Video Deblurring for Hand-held Cameras Using Patch-based Synthesis 论文总结

——金鹏

我们知道通过手持相机拍摄出来的视频，通常会因为拍摄过程中相机的抖动而导致很多帧出现模糊的现象，因此往往需要在拍摄后进行去模糊的工作。

一般的图像去模糊方法大部分都是依据某种和噪音降质的观测来估计原来的图像，将图像进行复原的一般方法是反卷积，那么这种方法最重要的就是得到模糊函数的所用的模糊核。而我们的视频去模糊与图像去模糊又有些不一样，视频模糊的模糊核会受到时间和空间的影响而不断的发生改变。因此一般的图像去模糊方法在这里效果并不明显，故 Video Deblurring for Hand-held Cameras Using Patch-based Synthesis 这篇论文里提出了一种新的去模糊方法，即通过视频的连续性，在视频中某一帧模糊掉的部分可以通过另外的一帧里相同的清晰的部分进行替换就可以得到清晰的图像。下面是这个算法的基本思想：

首先，我们将每一帧基于单应性矩阵的近似结果看成是真是的运动。然后我们用这个近似的结果来获得帧上每个像素的 luckiness 值，luckiness 值是用来进行清晰度的对比。去模糊的过程主要是我们在进行去模糊的帧的附近搜索清晰的块区域来代替过去模糊的块区域。我们估算的单应性矩阵来获得帧上相应的像素，然后通过最佳匹配图块的局部搜索算法来弥补误差。而对于清晰块与模糊块的对比，我们是用卷积的方法来讲清晰的块模糊化后再与模糊的块进行对比。最后，利用在连续帧对应的块的相似度约束来维护去模糊后的帧的时间相干性。

下面介绍下 KLT 算法，我们最初的单应性矩阵的计算就依靠于 KLT 算法找出的特征轨迹。

首先，KLT 算法要满足三个前提假设：

1）亮度恒定

2）时间连续或者是运动是“小运动”

3）空间一致，临近点有相似运动，保持相邻

如果我们要判断一个视频的相邻帧 I，J 在某局部窗口 W 内是一致的，则在窗口中有：$I(x , y , t) = J(x' , y' .t + \tau)$，亮度恒定的假设就是为了保证等号两边不受到亮度的影响，而时间连续或者是运动是“小运动”则是为了保证算法能够找到点，而空间一致则是为了：

在窗口 W 上，所有(x, y)都往一个方向移动了(dx, dy)，从而得到(x', y')，即 t 时刻的(x, y)点在 t+ τ 时刻为(x+dx, y+dy)，所以寻求匹配的问题可化为对以下的式子寻求最小值，或叫做最小化以下式子：

$$\varepsilon(dx,dy)=\sum_{x=ux-wx}^{ux+wx}\sum_{y=uy-wy}^{uy+wy}(I(x,y)-J(x+dx,y+dy))^2$$

用积分表示可等效的表示为：

$$\varepsilon=\iint_W[J(x+d/2)-I(x-d/2)]^2\,w(x)dx$$

若该式子要去的最小值，这个极值点的导数一定为 0，即

$$\frac{\partial\varepsilon}{\partial d}=2\iint_W[J(x+d/2)-J(x-d/2)][\partial J(x+d/2)/\partial d-\partial I(x-d/2)/\partial d]w(x)dx$$

这个式子值为 0，我们将 J 进行泰勒展开就可以得到：

$$J(\xi)\approx J(a)+(\xi_x-a_x)\frac{\partial J}{\partial x}(a)+(\xi_y-a_y)\frac{\partial J}{\partial y}(a)$$

然后我们可以得到：

$$J(x+d/2) \approx J(x) + \frac{dx}{2}\frac{\partial J}{\partial x}(x) + \frac{dy}{2}\frac{\partial J}{\partial y}(x)$$
$$I(x-d/2) \approx I(x) - \frac{dx}{2}\frac{\partial I}{\partial x}(x) - \frac{dy}{2}\frac{\partial I}{\partial y}(x)$$

于是问题就可以转化为：

$$\frac{\partial \varepsilon}{\partial d} = 2\int\int_W [J(x+d/2) - I(x-d/2)][\partial J(x+d/2)/\partial d - \partial I(x-d/2)/\partial d]w(x)dx$$
$$\approx \int\int_W [J(x) - I(x) + g^T d]g(x)w(x)dx$$

其中：

$$g = [\frac{\partial}{\partial x}(\frac{I+J}{2}) \quad \frac{\partial}{\partial y}(\frac{I+J}{2})]^T$$

从而问题即为：

$$\frac{\partial \varepsilon}{\partial d} = \int\int_W [J(x) - I(x) + g^T(x)d]g(x)w(x)dx = 0$$
$$\int\int_W [J(x) - I(x)]g(x)w(x)dx = -\int\int_W g^T(x)dg(x)w(x)dx = -[\int\int_w [g(x)g^T(x)w(x)]d$$

可将其简单的看成是 Zd=e，其中 Z 为一个 2*2 的矩阵 ，e 为一个 2*1 的向量，

$$Z = \int\int_w g(x)g^T(x)w(x)dx$$
$$e = \int\int_W [I(x) - J(x)]g(x)w(x)dx$$

为了要使 d 能够得到解，则 Z 需要满足条件，即 Z*Z'矩阵可逆，其中 Z'为 Z 矩阵的转置(ZT)，在一般情况下，角点具有这样的特点。Opencv 中可以用 cvGoodFeaturesToTrack 来找角点，用 cvCalcOpticalFlowPyrLK 来实施追踪。

这样我们就可以通过 KLT 来获得特征轨迹，通过特征轨迹我们就可以得到相应的单应性矩阵，Opencv 中可以使用 cvFindHomography 来计算单应性矩阵。

在我们接下来的算法中用到了块匹配算法，所以我们这里先接受一下块匹配算法:

1) 块匹配运动估计的基本原理:

运动估计的基本思想是将图像序列的每一帧分成许多互不重叠的宏块，并认为宏块内所有像素的位移量都相同，然后对于当前帧中的每一块到前一帧某给定搜索范围内根据一定的匹配准则找出与当前块最相似的块，即匹配块，由匹配块与当前块的相对位置计算出运动位移，所得运动位移即为当前块的运动矢量。示意图如下：

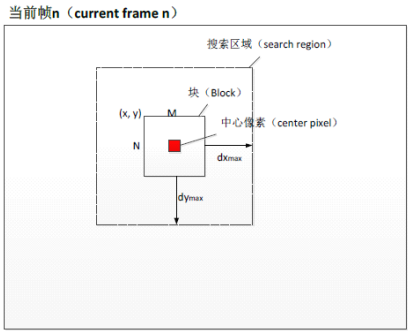


图 1

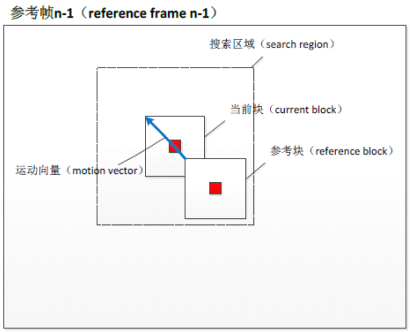


图 2

宏块大小为 M×N，一般取 M=N=16、8 或4。搜索范围一般由最大偏移矢量来决定，设可能的最大偏移矢量为(dxmax,dymax) 则搜索范围为(M+2*dxmax)*(N+2*dymax)。常用搜索最佳匹配准则有绝对误差和(Sum of Absolute Difference，SAD)、平均绝对误差(Mean Absolute Difference, MAD)、方差和(Sum of Sequence Error, SSE)、均方差(Mean Sequence Error, MSE)、绝对变化误差和(Sum of Absolute Transformed Difference , SATD)和归化互相关函数(NCCF)。最基本的匹配方法是全搜索匹配法，该方法对搜索区内的所有子块进行搜索,因此能够得到搜索区内与当前块最为匹配的块.其中前向水平、垂直搜索窗口高度和反向水平、垂直搜索窗口高度应当根据具体图像特征进行选择.如果图像变化不大,则为了提高搜索速度,可以用小的搜索窗口;反之,如果处理的序列有很多的激烈场面,帧与帧之间的变化很大,则必须用大的窗口进行搜索,以达到要求的。但由于全搜索法需要对搜索窗口内每个宏块进行匹配,运算量非常大，因此我们一般采用起的匹配方法这里介绍一种叫新三步法的块匹配算法：

2) 新三步法：

step1：新三步法首先建一个以起始点为中心，包括最大搜索范围的窗口。计算如图3(a)所示8 个步长等于或者略大于最大搜索范围的一半的点，8 个相邻点和起始点的匹配误差。

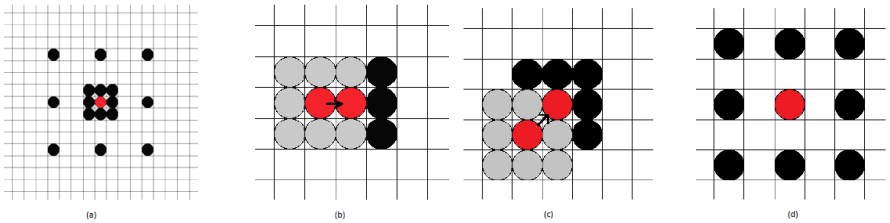
A. 如果匹配误差最小的点是起始点，以起始点作为搜索结果，搜索结束。

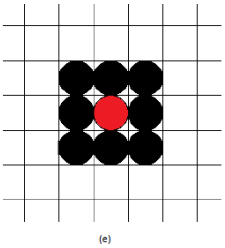
B. 如果匹配误差最小的点是起始点周围相邻的点，以这个点为中心，找出如图3(b)所示的3 个黑色点或如3(c)所示的5 个黑色点的匹配误差。此时的匹配误差最小的点即为最终搜索结果。

C. 否则，跳到第二步。

step2：以第一步中匹配误差最小的点为中心，步长为原来一半，计算如图3(d)所示的9 个点的匹配误差。

step3：重复第二步直到步长为1。从如图3(e)所示的9 个点中找出最小匹配误差点，即为最终搜索结果。





(e)

接下面我们来介绍去模糊算法的具体实现：

我们通过单应性矩阵来近似模糊的视频帧，我们假设相机的工作后期为2τ，帧 f_i 的曝光时间间隔为[t_i-τ, t_i+τ]，令时间间隔 T= t_i- t_{i+1}， f_i 的内在图像为 l_i，相邻帧的运动可用单应性矩阵来进估算，即 $l_{i+1}=\hat{H}_i(l_i)$，这里的 $\hat{H}_i$ 是一个参数化的单应性矩阵，由于相邻帧之间的运动速度是恒定的则：

$$f_i = \frac{1}{2\tau}\int_{t=0}^{\tau}\left(\hat{H}_{i-1}^{t}(l_i)+\hat{H}_i^{t}(l_i)\right)dt,$$

其中：

$$H_{i-1}^{t}=\frac{T-t}{T}I+\frac{t}{T}H_{i-1}^{-1},\ H_i^{t}=\frac{T-t}{T}I+\frac{t}{T}H_i.$$

其中 $\hat{H}_{i-1}^{t}$ 和 $\hat{H}_i^{t}$ 为 warping functions，I 是3*3的单位矩阵，H_{i-1}^{-1} 是 H_{i-1} 的转置。将等式离散化就可以得到：

$$f_i \doteq b_i(l_i)=\frac{1}{1+2\tau}\left[\sum_{d=1}^{\tau}\left(\hat{H}_{i-1}^{d}(l_i)+\hat{H}_i^{d}(l_i)\right)+l_i\right]$$

这里 T 为 $[t_i, t_{i+1}]$ 内的采样率，我们将其固定为20，τ则是工作期间内的采样数。bi 则为帧 i 的模糊函数。这样我们就得到了一个关于当前帧 fi 与内在帧 li 的一个关系等式。在这个式子中我们需要得到两个变量的值：H_i 和τ，其中 H_i 可以用 KLT 算法来找到特征轨迹，然后应用追踪点来计算单应性矩阵。在算法那种不仅估算相邻两帧之间的单应性矩阵，还要估算局部时间窗口 $W_i=[i-M, i+M]$ 内的任意两帧之间的单应性矩阵，其中 M=5。

下面我们介绍下关于视频中每个像素的 luckiness 的定义，

$$\alpha_i(x) = \exp\left(-\frac{\|\tilde{H}_{i-1}^{-1}(x)-x\|^2+\|\tilde{H}_i(x)-x\|^2}{2\sigma_l^2}\right)$$

其中 $\tilde{H}$ 是基于单应性矩阵的映射函数，σ_l 是一个常量值，这里我们设置为20。这个公式计算的是当前帧 f_i 中的像素 x 与前面一帧 f_{i-1} 和后一帧 f_{i+1} 的对应像素平方差的和，我们知道若帧与帧之间移动较小那么相应的单应性矩阵就会近似为单位矩阵，相应的 $\alpha_i(x)$ 的值

也就近似为1，否则反之。一帧整体的 luckniess 可以定位为帧上所有像素点 luckniess 的平均值。

为了计算τ，我们首先选取一系列的帧对，这些帧对的 luckniess 有着较大的差值， 这样我们就要有效的测试模糊函数的正确性。这里令 $(f_i^k,\ f_j^k)$，其中 K=1,2,3....K，$j \in W_i$ 并且帧 i 和 j 的 luckiness 差值 α_i-α_j 必须大于一个阀值（这个阀值的值为 max(α_j), $j \in W_i$）。通过将下面的式子最小化去求解τ：

$$E(\tau) \quad = \quad \sum_{k=1}^{K} \left\| b_j^k \left(\hat{H}_{ij}^k (f_i^k) \right) - f_j^k \right\|^2$$

我们可以看出若模糊函数 b 正确，则式子值较小否则反之。因为τ的值取值范围在1到[T/2]之间，故我们使用暴力搜索的方式来确定τ。

当我们确定了模糊函数 b 之后，就可以进行接下来的去模糊操作。首先，我们说明几个变量的含义，$f_{i,x}$ 表示一块位于帧 f_i 上的以坐标 x 为中心的 n*n 的图像块，其中 n=11，然后我们通过对以帧 f_i 周围的帧集合 W_i 内的清晰块进行加权平均的方式来获得 $f_{i,x}$ 的潜在图像块 $l_{i,x}$：

$$l_{i,x} \quad = \quad \frac{1}{Z} \sum_{(j,y)\in\Omega_{i,x}} w(i,x,j,y) f_{j,y}, \qquad (1)$$

其中 $f_{j,y}$ 是帧 f_j 经过单应性矩阵变换 $\hat{H}_{ji}(f_j)$ 后在坐标点 y 上的图像块，而

$$w(i,x,j,y) \quad = \quad \exp\left(-\frac{\|b_{j,y} - f_{i,x}\|^2}{2\sigma_w^2} \right)$$

$\omega(i,x,j,y)$ 是一个权重函数，其中 $b_{j,y}$ 是在帧 f_j 经过单应性变换以及模糊化后即 $b_i(\hat{H}_{ji}(f_j))$ 上的以 y 为坐标中点的图像块，σ_w 是一个常量值为0.1。我们可以从这个公式看出当帧 f_j 经过映射以及模糊化后位于 x 坐标上的像素值如果与帧 f_i 在 x 坐标上的像素值非常接近，则说明 f_j 在 x 坐标上的像素值为 f_i 去模糊后的像素值的可能性较大，故所占权值较大，否则反之。而 Z 为标准化因子 $Z = \sum_{(j,y)\in\Omega_{i,x}} w(i,x,j,y)$。$\Omega_{i,x}$ 是所有在块 $f_{i,x}$ 附近并且和 $f_{i,x}$ 相匹配的块的集合。

为了确定块集合 $\Omega_{i,x}$，我们从当前帧的 W_i 中选取 N 个最佳匹配块，用 $w(i,x,j,y)$ 来表示匹配程度，也就相当于求解公式 $\arg\min_{j,y} \| b_{j,y} - f_{i,x} \|^2$ 来获取匹配块。

为了能够获得精确的匹配块，我们将在对应帧中以 x 为中心，以 m*m 为窗口来进行搜索，其中 m=21以保证搜索窗口够大，能够准确的找到匹配块。当我们找到相应的块后可以进行块的去模糊作。我们可以简单的通过公式（1）来直接获得 f_i 的内在图像 I_i，但是这么获得的 I_i 上的像素点并不具有空间相干性，容易出现断裂，因此我们在求解 I_i 对应像素点的时候需将其与领域的像素相关联。我们通过下面的式子来计算 I_i 上对应的像素点：

$$\begin{aligned} l_i(x) &= \frac{1}{Z}\sum_{x'\in\Omega_x} Z_{x'} l_{i,x'}(x), \\ &= \frac{1}{Z}\sum_{x'\in\Omega_x}\sum_{(j,y)\in\Omega_{i,x'}} w(i,x',j,y) f_{j,y}(x) \end{aligned}$$

这里 Ω_x 是像素点的集合，这些像素点都是 f_i 位于以 x 为中心的 n*n 的窗口中的像素点。$I_{i,x'}(x)$ 表示帧 f_i 上以 x’为中心的图像块上的坐标 x 的像素值，$Z_{x'} = \sum_{(j,y)\in\Omega_{i,x'}} \omega(i,x',j,y)$ 而 Z 为归一化因子故 Z=$\sum_{x'\in\Omega_x} Z_{x'}$ 。那么我们计算像素点 x 的时候就需要用到 n^2个匹配块计算量较大，因此为了加速我们的去模糊操作，我们可以只对稀疏的规则网格上的像素点进行去模糊工作。

为了得到更好的去模糊效果，我们这里修改一下权重函数为：

$$\omega'(i,x,j,y) = \omega(i,x,j,y) \bullet \exp\left(-\frac{\left\|1-\alpha_{j,y}\right\|^2}{2\gamma^2}\right)$$

这里 $\alpha_{j,y}$ 是一个以 y 为中心的 n*n 的像素块每个像素点所对应的 luckiness。1是一个 n*n 的值全为1 的矩阵。当块越清晰的时候其像素所对应的 luckiness 值越接近于1，否则反之，这样在新的权重函数中，越清晰的块所占权重越大，因此可以获得更好的去模糊效果。

同样我们会依据每幅图的 luckiness 来决定视频每一帧的处理顺序，首先我们先根据 luckiness 来对帧序列进行排序，然后从 luckiness 值最接近于1的帧开始处理，这样可以使得块替换的顺序得到最优化，像 RGB 一样，我们将 luckiness 值作为帧上每个像素点的一个通道每次处理都会不断的更新 luckiness 值。

在去模糊之后，为了进一步提高去模糊效果，我们将对去模糊后的帧进行迭代优化，我们修改一下以前的公式为：

$$l_{i,x}^{p+1} = \frac{1}{Z}\sum_{(j,y)\in\Omega_{i,x}} w'(i,x,j,y) l_{j,y}^{p},$$

$$w'(i,x,j,y) = \exp\left(-\frac{\|b_{j,y}^{p} - f_{i,x}\|^2}{2\sigma_w^2}\right)\exp\left(-\frac{\|\mathbf{1}-\alpha_{j,y}^{p}\|^2}{2\gamma^2}\right)$$

其中 p 代表的是经过 p 次迭代后的结果。这样经过一定次数的迭代就可以进一步提高去模糊的效果。那么我们如何来确定迭代的次数呢，这个迭代次数我们可以依据时间窗口的大小 2M+1来计算得出，首先我们计算输入视频的每一帧的 luckiness，并挑选出 luckiness 值大于

luckiness 最大值的0.85的帧作为清晰帧，然后计算出相邻两帧的距离最大值 M_s，然后我们的迭代次数就可以确定为[(M_s -1)/2M].

以上方法的时间相干性不强，因此我们在每次迭代的过程中加一点限制来提高时间相干性，在每次迭代优化中，我们将时间窗口约束为 W_i=[i-1,i+1] ，使用 I^p_{i-1,x_i-1}，I^p_{i,x_i}，I^p_{i+1,x_i+1} 这三帧来计算 I^{p+1}_{i,x_i} 。这样我们就可以增加其时间相干性。

以上就是我们去模糊算法的主要过程。