光流法

LK光流（Lucas-Kanade Optical Flow)

光流是物体在三维空间中的运动（运动场）在二维图像平面上的投影，由物体与相机的相对速度产生，反映了微小时间内物体对应的图像像素的运动方向和速度。

光流法假设：

1）亮度恒定，图像中物体的像素亮度在连续帧之间不会发生变化；

2）短距离(短时)运动，相邻帧之间的时间足够短，物体运动较小；

3）空间一致性，相邻像素具有相似的运动；

记 $I(x,y,t)$ 为 t 时刻像素点 (x,y) 的像素值，那么根据前两个假设，可得到：

$$I(x,y,t)=I(x+dx,y+dy,t+dt)$$

一阶泰勒展开：

$$I(x+dx,y+dy,t+dt)=I(x,y,t)+\frac{\partial I}{\partial x}dx+\frac{\partial I}{\partial y}dy+\frac{\partial I}{\partial t}dt$$

化简，同除 dt，可得：

$$\frac{\partial I}{\partial x}dx+\frac{\partial I}{\partial y}dy+\frac{\partial I}{\partial t}dt=0 \iff \frac{\partial I}{\partial x}\frac{dx}{dt}+\frac{\partial I}{\partial y}\frac{dy}{dt}=-\frac{\partial I}{\partial t}$$

记 $\left(\frac{dx}{dt},\frac{dy}{dt}\right)=(u,v)$, 表示的即为所要求解的像素光流，$\left(\frac{\partial I}{\partial x},\frac{\partial I}{\partial y}\right)=(I_x,I_y)$ 为图像在该点处像素灰度在x和y方向上的梯度，I_t 为像素灰度随时间的变化量。写成矩阵形式：

$$\begin{bmatrix} I_x & I_y \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}=-I_t$$

这是一个二元一次方程，需要引入额外的约束求解。在LK光流中，引入假设3，空间一致性，某一个窗口内的像素具有相同的运动。

考虑一个$w\times w$大小的窗口，则：

$$\begin{bmatrix} I_x & I_y \end{bmatrix}_k\begin{bmatrix} u \\ v \end{bmatrix}=-I_t k,k=1,\dots,w^2$$

即：

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ I_{x3} & I_{y3} \\ \vdots & \vdots \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}=-\begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix} \iff A\vec{u}=b$$

这是关于 u,v 的超定线性方程，可用最小二乘法求解 $\vec{u}=\left(A^TA\right)^{-1}A^Tb$，也可以迭代求解。

KLT 算法

KLT 是基于光流原理的一种特征点**跟踪算法**，本质上也基于光流的三个假设，不同于前述直接比较像素点灰度值的作法，KLT 比较像素点周围的窗口像素，来寻找最相似的像素点。由光流假设，在很短时间 τ 内，用 J 和 I 表示前后两帧图像，满足：

$$J(A\mathrm{x}+d)=I(\mathrm{x}),\text{其中}\ A=1+D=1+\begin{bmatrix}d_{xx} & d_{xy}\\ d_{yx} & d_{yy}\end{bmatrix}=\begin{bmatrix}1+d_{xx} & d_{xy}\\ d_{yx} & 1+d_{yy}\end{bmatrix}$$

像素位移 (displacement) 向量满足仿射运动模型 (Affine Motion) $=Dx+d$，其中 D 称为变形矩阵 (Deformation Matrix)，d 称为位移向量 (Displacement Vector)。D 表示两个像素窗口块运动后的变形量，所以当窗口较小时，会比较难估计。通常 D 可以用来衡量两个像素窗口的相似度，即衡量特征点有没有漂移。而对于光流跟踪量，一般只考虑平移模型 (Translation Model)：

$$J(\mathrm{x}+d)=I(\mathrm{x})$$

为了普遍性，用仿射运动模型来推导 KLT 算法原理。在像素窗口下，构造误差函数：

$$\epsilon=\iint_W[J(A\mathrm{x}+d)-I(\mathrm{x})]^2w(\mathrm{x})d\mathrm{x}$$

其中 $w(x)$ 是权重函数，可定义为高斯形式，最简单的情况为 $w(x)=1$。为了最小化误差损失（上式），分别对变量 D 和 d 求导，并将结果置零：

$$\begin{cases}\frac{\partial\epsilon}{\partial D}=2\iint_W[J(A\mathrm{x}+d)-I(\mathrm{x})]g\,\mathrm{x}^T\,w\,d\mathrm{x} & =0\\ \frac{\partial\epsilon}{\partial d}=2\iint_W[J(A\mathrm{x}+d)-I(\mathrm{x})]g\,w\,d\mathrm{x} & =0\end{cases}$$

其中$g=\left(\frac{\partial J}{\partial x},\frac{\partial J}{\partial y}\right)^T$，表示灰度梯度。记光流 $u=Dx+d$，则对运动后的像素点进行泰勒展开：

$$J(A\mathrm{x}+d)=J(\mathrm{x})+g^T(u)$$

仿射运动模型 D 结果可见 [1] [5]，这里给出平移运动模型 d 结果。令 $D=0$：

$$J(\mathrm{x}+d)=J(\mathrm{x})+g^T(d)$$

令导数为零：

$$\begin{aligned}&\iint_W[J(A\mathrm{x}+d)-I(\mathrm{x})]g\,w\,d\mathrm{x}=\iint_W[J(\mathrm{x}+d)-I(\mathrm{x})]g\,w\,d\mathrm{x}=\iint_W[J(\mathrm{x})-I(\mathrm{x})+g^T(d)]g\,w\,d\mathrm{x}=0\\ \iff&\iint_W[J(\mathrm{x})-I(\mathrm{x})]g\,w\,d\mathrm{x}=-\iint_W g^T\,\mathrm{d}\,g\,w\,d\mathrm{x}=-\left[\iint_W g\,g^T\,w\,d\mathrm{x}\right]\mathrm{d}\\ \iff&e=-Z\mathrm{d}\\ \iff&Z\mathrm{d}=e\end{aligned}$$

其中 Z 是 2×2 矩阵，e 是 2×1 向量。这是线性方程组优化问题，当 Z 可逆时，这个方程可容易求解。因为推导过程用到了泰勒展开，所以只有当像素位移较小时，才成立。实际操作中，一般迭代式的来求解，每次用上次结果做初始化，进一步求解(In a Newton-Raphson Fasion)。

$$Z=\begin{bmatrix}g_x^2 & g_xg_y\\ g_xg_y & g_y^2\end{bmatrix}$$

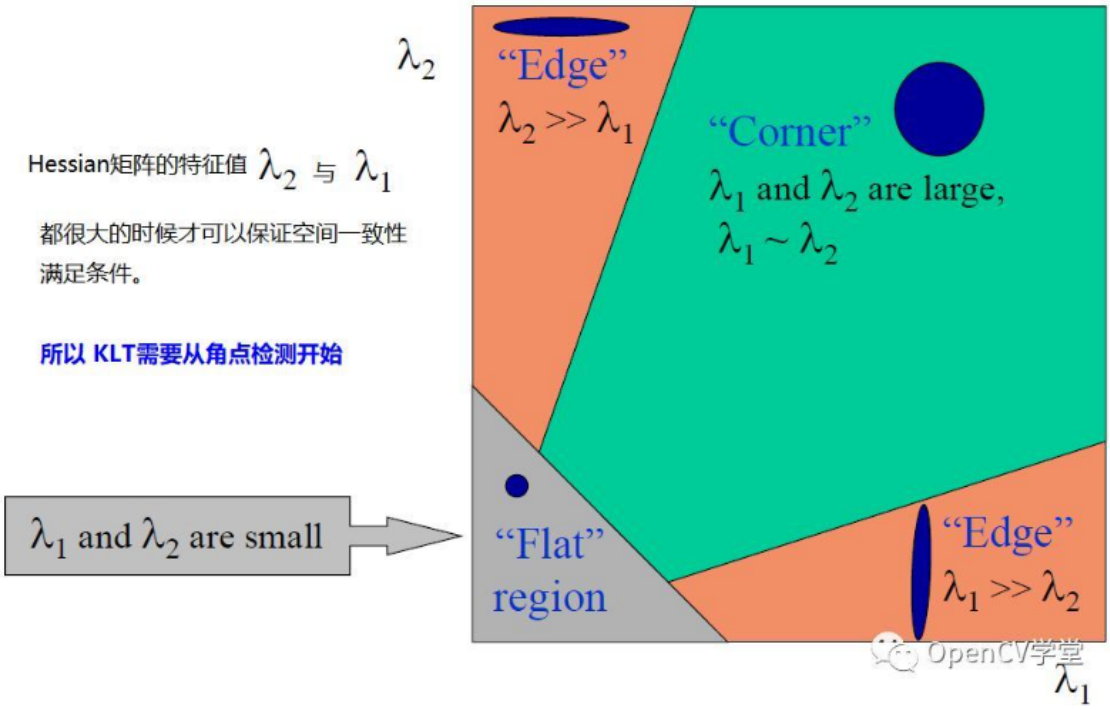
上式可以估计出位移向量d，显然这个估计是建立在窗口内的像素的是平滑(与粗糙相对）的，即其强度是一致的，而实际上这会在窗口内一些变化较大（如角点）的位置，造成其位移估计d偏差较大。然而可以通过迭代的方式，不断的应用新的 d 值，而图像块每次迭代都可以采用双线性插值（获得子像素精确度）得到新位置。
角点的选择问题：

最为重要，这个角点是根据 Z 矩阵的两个特征值来选择的，一方面两个特征值不能太小，排除噪声影响，另一方面两个特征值不能差别太大，说明这是角点。这里提出了下面的公式：

$$\min(\lambda_1, \lambda_2) > \lambda$$

特征点的选择：首先将最小特征值降序排列，优先选择最小特征值大的。为了防止窗口重叠，如果发生选择的特征点在先前选择特征点的窗口内，就删除。最后再通过能量偏差函数删除。

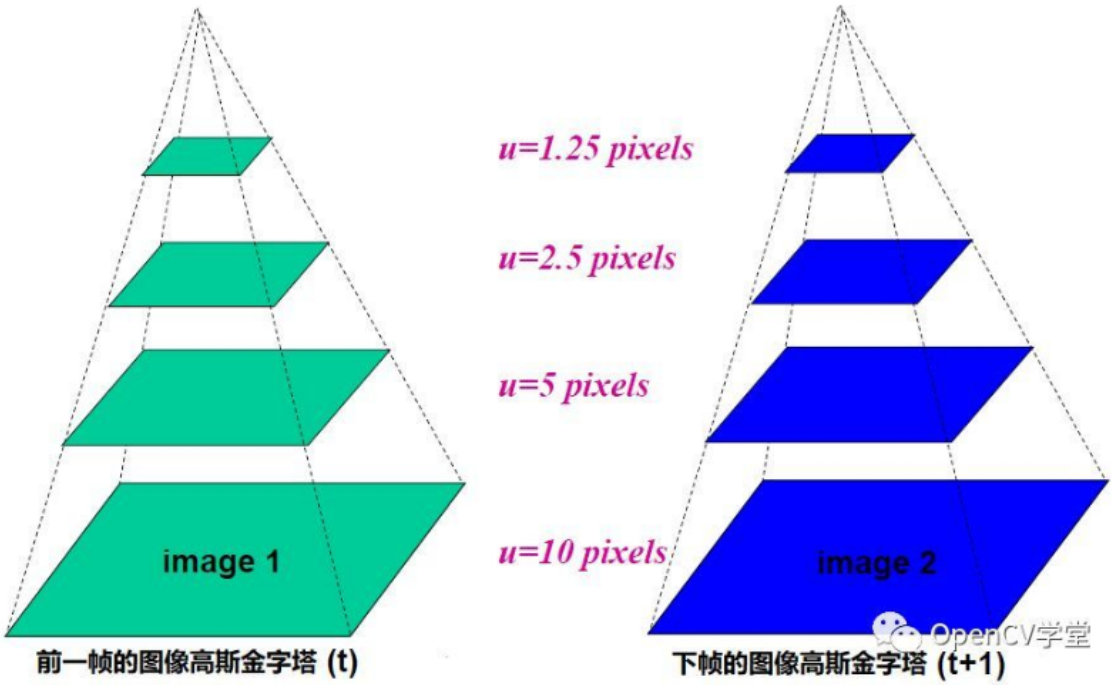
对于特征值的说明：对于矩阵 A，存在一个实数 Q，一个向量 X，使得 $AX = QX$，则矩阵的特征值为 Q。



金字塔 KLT

空间尺度不变性

通过建立每一帧的图像金字塔，实现尺度空间窗口目标对象搜索



https://zhuanlan.zhihu.com/p/73820608

SVO

半直接法

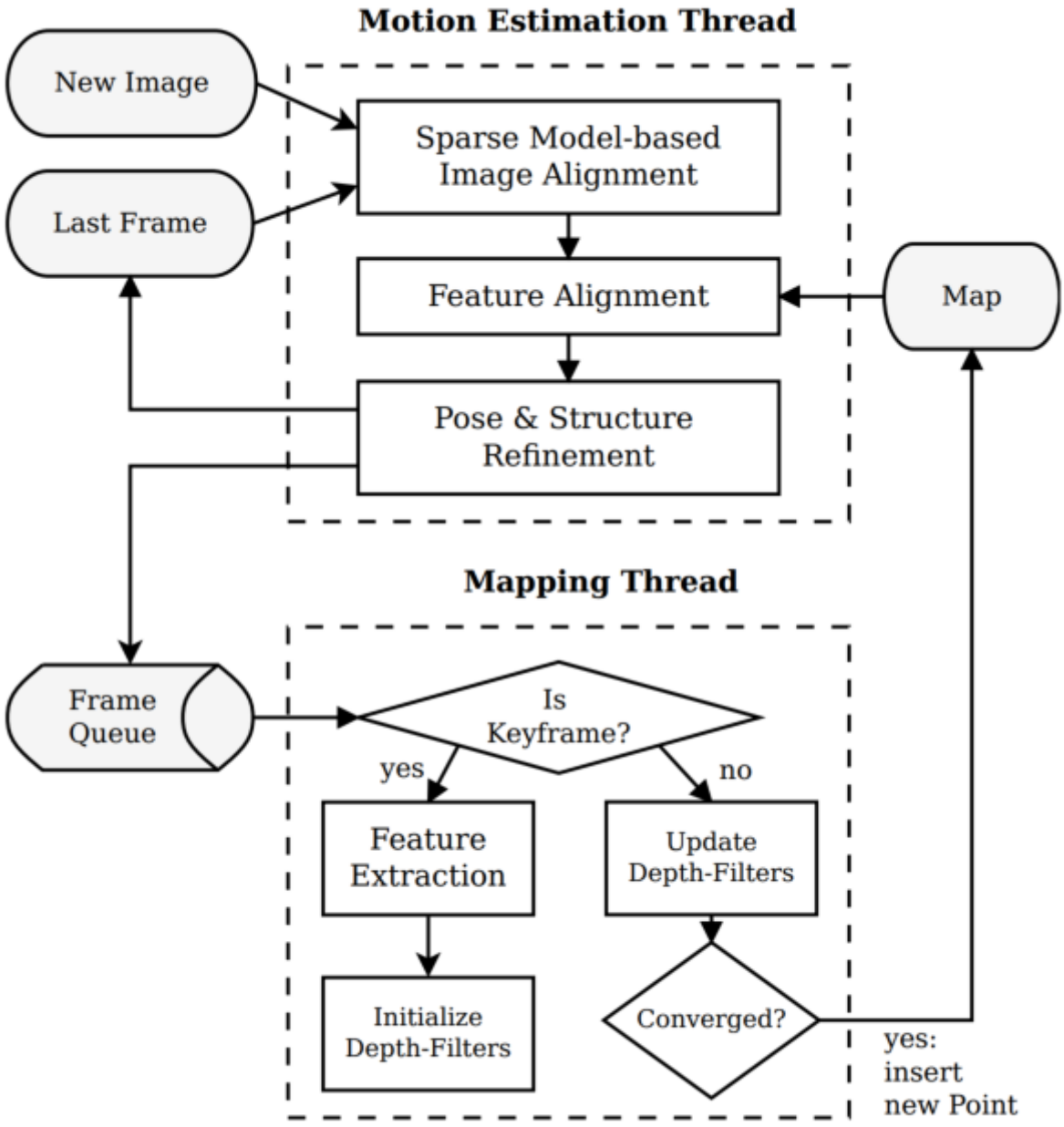


Fig. 1: Tracking and mapping pipeline

SVO中的光流法

目的：跟踪特征点，实现特征点匹配。

方法：金字塔 KLT

待补充，正在看

---

[1] Shi, Jianbo, and Carlo Tomasi. Good features to track. Cornell University, 1993.
[2] Birchfield, Stan. "Derivation of kanade-lucas-tomasi tracking equation." unpublished notes (1997).
[3] Bouguet, J.-Y.. “Pyramidal implementation of the lucas kanade feature tracker.” (2000).
[4] Suhr, Jae Kyu. "Kanade-lucas-tomasi (klt) feature tracker." Computer Vision (EEE6503) (2009): 9-18.
[5] Bouguet, Jean-Yves. "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm." Intel Corporation 5.1-10 (2001): 4.

[6] https://leijiezhang001.github.io/KLT/