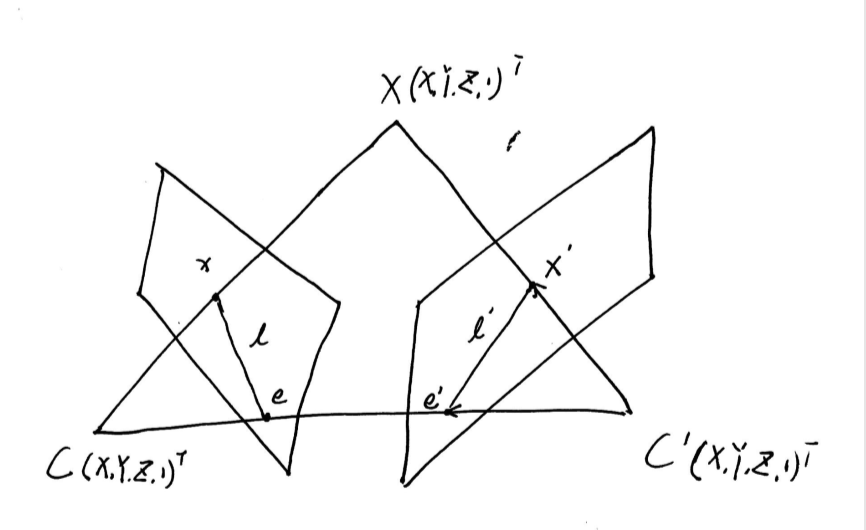
1.

三维空间中的同一个点在不同的两个相机拍摄到的图像下会位于不同的点，但是这两个点之间应该满足一定的关系，可以通过对极几何用Fundamental Matrix 和Essential Matrix来描述这样的关系。其中，首先定义两个相机中心C和C’，分别可以用一个拓展的4维向量来表示，如。在两个相机各自的相机平面中，定义epipole为另一个相机在自己平面上的投影，即e为C’在相机C平面上的投影，这是一个平面上的点，可以用一个拓展三维向量来表示 和上述四维向量类似，最后的维度可以被归一化，因此对该向量的放缩都将表示相同的点。对于三维空间中的一点，其在两个相机平面上的投影分别为x和x’。而上述的e,e’,x,x’应为C,C’,X组成的平面三角和两个相机平面的交点。连接x,e和x’,e’的线分别定义为极线l,l’。可以看出对于x’而言，它应该位于极线l’上，而l’的一端由e’决定，方向受x的影响决定。

Fundamental Matrix 可以用来表示一个图像点x到它在另一幅图像中所对应的极线l’的一个映射关系。也即是，在这里用法向量表示这条直线，先假设为，即（其中P为相机的投影映射关系，即）,从而在另一个图像平面上，同一个三维空间中的点X所对应的点x’应该在这条极线l’上，因此应该满足。

Essential Matrix 可以看作是Fundamental Matrix 的一种特例，也即是在标准化的相机坐标系下的上述映射关系。可以认为是不考虑的相机内参，而仅仅考虑了外参的转换关系的一个矩阵。也即是若相机内参K已知，可以将图像上的点还原为标准化的相机坐标下的点，从而Essential Matrix和Fundamental一样，可以表示的转换关系为。Essential Matrix有5个自由度，考虑 其中I,θ,R,t分别为3\*3单位矩阵，3\*1零向量，3\*3的旋转矩阵和3\*1的转移矩阵,从而 。其中，t有3个自由度，R因为是反对称矩阵（skew-symmetric），因此实质上也只有3个自由度，又由于E的尺度无关性，也即是矩阵元素的幅值大小同等比例放缩不会影响效果，因此总共有3+3-1=5个自由度。

Nister五点法基于Essential Matrix: E具有5个自由度，因此五点可以满足对其计算的要求。对于一个3\*3的E，应该满足秩为2而且他的两个非零特征值相同，这推导出一个充要条件:，这个等式被用来构造方程。首先，由于，把这样方程改写，改写成E的各项所组成的向量和与交叉项组成的向量的内积的形式。具体而言，(θ是3\*1零向量)：



这样，问题转化为求解E的各个元素的问题。由于每一个与的点对得到一个列向量，将5个这样的列向量堆叠起来，可以得到一个矩阵。也即是是一个9\*5的矩阵（其转置矩阵维度为5\*9）。对于可以求得张成其右零空间的四个9\*1向量从而，对应的E应该为这几个向量的线性组合。；，由于E的尺度无关性，可以对其归一化使得w=1。最终要求的化简为x,y,z的问题。把E的结果带入上述充要条件，再利用Gauss-Jordan消元法，可以得到并构造出相应的方程。构造的方法首先是消去x,y的高次项仅保留xy,x,y以及z的幂次项构成的方程，把z和xy分离，得到和的形式。其中B,C都是关于z的高次多项式组成的矩阵。由于非零，因此是一个零向量，可以看出B和C不满秩，行列式为零。计算行列式得到关于z的两个11次多项式，两式消元可以最终得到一个10次多项式。解这个多项式可以得到z的解。把解带入B方程可以解得对应的x和y。

2.

Signed Distance Function（符号距离函数）：在数学上，在一个集合Ω的度量空间内，SDF定义为一个点到Ω边缘的带符号的最小距离，其中定义在Ω内为正，在Ω外为负。可以形式化的表示为：

 其中

其中表示Ω的边界，而inf表示下确界，这里计算的也就是所有表面点中到目标点的最短距离，可以认为是minimum。在三维重建中可以利用SDF对voxel(体元)进行编码，存储体元到物体最近的表面的带符号的距离。更进一步的，TSDF (Truncated SDF) 对SDF进行了阈值化的截断，因为只关心靠近表面的体素，仅存储到观察到的表面的距离绝对值小于一定阈值的体素的带符号距离，从而减少了计算消耗。在kinectFusion中体元中存储的结构除了SDF还有权重，与表面测量的不确定度有关。只有在表面附近范围内的体素才被更新并被用来估计表面。

另外，对于映射到图像中同一个像素的不同体元（由于深度不同）都是深度采样图像的footprint，这些体素如果落在截断区域以外，则被记为free-space。可以通过free-space violation 的检测来去除outlier噪声点。

VoxelHashing在原本TSDF的基础上，为了解决空间中voxel有效数据其实是稀疏的（集中在潜在表面附近），构建空间等大小划分的voxel占用的存储空间太大，消耗GPU资源，不利于实时三维重建的问题，利用了空间哈希表的结构来实时存储和更新voxel数据。首先是将整个空间划分为voxel blocks（每个由8^3个voxel组成）。Voxel的数据结构为一个TSDF值，一个颜色值和一个权重。对于每一个voxel block通过大质数加权求和block坐标的哈希方程计算得到哈希值，把指向这个voxel block存储到哈希表中。而在哈希表中每一个bucket对应于一个哈希值，每个bucket有n个哈希条目。有哈希值冲突的block存储在bucket的空闲条目位置，若bucket也满，则在bucket最后以链表的方式存储剩余的有哈希值冲突的voxel blocks，从而解决冲突和溢出问题。每个哈希条目的数据结构为一个位置信息，一个用于解决冲突的链表的偏移指针以及一个指向voxel block数据位置的指针。在查询的时候，利用block的坐标首先计算哈希值，在对应的哈希bucket以及bucket尾部的list中找坐标信息相同的条目，取用该条目的指针得到对应的block的数据。对哈希表的操作包括插入，取用和删除。利用DDA的方法分配voxel Block，把深度样本中和射线相交的block插入哈希表中，同时在GPU显存的堆中分配一个实际存储block数据的空间，用一个list来存储未分配的空间。在TSDF integration中，可以使用更有效更加紧凑的哈希表，通过筛选出哈希表中已经被占用以及在当前视锥中的的条目，合并并拷贝到另外的缓存中处理。同时也有GPU到host（CPU）的数据流以block的方式吧在视锥之外的点给到CPU，而反向以更大的chunk的数据流给回GPU。在论文的结果上，哈希表只需要34MB的大小就可以高效的操作以及访问1GB的堆数据（存储voxel blocks）。在实时的效率以及质量上都有所提升。