# 2.1光流文献综述 (1 分)

## 1. 按此文的分类,光流法可分为哪几类?

Additive， compositional ，forwards or inverse的四种组合

## 2. 在 compositional 中,为什么有时候需要做原始图像的 wrap?该 wrap 有何物理意义?

因为compositional和additive的更新方式不一样。Warp是对图像的一种变换，比如affine变换，homograph变换或者rotation变换。这种方式的更新量通常是一个矩阵乘在另一个矩阵上，类似T乘以delta T。（而不是T加delta T）。

## 3. forward 和 inverse 有何差别?

## Forward中图像梯度放在图像2上面计算，每次迭代都要重新算。

## Inverse法图像梯度放在图像1上面计算，不随迭代而改变。 2.2forward-addtive Gauss-Newton 光流的实现(1 分)

## 1. 从最小二乘角度来看,每个像素的误差怎么定义?

我们需要找到运动使得同一个特征点在两幅图像中像素值之差最小。

则这里的误差即是图像1中(x,y)的像素值和图像2中(x+dx,y+dy)的像素值之差。

## 2. 误差相对于自变量的导数如何定义

像素值相对于dx对dy的偏导其实就是像素值在这两个方向上的梯度，按照课程视频介绍我们用右边一个像素减去左边一个像素值再除以二来近似X方向的梯度。Y方向同理。

同时还有其他求梯度的算法比如Laplacian。

代码

A screenshot of a computer screen

Description automatically generated

结果，可见有一些点的运动明显不合群。

A picture containing indoor, green, computer, sitting

Description automatically generated

# 2.3反向法(1 分)

代码见上图inverse == true部分。

# 2.4推广至金字塔(2 分)

## 1. 所谓 coarse-to-fine 是指怎样的过程?

即粗到精过程，从最小的缩放图做光流，逐步到在原始图像上做光流。

因为光流法要假设相机运动较小，特征点的亮度不变特性只在局部生效，所以遇到大运动的情况需要“缩放”图像，相当于在更大的局部范围中找到dxdy的运动。

## 2. 光流法中的金字塔用途和特征点法中的金字塔有何差别?

光流法的金字塔是通过缩放图像应对大运动。光流法要在同一个level的金字塔图像中进行。

特征点法金字塔是通过缩放图像来应对尺度变化。特征点可以跨level寻找。

但是思想上是类似的，都是通过金字塔来把让一些利用局部信息的算法更鲁棒。

代码

A screenshot of a computer screen

Description automatically generated

A screenshot of a cell phone

Description automatically generated

结果，明现多数点的运动向着一个方向。

A picture containing indoor, green, table, computer

Description automatically generated

# 2.5讨论

## • 我们优化两个图像块的灰度之差真的合理吗?哪些时候不够合理?你有解决办法吗?

图像块的灰度之差本身就是怕只用单个特征点的像素差不够鲁棒。但是图像块也有自己不够鲁棒的时候，比如光照变化剧烈，反光之类的。可以试试除以灰度均值，得到一个去除亮度变化的结果。

## • 图像块大小是否有明显差异?取 16x16 和 8x8 的图像块会让结果发生变化吗?

一些特殊位置的点的运动结果有差异。

下图为多层金字塔Batch8结果

A picture containing indoor, green, table, computer

Description automatically generated

下图为多层金字塔batch为16的结果

Batch增大之后会把更多的周围像素纳入进计算，有些特殊位置的特征点的运动结果就改变了（也可能有些特征点就匹配失败了）。比如上图中央部分那块黑色的物体上有一个特征点的方向是朝右下，batch扩大之后那个点的方向就看起来更正确了。

A picture containing indoor, green, computer, table

Description automatically generated

## • 金字塔层数对结果有怎样的影响?缩放倍率呢?

下图为原始的4层0.5倍率结果

A picture containing indoor, green, table, computer

Description automatically generated

下图为8层0.5倍率结果。

理论上8层可能更精细一些，对于大运动的鲁棒性更强一点，不过实验中可能相机运动比较小，体现不出来。

A picture containing indoor, green, computer, table

Description automatically generated

下图为4层scale=0.3333结果。

可见图中央黑色块状物有一个特征点的方向发生了变化（类似上题batch改变的结果）

同时左下角棋盘格有一个特征点的运动看起来不太正确，地板右边边缘处有一个特征点的运动幅度变大。

A picture containing indoor, green, table, computer

Description automatically generated

# 3 直接法 (5 分,约 2.5 小时)

## 该问题中的误差项是什么?

类似光流法，是光度误差。

## 误差相对于自变量的雅可比维度是多少?如何求解?

误差e是1维，自变量T是6维度，则J为1行6列。

其中

## 窗口可以取多大?是否可以取单个点

课程中窗口为3x3.理论上可以取单个点，鲁棒性降低而已。还有就是没有像素变化的单个像素点就不对结果产生贡献。所以还是不要只取一个点。

代码（主要是构建error，J和updates）

A screenshot of a computer screen

Description automatically generated

下图为0001图的结果，可见cost一直在下降。（0002-0005图略）

A close up of a logo

Description automatically generated

# 3.2多层直接法(2 分)

主要就是把图像resize到不同的level。同理resize内参。

在不同的level上跑单层直接法

代码

A screenshot of a cell phone

Description automatically generated

结果

0001

Z位移-0.697接近参考值-0.725

A screenshot of a cell phone

Description automatically generated

0005

Z运动-3.79接近参考-3.99。

A screenshot of a computer

Description automatically generated

# 3.3\* 延伸讨论

## 1. 直接法是否可以类似光流,提出 inverse, compositional 的概念?它们有意义吗?

## 不行，直接法怎样都需要把第一帧图像的点投影到第二帧图像上面。链式jacobian中就包括一项第二张图像的像素梯度。而上述思想避免了求第二张图像的像素梯度，这跟直接法的jacobian定义相悖。

## 2. 请思考上面算法哪些地方可以缓存或加速?

雅可比矩阵可以并行计算，反正是每个点的结果累加。

## 3. 在上述过程中,我们实际假设了哪两个 patch 不变?

除了定义灰度不变外，同一个patch内我们用的是同一个深度，即深度也不变。

## 4. 为何可以随机取点?而不用取角点或线上的点?那些不是角点的地方,投影算对了吗?

直接法不需要寻找特征点的匹配，它计算的是像素前后帧的运动。不需要像素本身有特征性质。

下图可以看到，中间天空部分的点都找对了，而这些点显然不是特征点。

同理还有建筑物墙壁点，地面点。A close up of a street

Description automatically generated

## 5. 请总结直接法相对于特征点法的异同与优缺点。

都是通过像素在前一帧和后一帧的对应关系来计算两帧图像之间的运动。

相同点

特征点法是找到特征点对，构建的误差是x-y坐标的误差。

直接法找的是同一个像素，构建的误差是像素灰度误差。

不同点/优缺点

因为像素亮度不会真的一成不变，所以特征点法应该是更精确一些。

不过特征点法要找特征点，计算描述子，还要匹配特征点，比较耗时。

直接法省去了这些步骤，更快一些。而且可以在低特征的场景工作。

同时如果重建场景，特征点只能还原特征点在场景中的3d位置，也就是稀疏点云。

直接法在算力足够的情况可以计算所有像素的运动，从而重构稠密场景。

# 4 \* 使用光流计算视差(2 分,约 1 小时)

即用第一张图片的特征点kp1的x坐标跟第二张图的特征点kp2的x坐标相减得到视差。

代码

A screenshot of a cell phone

Description automatically generated

为了方便显示，我把得到的视察数据（假设范围是-100到100）通过一个scalar变换到了(0,255)的范围。

显示结果如下。大致可以看出来视察点的位置和特征点是对应的。

A picture containing sitting, window, table, front

Description automatically generated