误差。假设直线被相机观测到，那么对于每一次的观测，可以定义误差：



可以看到基本上就是在第 i 帧上的图像上的两个对应点到投影直线的距离。所以所有的观测都考虑之后有：



其中：

* 设，那么；

该方程有很多的解法，这里就不赘述了。需要注意的是这个部分只是求解了方程，并没有考虑Plucker约束，即。

### 对Plucker约束进行补偿

这部分可以着重参考论文【2】，基本思路是把这个问题当做纯数学问题进行求解，获取与初值最接近且正交的向量，但是这个解就不一定满足公式（9）的最优解了。

### 类线性算法（Quasi-linear algorithm）

这个算法主要是使用迭代的思路使得解既能满足公式（9）的优化函数，又能满足Plucker正交关系的解，主要的思路如下：

1. 先按照公式（9）求得一个初始；
2. 构建一个迭代公式：

* 可以看到这个思路主要想使得**迭代前后两次的方向向量和法向量是正交的**；

1. 为了求解上述公式，显然可以得到最优的是在的零空间中，所以对进行了SVD分解：

* 上述公式中，是唯一一个特解，而就是所在的零空间，所以可以被的五个列向量组成的空间基底线性表示；

1. 那么现在的问题变为如何求解这个线性表示了，或者说变为求解这个向量。其实也简单，经过第三步其实已经得到了，那么设，代入公式（9）后得到：

* 这个时候就比较明了了，构建优化目标函数为：
* 这里其实没有特别想明白为什么约束条件是而不是，私以为可能是为了方便运算，因为上式直接用拉格朗日乘子法解的话，用的约束显然比要容易的多；

1. 经过这个之后，还有一个步骤是要更新公式（9）中的A，因为每个直线与观测线之间的误差都与直线表示有关，因此当获得了新的直线表示之后记得更新这个地方；
2. 重复2、 3、 4、 5步骤直到得到的收敛；

## 线特征的正交表示（Orthonormal Representation）

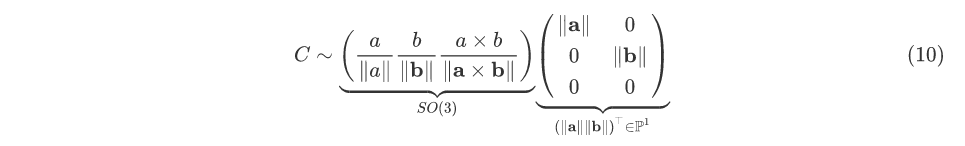
正交表示主要的贡献点在于可以使得整个表示的未知数与自由度是对应的，也就是可以用最小表示进行线特征的表示，这样的好处对于优化问题是不言而喻的，就和李群和李代数的关系一样。下面就简单的介绍一下这个表示方法。

### 正交表示的推导

假设直线的表示为：，写作矩阵形式为，其中：

* 表示直线的法向量，表示直线的方向向量；
* ；

将矩阵进行QR分解有：



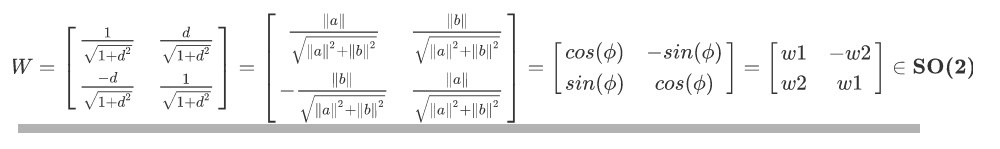
整个分解之后由两部分组成，一部分是旋转矩阵，另一部分是一个一维的齐次坐标的表示。

简单的说一下这个部分：

1. 第二部分的右上角的元素为0是必然的，因为；
2. 因为第二个部分本质上只有两个变量，都除以之后，其实可以得到两个元素为，其中第一个元素根据公式（1）可以知道这部分就是直线到原点的距离。

所以看到，经过QR分解之后的整个直线表示的状态量刚好是4个（第一部分有3个，第二部分有1个），因此说正交表示可以最小表示直线。

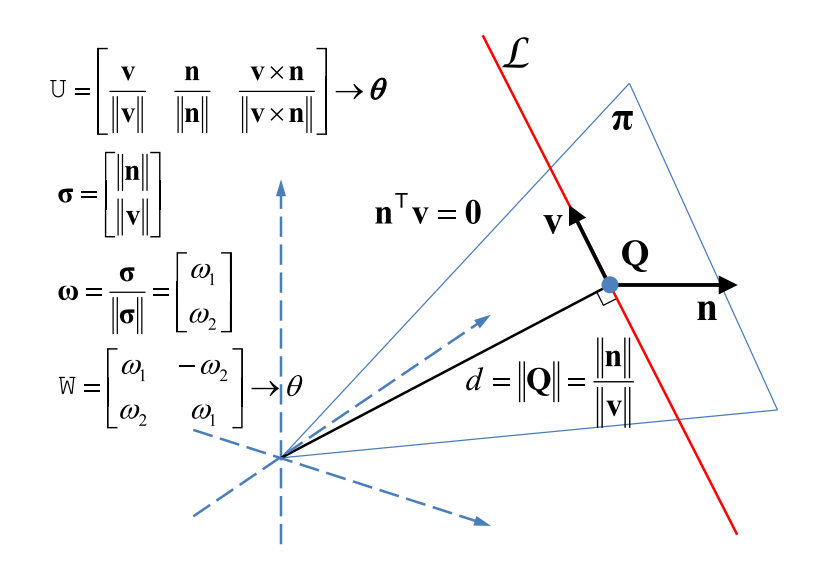
到这里其实还并没有完成正交表示，因为作者认为第二部分并不容易进行优化和更新，所以做了一个变化，把这部分变作了一个的表示，毕竟相比于这个的矩阵来说，的更新更加稳定且容易，最重要的是并没有造成信息的缺失，有如下的过程：



在公式（10）中，比较容易看到，当对进行更新了之后，可以通过的公式来得到直线到原点的距离。

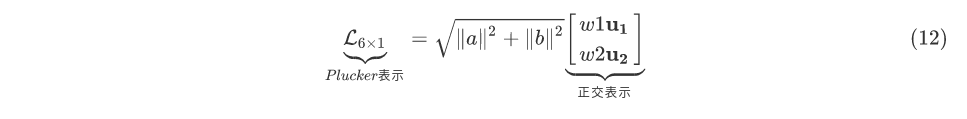
综上所述，直线的正交表示可以写作如下形式：

这部分用下图表示十分合适：



### 正交表示与Plucker坐标系的转换

正交表示可以最小的表示直线，但是该表示方法的投影方程并没有线性的表示，即该方法并不能像Plucker坐标系表示方法一样，通过公式（7）得到一个线性的映射关系，但是我们又特别的看重该表示方法的最小表示的性质，好在两者之间的互换关系还是十分简单的，如下：

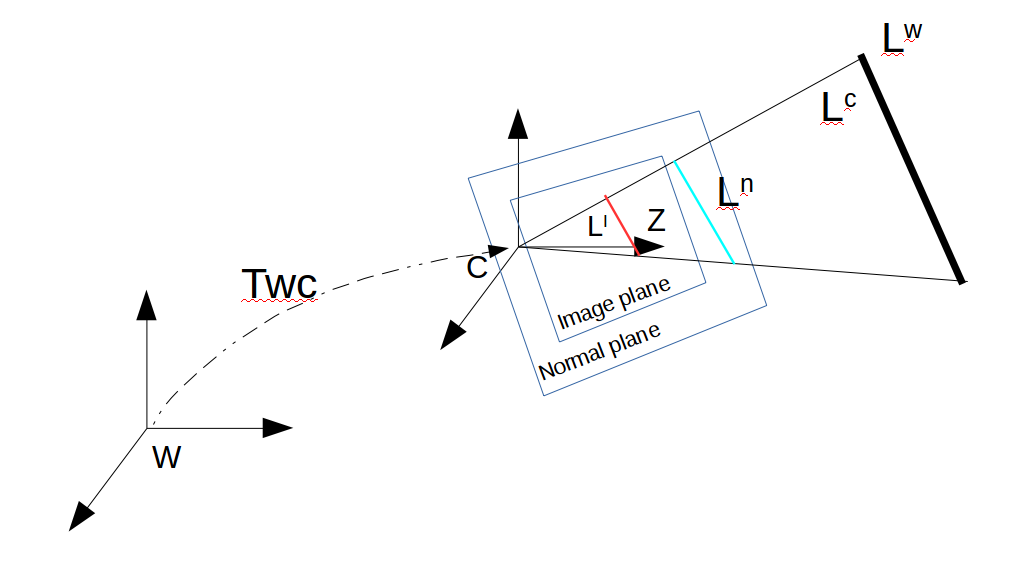


这里面差了一个归一化参数，这部分其实可以不用管，因为可以在算的时候稍微进行一下归一化或者其他的方式限制这个值。

## 优化问题

这部分其实跟点的过程一样，也主要是**观测方程**和**误差函数**，上面谈到的公式（6）和公式（8）其实已经简单的涉及到了这部分，下面的内容主要是从头到尾的串一遍了。

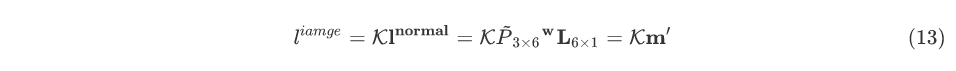
这部分用下面的一张图表示：



其中：

* 表示在世界坐标系的表示，表示在相机坐标系下的表示；
* 表示归一化平面上的线，表示在图像坐标系下的线；
* 这里故意把归一化平面与图像平面分开画，两个平面之间其实差了一个内参矩阵，不过这个矩阵与常见的K不同，这里需要稍微进行推导。

所以这里进行从世界坐标系到图像坐标系的整个投影可以表示为：



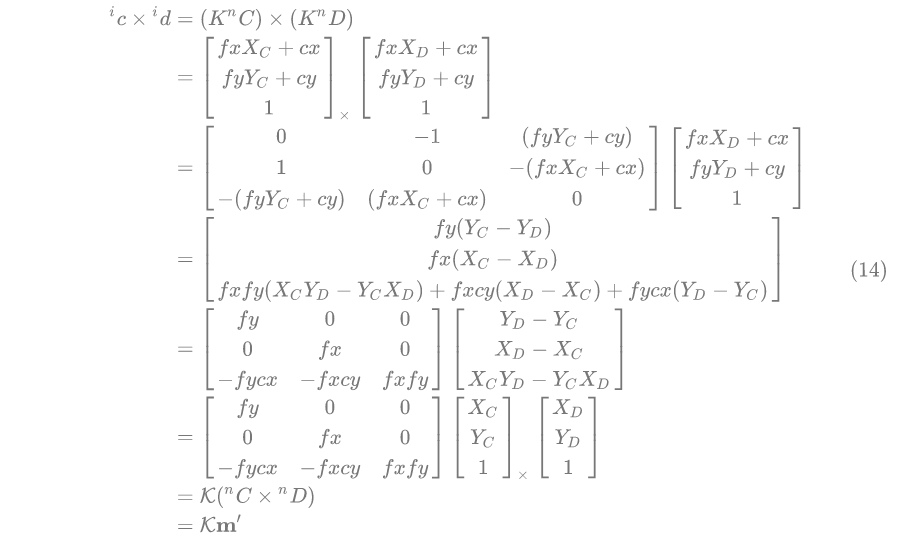
其中：

* 表示直线的内参矩阵；
* 参考公式（６）；
* 这里的表示在相机的归一化坐标系下的法向量；

简单的说一下的推导：

1. 在归一化平面上，假设直线的点为：；
2. 在图像平面上，经过内参映射之后的点为：，其中两者的关系为，其中K就是熟悉的相机内参矩阵；

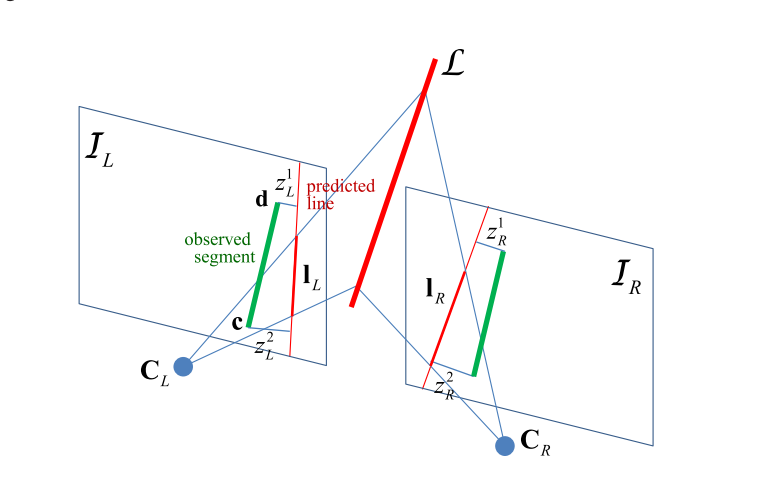
对于归一化平面和相机平面，两者其实都是2维的射影平面，所以在归一化平面和归一化平面上的直线的表示为：



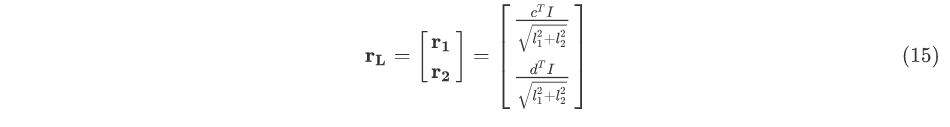
上述的推导其实很容易看出，因为两个归一化平面的点的叉乘就是在相机坐标系下的Plucker表示中的法向量，因此可以看到在归一化坐标系到图像坐标系的转换中，仅仅涉及到了直线表示中的法向量部分。

### 误差函数

用下面这张图表示误差量，如下：



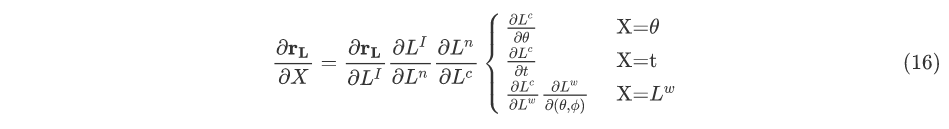
图中的表示直线在图像平面的投影，所以定义误差项为（就是简单的两个点到直线的距离）：



### 求解Jacobian

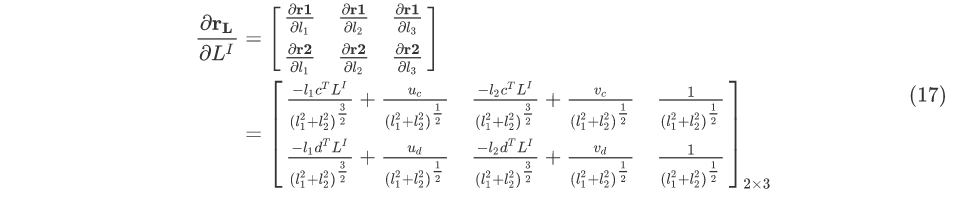
跟对3D点的优化问题一样，就是从误差不停的递推到位姿以及直线表示上，用到最最最基本的求导的链式法则：

通用的公式如下：



先对前面最通用的部分进行求解：

第一部分：

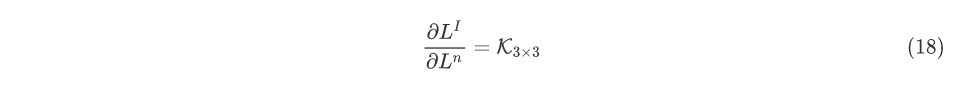


其中：

* 表示图像坐标系下直线的三个参数；
* 表示点c的xy坐标值，同理；

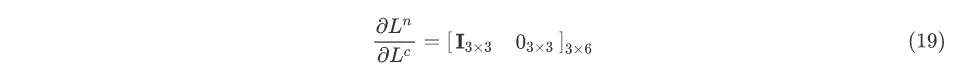
第二部分：

根据公式（13）可知：



第三部分：

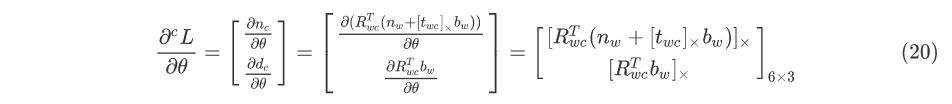
由公式（6）和（13）可知，直线的Plucker表示在归一化平面上只用了其中的法向量部分，因此若有，那么，所以求导有：



第四部分就分这几种情况进行讨论：

#### 对于位姿的姿态部分

根据公式（7）有：



上述的推导使用了李群的右扰动模型，即。

#### 对于位姿的位移部分

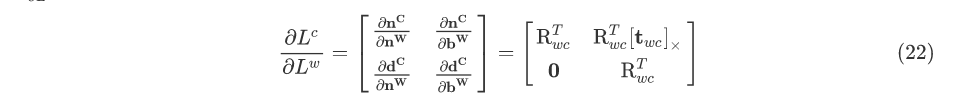
同样根据公式（7）有：

#### 

#### 对于世界坐标系下直线表示部分

这部分按照公式（16）的步骤，依旧分两个部分：

1. 部分：

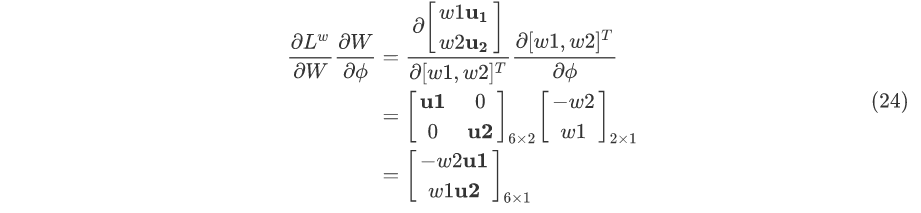


1. 部分，这部分其实还可以继续分，如下：

##### 第一部分

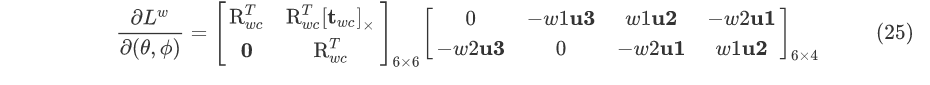
##### 

##### 第二部分



* 其中。

1. 两个部分合起来为：



## 总结

线特征这部分确实比点特征要复杂很多，表示方法就很不同（这部分也是笔者花了很久才接受的一部分），但是好在整个推导过程都可以借助点的转移方程逐步进行推导。