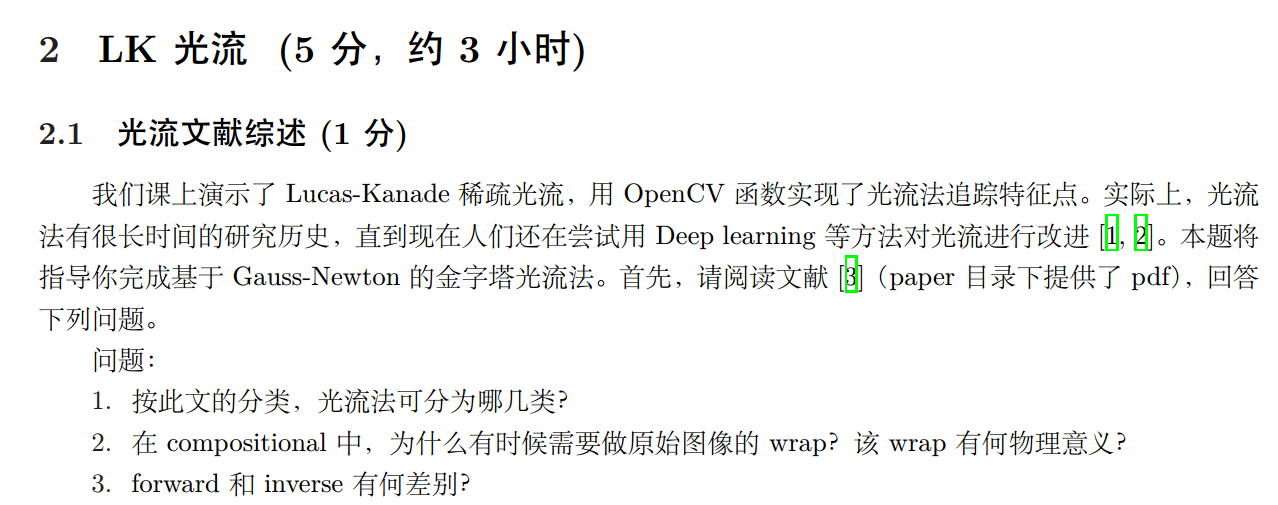
**2018年第6次课习题答案**

**——Vance吴方熠**



答：

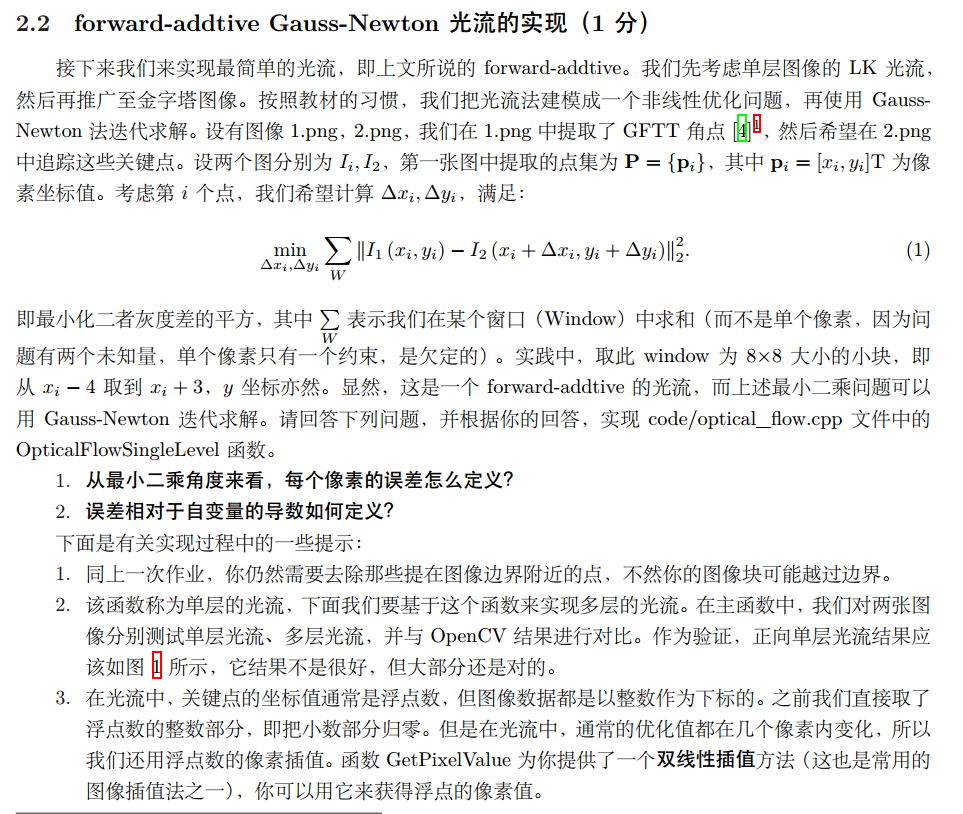
1.图像配准的方法包括图片梯度下降法，差分分解法和线性回归法等，其中梯度下降法是一个标准的解决方案。对于梯度下降法来说，又因是否估计参数的累加增量，或估计的增量变换（wrap）是否与当前的变换估值相组合，而有所不同。另一个区别是算法是否在每个梯度下降步骤中是否执行了高斯牛顿法（Gauss-Newton）、牛顿法（Newton）,最速下降法（steepest-descent）或列文伯格马夸尔特（Levenberg-Marquardt）近似法。

本文中以wrap的更新规则作为分类，将图像配准的方法分为了正向（forward）和反向（inverse）、加性（additive）和组合（compositional）两两匹配出的四种方法。

2.在组合法（compositional）中，参数更新增量引起的增量变化必须组合到当前的变化估计中（即），而不是简单地作参数加性更新（即），故在compositional中需要做原始图像的wrap.

在前面提到的更新规则里有两个要求：①这组wrap必须包含“幺元”；②这组wrap对其“compositional运算”是封闭的。因此这组wrap运算必须形成一种“半群”（semi-groups）,这就是其物理意义。

3.forward的方法在每次迭代更新时，计算要追踪图像上的特征点的灰度梯度；而inverse的方法，可直接使用原图像中特征点的灰度梯度进行迭代，不需要在每次迭代过程中重新计算。



答：

1.每个像素的误差定义为两幅图像对应点之间的**光度误差**，其中对应像素是根据对应特征点上取相同的patch上对应位置的像素。

2.误差相对于自变量的导数定义为该点在x和y方向上的梯度，是一个二维向量。

正向光流效果与参考答案的对比如下图所示：

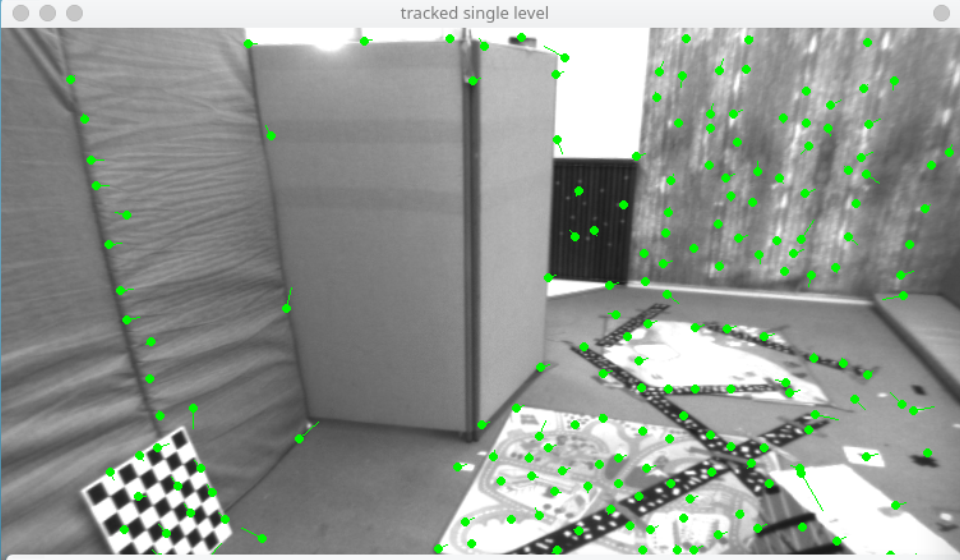
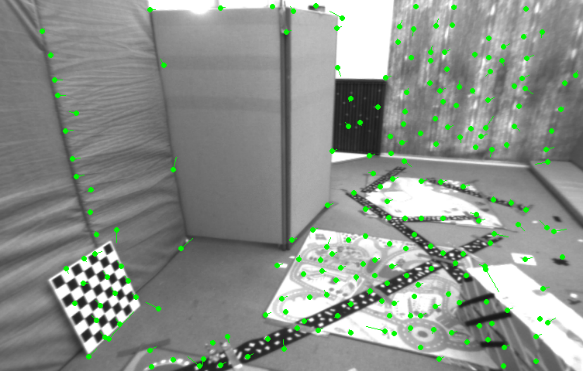
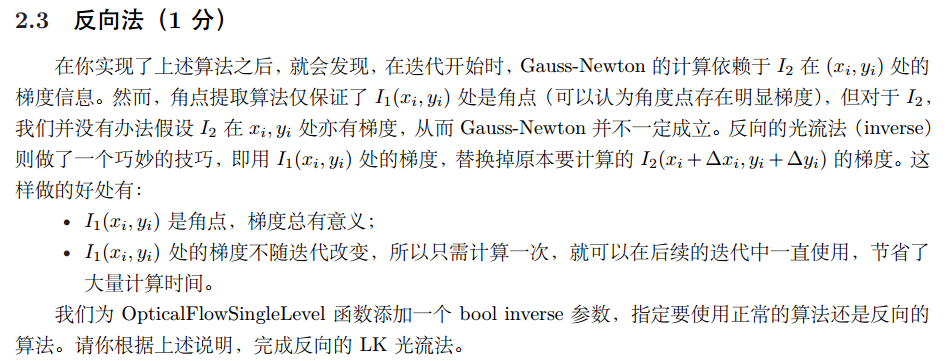


图1 正向单层光流法效果（左）与参考答案（右）的对比



解：

反向光流效果如下图所示：

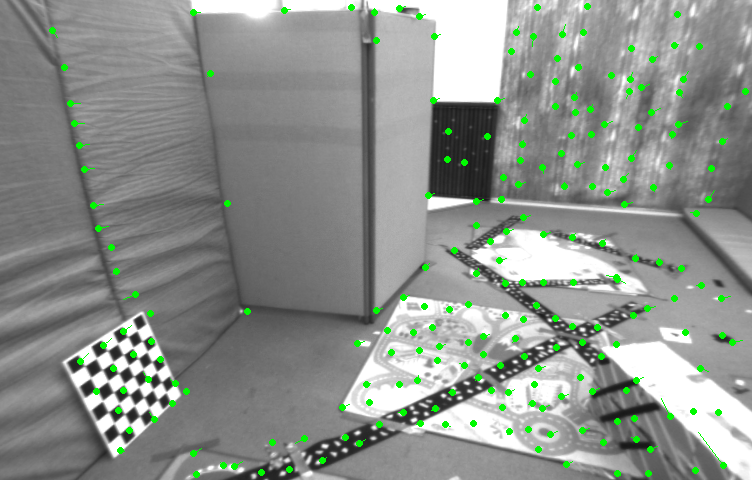
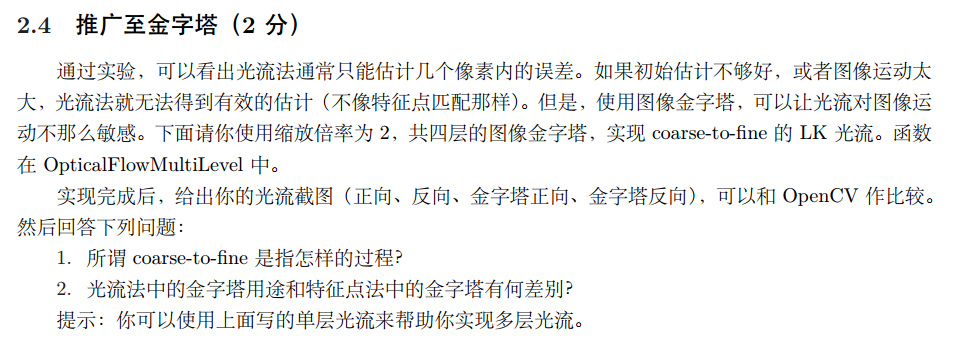


图2 反向单层光流法效果



答：

1.由于光流法要求相机的运动是微小的，它对长距离的移动表现不够好，这时可以用coarse-to-fine的方法来提高其健壮性。所谓的coarse-to-fine的过程是指为图像构建图像金字塔，并通过在每层金字塔上进行光流来摆脱小运动的约束。如果构造了一个n层的金字塔，那么顶层中一个像素可以代表底层个像素的距离。

2.光流法中的金字塔用途，是为了解决其“微小运动假设”的约束，使得光流法在相机进行较大尺度运动的情况下任可以起作用。特征点法中的金字塔，是为了让特征点具有尺度不变性，加强特征点的“可识别性”和“独特性”，为后续的特征点匹配搭建一个更好的条件。

金字塔正、反向光流效果如下图所示：

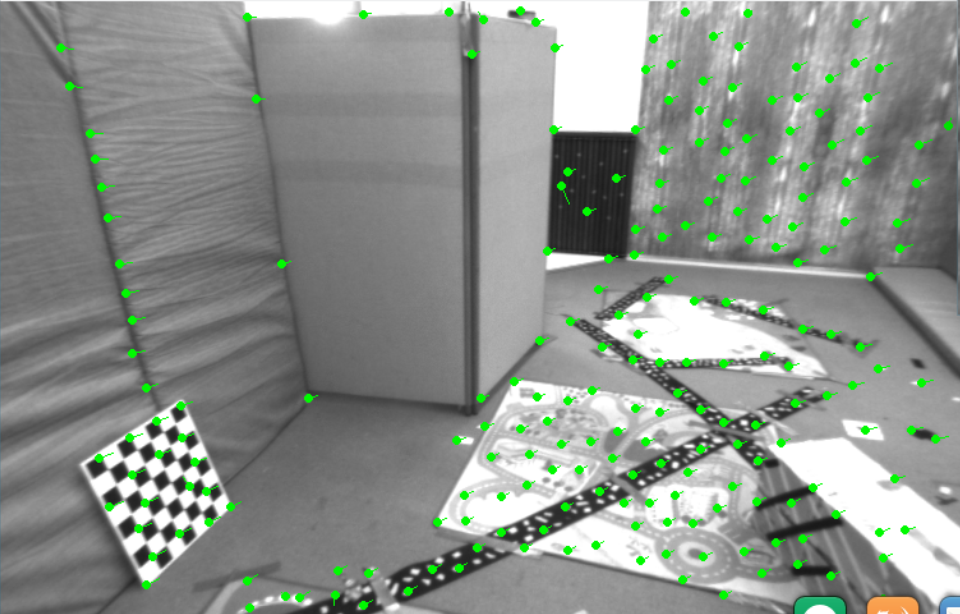
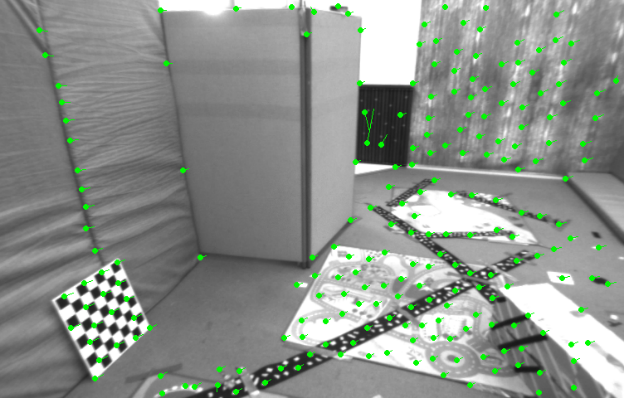


图3 正向多层光流法效果（左）与参考答案（右）的对比

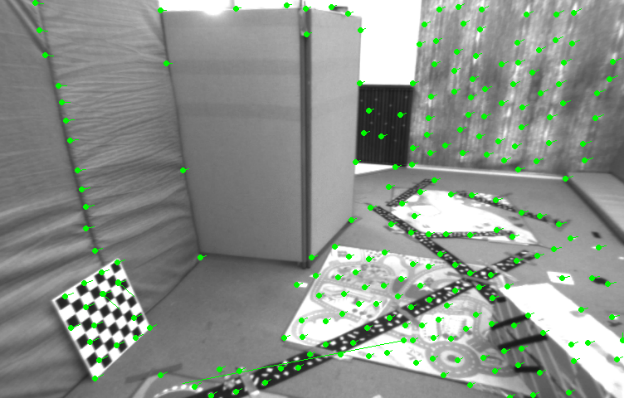
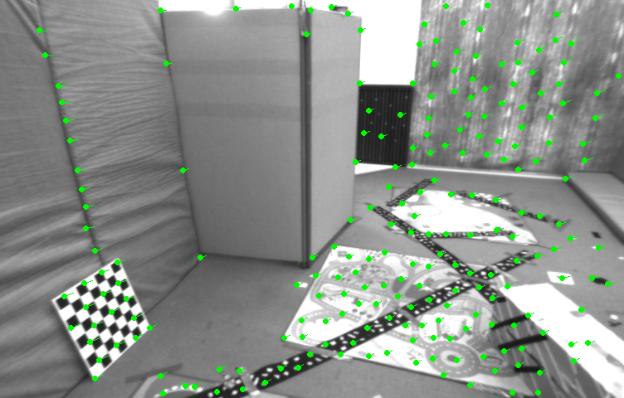
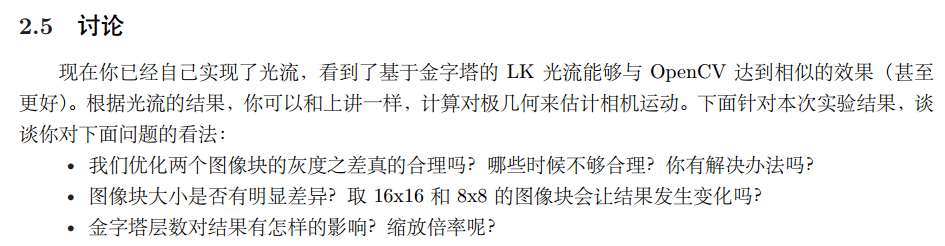


图4 反向多层光流法效果（左）和使用OpenCV计算的多层光流法效果（右）



答：

1.不太合理。在实际应用中，由于遮挡性、多光源、透明性和噪声等原因，使得光流的“灰度守恒假设条件”不能满足，所以优化两图像块的灰度之差在实际应用一般不会被对精度要求比较高的系统采用。我的解决思路：可以考虑分通道计算光流。

2.patch的大小对多层光流法最后的结果从程序上来看没有太明显的差异，只是随着patch的增大，图像边界上的特征点会被逐渐剔除掉，这是因为我们程序中对patch超出边界的特征点选择了删除处理。patch对单层光流法的效果影响相对较大，大的patch会提高单层光流的精度。图5是正向单层光流法在patch取32x32和100x100时效果。

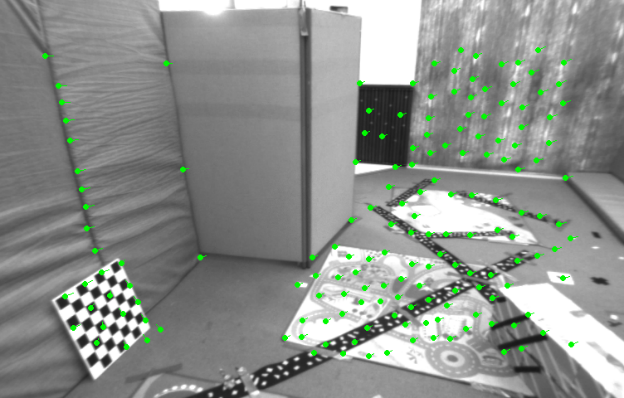
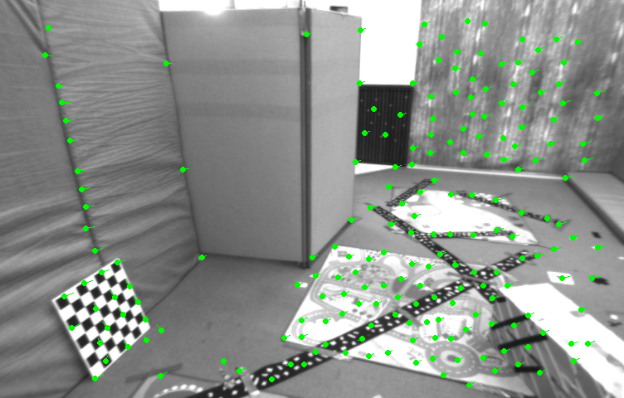
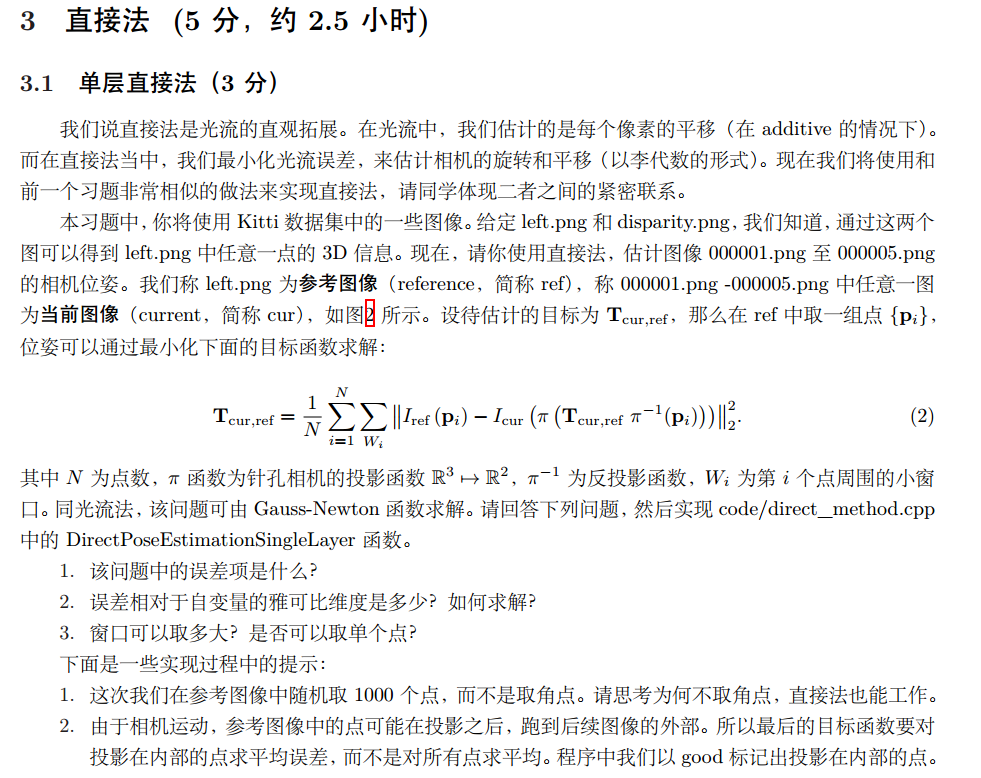


图5 正向单层光流法patch-32x32效果（左）

和正向单层光流法patch-100x100效果（右）

但在实际应用中，一般都使用多层的光流法。OpenCV里计算的patch默认尺寸为21x21，而在本例程序中OpenCV计算的光流图像块仅取到8x8大小。patch的大小对多层结果影响不大，但计算时间会随着patch增大而增加。另一方面，patch取太大的话也会违反其“空间一致性”的假设，故实际使用时patch不必取的太大。

3.金字塔层数越多，缩放倍率约小，就意味着更精细的光流效果和更大的计算量。金字塔层数一般取3-4层即可，缩放倍率一般为2。



答：

1.误差项是对应像素点的**光度误差**:

2.误差项是常量，自变量是6维李代数，故误差相对于自变量的雅克比维度是6维的，即：

其中为投影点在相机坐标系下的三维坐标。参考《十四讲》8.4.1小节可知：

|  |  |
| --- | --- |
|  | （3.1） |
|  | （3.2） |
|  | （3.3） |

式（3.2）与式（3.3）相乘后得到的即是《十四讲》中的式（8.15）。

3.①窗口的大小不要取得太大，一方面大窗口会让计算量增加，另一方面一个大窗口内可能会包含多个要计算的像素点，这样就产生了很多重复的计算。②不可以取单个点。由于程序初始迭代时传入的相机位姿初始值为“幺元”变换（即无变换），如果窗口仅取单个点，那么每次迭代时求出的位姿更新量均为0，两幅图像间的匹配全部变成了相同像素位置的匹配了，见图6.（由于1000个点太过密集，这里大致画出了1/3的匹配点）

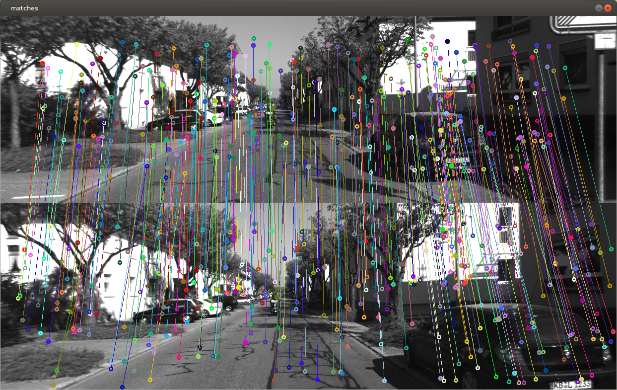
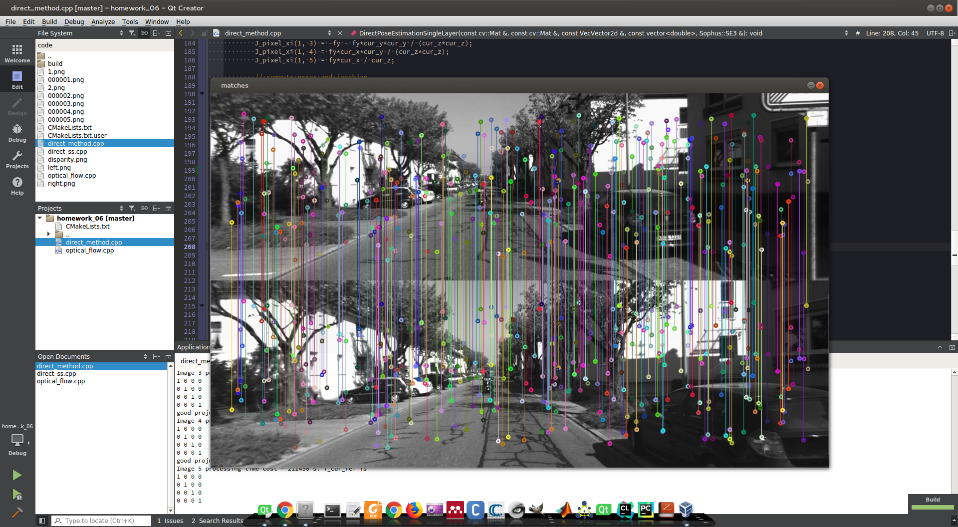


图6 单层直接法窗口取1个点（左）和取64个点（右）时两帧间的匹配情况

4.直接法是根据当前相机的位姿估计值来寻找对应的像素点，它可以对所有的像素进行处理，所以并不需要提取特征点也可以工作。

单层直接法的匹配情况见图6右侧，计算输出结果见图7.

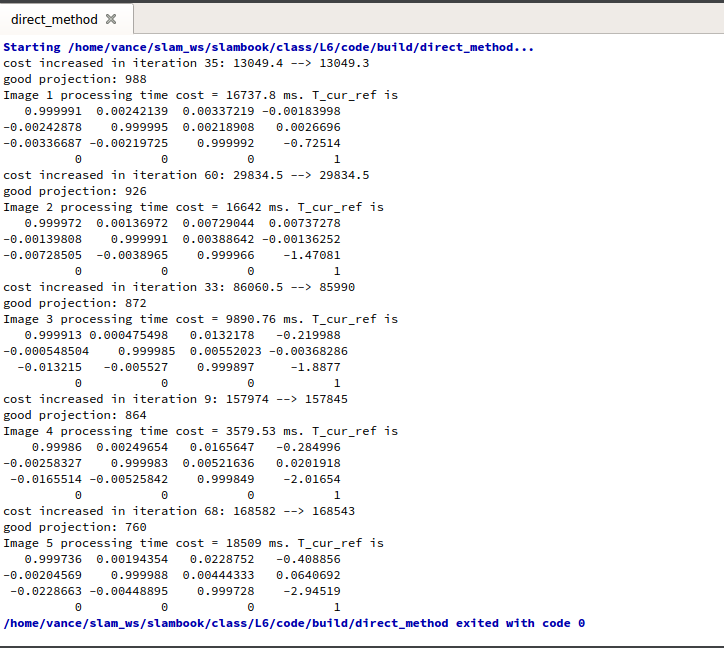
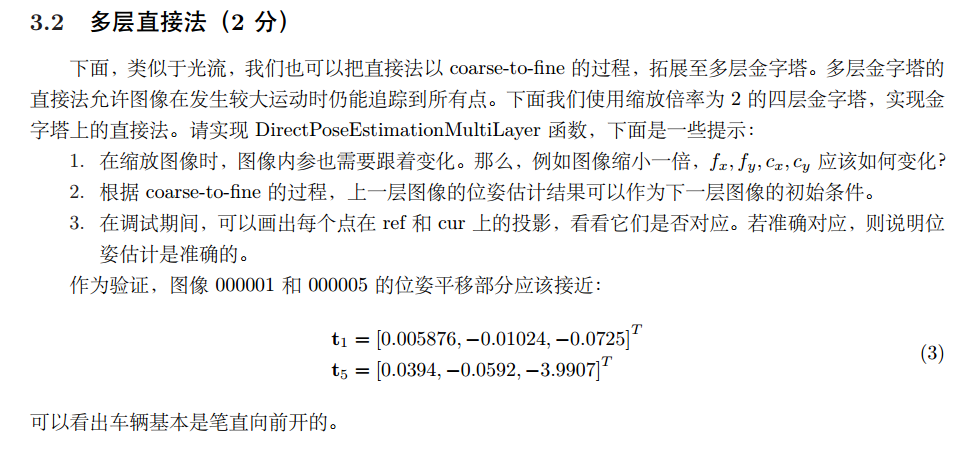


图7 单层直接法计算输出结果



答：

当图像缩小一倍时，即的值缩小了一倍，由于点的三维空间坐标是不变的，根据针孔相机投影模型：

可知相机内参矩阵缩小了一倍，故相机的内参值也缩小了一倍。

多层直接法的计算输出结果如图8所示。

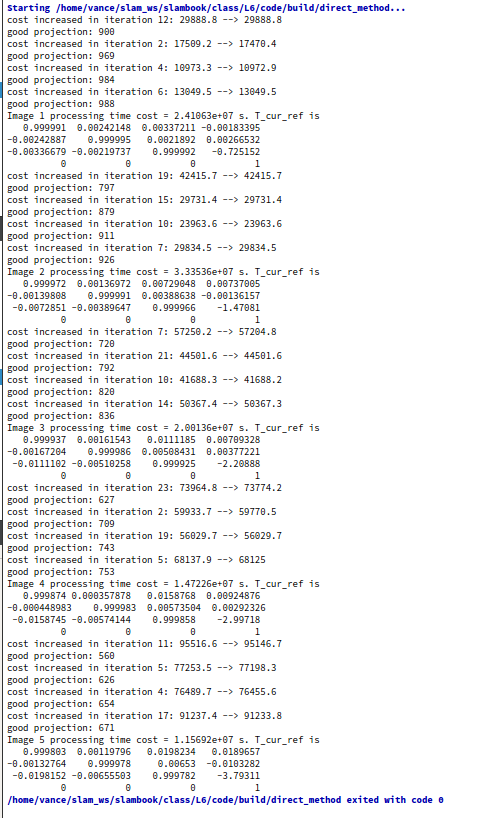
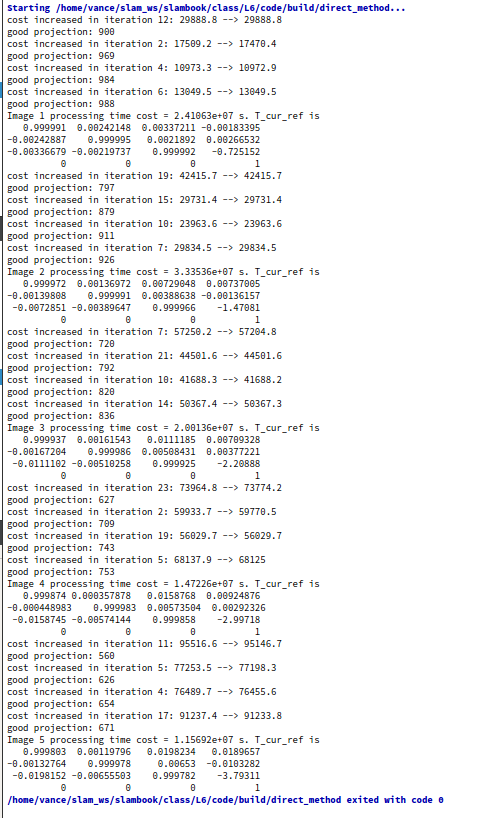
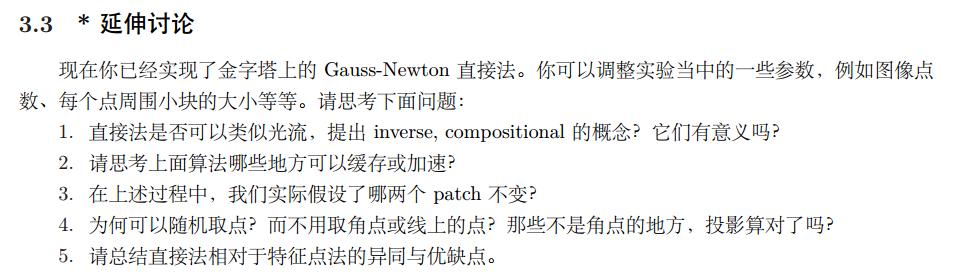


图8 多层直接法计算输出结果



答：

1.inverse的概念对直接法来说没有意义。光流提出的inverse是为了保证估计点的梯度总是很明显（因为角点处总有明显的梯度），且降低计算量。而直接法的取点并不用取特征点，所以就算是inverse也无法保证点有明显的梯度。

由于直接法完全依靠优化来求解相机位姿，这里的位姿由一个6维的李代数来表示，其位姿的更新规则采用compositional会比较合理。

2.暂未发现。

3.实际假设了两帧间对应像素点的patch不变，均为8x8的大小。

4.直接法是根据当前相机的位姿估计值来寻找对应的像素点，它可以对所有的像素进行处理，所以并不需要提取特征点也可以工作。投影依然可以算对，投影情况见图6右侧。

5.见下表：

表1 直接法和特征点法的异同与优缺点统计表

|  |  |  |
| --- | --- | --- |
| **方法** | **直接法** | **特征点法** |
| **相同点** | 目的都是为了找到匹配的像素点，进而求解相机运动 | |
| **不同点** | * 特征点法要计算特征点和描述子，直接法不要 * 特征点法通过最小化重投影误差来优化相机运动，直接法通过最小化光度误差 * 特征点法智能构建稀疏地图，直接法可以构建半稠密甚至稠密地图 | |
| **优点** | * 节省了计算特征点和描述子的时间 * 只要求有像素梯度，不需要特征点 * 可构建半稠密甚至稠密地图 | * 运行稳定 * 对光照、动态物体不敏感 * 特征信息可做回环检测的依据 |
| **缺点** | * 非凸性，对大运动效果不好 * 单个像素没有区分度 * 灰度值不变假设太强，容易受相机暗角、模糊、曝光等因素影响 | * 计算特征点和描述子耗时 * 忽略了特征点以外的信息 * 在特征缺失的地方无法工作 |



由于时间有限，本题暂未解决。