2021.12.05周报

纪新龙

本周计划任务

转入阅读论文，至少读三篇相关的论文。

具体完成情况

完成

下周计划任务

继续阅读论文，至少读三篇相关的论文。

具体完成情况

考虑到最终要分解的是视频，阅读论文的时候没有仔细研究视频和图片的不同，这周先是阅读了丁守鸿引文中的两篇关于视频本征分解的论文，

- 7_Ye2014_Intrinsic_Video_and_Applications
- 8_Bonneel2014_Interactive_Intrinsic_Video_Editing

然后第三篇仍然想读关于视频的，就到张磊老师之前推荐的3_MEKA2021_Real_Time_Global_Illumination_Decomposition_of_Videos的“相关工作”部分找到一段描述：

> Bonneel et al. [2015] and Meka et al. [2016] employ temporal consistency priors, and Bonneel et al. [2014] and Ye et al.[2014] use an optical-flow based consistency constraint... Among these methods, only Bonneel et al. [2014] and Meka et al. [2016] can perform more than one decomposition per second, with the latter achieving real-time frame rates.

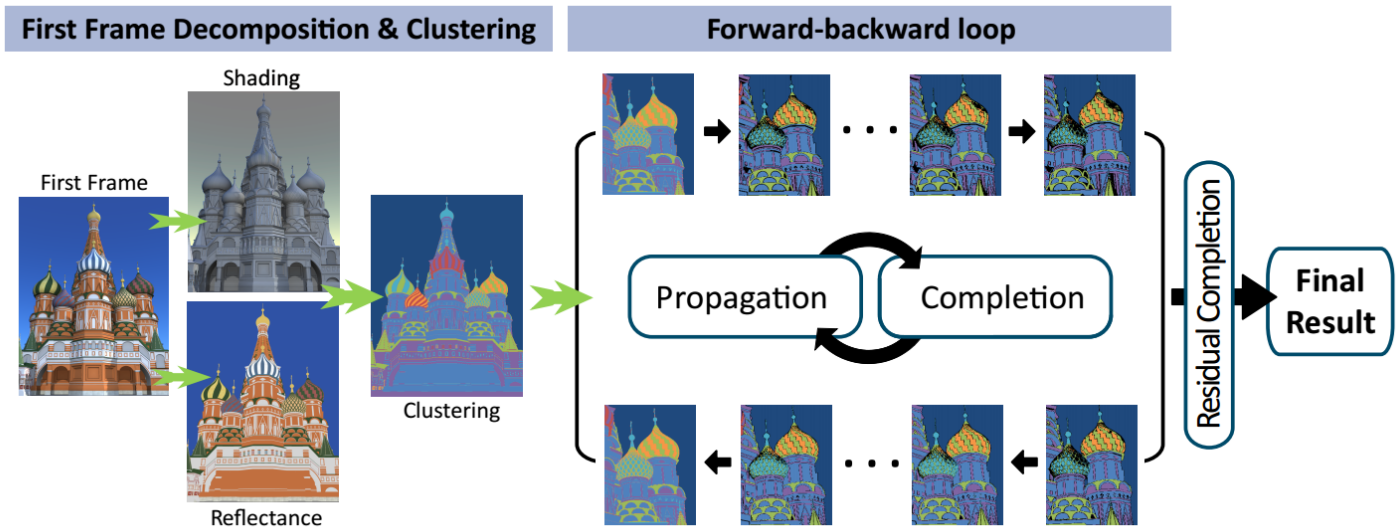
意思是我读的前两篇都用了光流法。除此以外还有另外第三篇Meka et al. [2016] 和Bonneel et al. [2015]一样都使用了时间一致性先验，并且都能在一秒内分解超过一帧图像，且只有Meka et al. [2016]是能真正实时分解的。于是读了Meka et al. [2016]

- 9_Meka2016_Live_Intrinsic_Video

本周读的三篇论文，分解速度上Meka2016>Bonneel2014>Ye2014，最终Meka2016达到了在线分解的效果。前几周都在复现论文，亲自实现代码并观察体悟各个约束项的初衷和效果。本周阅读论文的时候再次验证了心中积累的一些想法。

7_Ye2014_Intrinsic_Video_and_Applications

这篇论文第一个提出能够保证帧与帧之间分解效果一致性的视频本征分解算法。算法架构也和其他常见算法不同。**思维很缜密，具有首创性，但是耗费时间极其长，需要用户交互，基于聚类**。第一作者是清华大学的。下图是该算法的流程图：



- 第一帧：
  - 用户的笔画画在色度变化处。用 [Li et al. 2004]方法将用户的笔画扩展成一个掩膜，取掩膜内像素的色度梯度平均值为色度梯度阈值。
  - 用Zhao et al. [2012]的方法，但是采用上一布中得到的色度梯度阈值，对第一帧图像本征分解。
  - 用[Felzenszwalb and Huttenlocher 2004]的方法粗略分割得到的反射率层。
  - 手动判断各个分割小块之间的RGB平均值的差距，将差距小于阈值的块合并。聚类完毕。
  - 若一类中有色度相同但是反射率不同的点，一律取反射率为取反射率平均值。
- 从第二帧开始：反射率的传播
  - 用光流法追溯k帧中某点p在k-1帧中对应的点坐标p'。

    - 以p'为中心画一个窗口，从颜色上计算p属于该窗口中的各个聚合的概率。某聚合在窗口中占有点数越多$P(p|C_k)$，聚合与p'的最小距离越小$P(C_k)$，这个聚合收纳点p越有优势越大。
      附加条件：概率最大的聚合包含这个点的概率是否大于一个阈值；此概率是否大于第二大可能性聚合的两倍。
      若$P(C_k|p)$满足附加条件，直接判定点p属于C_k，否则将其标记为黑色待定，继续进行下一帧的传播。随着传播，黑色待定的区域越来越大。
    - 停止传播：设定一个窗口大小，当图中任意一个窗口的待定像素占比超过80%，停止传播。要在这一帧对黑色区域进行修复。
        - 补丁、反射率估计
          首先对黑色区域向外扩展4个像素点宽度，针对扩展后的补丁样式的区域（扩展的边沿+原先的黑色内部）进行反射率估计。估计的主体方法是Zhao et al. [2012]，由于扩展了4个像素点宽度的边沿，要在边沿区域加一个新估计出的反射率和原先反射率之差的平方能量项约束；为了保证补丁区域的时间一致性，对补丁内部进行梯度的泊松约束。
        - 聚合归属
          参照之前的聚合归类步骤，但仅仅考虑$P(p|C_k)$，且不从颜色上考虑而从反射率上考虑。简单的将该帧所有黑色区域的反射率像素点归入各个聚合。
    - 反向传播
      仅仅考虑上一步中估计出的补丁区域，反向传播，将之前的帧中的黑色点分配到具体聚合。直到开头第一帧。
      倘若到第一帧还有黑色区域，就减小用于检测黑色区域占比的窗口大小（因为一直在优化着），继续正向传播。正向-反向 总共三次。
- 倘若三次传播迭代都没能将所以黑色区域清除，直接用“光照层的平滑性”这个约束强行估计黑色区域。算法到此完成。

这不是一个在线算法。可以看出设计得很严密，每一步反射率的估计和聚合的归类都小心翼翼，最后仍有无法归类的区域才强制用光照平滑来确定。同时，正向、反向传播这些设计都很容易理解，以及针对补丁区域的反射率估计，很形象。**最有特点的还是整个算法基于对第一帧的反射率层聚类后的“传播”，也就是说后续的所有帧的反射率其实都是从第一帧的聚类结果中传播开的，高度依赖聚类的思想，利用传播的思想实现时间上的一致性。而传统的retinex变法方法仅仅作为修修补补的工具和第一帧反射率的初始化预测。** 这种做法是其他方法没有的。用到很多细碎的方法，比较耗时。

8_Bonneel2014_Interactive_Intrinsic_Video_Editing

第二篇的算法设计回归到各个能量项的风格上，但是是先计算出梯度，然后在梯度图上重建出光照，再根据光照求反射率。也很有技巧。为了相比较前一种算法大大提速，算法处处体现了对计算时间上的考虑。
特点：

- 计算梯度、重建图像 分离，预计算梯度查找表加速。

- 多尺度金字塔来加速重建。
- 重建的是光照而不是反射率。

一种快速分离图像梯度的算法

平滑的照明和稀疏的反射梯度使用混合l 2 - l p优化，可以使用预先计算的查找表非常有效地解决。r和s都是梯度。

$$E(s,r) = \sum_x \|i - s - r\|^2 + \lambda_s \|s\|^2 + \lambda_r \|r\|^p \qquad (2)$$

上图就是能量项，很常见，分别是数据保真、光照平滑、反射率稀疏。将s看成自变量，对s求导令等于0，可以得到

$$s = \frac{i - r}{1 + \lambda_s}$$

带入后消去s，此时只剩下r一个自变量了，但是r的头上还有一个小于2的p。我们用变权重的最小二乘法[Bjorck 1996 ¨ , x 4.5]来迭代求解，问题变成：

$$\sum_x \frac{\lambda_s}{1 + \lambda_s} \|i - \tilde{r}_{k+1}\|^2 + \lambda_r w_{k+1} \|\tilde{r}_{k+1}\|^2 \qquad (5a)$$

$$\text{with } w_{k+1} = \frac{p}{2} |\tilde{r}_k|^{p-2} \qquad (5b)$$

对r求导令等于0，最有解：

$$\tilde{r}_{k+1} = \frac{2\lambda_s i}{2\lambda_s + \lambda_r (1 + \lambda_s) p |\tilde{r}_k|^{p-2}} \qquad (6)$$

这个式子需要自行迭代600这样方可逼近最优解。确定了参数之后，每次迭代其实是一种确定的映射关系，这种映射可以用查表替代，即根据r_k要能查到r_{k+1}的值。这张表预先计算好，不计入分解时间内。这就是所谓的“快速”。

根据梯度重建光照-反射率

提出了一种技术从这些梯度重建反射率和照明，同时加强空间和时间平滑使用快速多尺度并行求解器能产生高质量的，时间一致的视频分解。
这是根据梯度重建光照的总的能量公式，S'_t是t时刻的log后的光照层。**之所以重建光照层而不是反射率层，是因为光照层是稠密的梯度值。而反射率层是数值大但是稀疏的梯度，相比之下重建光照层更加准确。**

$$E(S'_t) = E_p(S'_t) + \lambda_{nl} E_{nl}(S'_t) + \lambda_t E_t(S'_t), \qquad (10)$$

- 第一项是S'_t的偏导与已有梯度之间差值的平方，约束待估计的光照服从已知的梯度变化。这就把梯度用上了。
- 第二项是空间反射率约束。先用k-means按照色度对图像聚类，然后选出中心色度和S'_t的色度最接近的两个聚合，计算S'_t的反射率和这两个聚合中心的反射率的差的平方和。

$$
\begin{aligned}
E_{nl}(S'_t) &= \sum_x \sum_{n=1}^{N} w(x, x_n) \| \log R_t(x) - \log R_t(x_n) \|^2 \\
&= \sum_x \sum_{n=1}^{N} w(x, x_n) \| S'_t(x) - S'_t(x_n) \\
&\qquad - \log I_t(x) + \log I_t(x_n) \|^2, \qquad (8)
\end{aligned}
$$

- 第三项是**视频时间上的一致性约束**。根据光流法得到当前帧中的点再前一帧中的对应点，然后计算两点之间反射率的平方差作为约束项。这是防止帧与帧之间反射率的估计太不一致。

$$
\begin{aligned}
E_{t+1}(S'_{t+1}) &= \sum_x \| \log R_{t+1}(x_{t+1}) - \log R_t(x_t) \|^2 \qquad (9) \\
&= \sum_x \| S'_{t+1}(x_{t+1}) - S'_t(x_t) - \log I_{t+1}(x_{t+1}) \\
&\quad + \log I_t(x_t) \|^2.
\end{aligned}
$$

解上述表达式用的是Jacobi迭代，但是图像太大耗费时间。此处引入多层次方法，先构建一个金字塔，每一层都配置上述约束，然后在最顶层重建出光照层，将最顶层的光照层作为次顶层Jacobi迭代的初始值，解出次顶层的光照层...反复这样，最终解出分辨率最高的底层真实图像的光照图。

用户涂鸦

通过互动反馈，以及一种传播这种信息的技术在空间和时间上的投入来限制互动的数量用户需要做什么。用户笔画表示反射率层或者光照层的梯度为0，这种笔画会在时间（PCA）、空间上扩展（光流法）。用户涂鸦可以在觉得效果不好的时候随时加入，这和前一种方法只需要在第一帧涂鸦的做法不同。

光流和look up表的建立都被作者作为预处理，不算入分解时间。耗时：0.25-0.50s to process a 0.5MP video fr

尽管如此，这是一种更快的算法，但不是一种真正能够在线运行的算法。因为光流法很耗费时间，作者却把它作为预处理，look up表受参数影响，作者也直接把它的生成放入预处理。下图是分解一个合成场景的效果。

(a) Input video frame (b) Ground truth (c) [Zhao et al. 2012] (no user strokes) (d) Our result (no user strokes)

9_Meka2016_Live_Intrinsic_Video

此算法速度最快，风格完全回归到能量项组合上。特点是，作者似乎对各个约束的作用有很清晰独到的认识，这帮助他构建了比较细致有效的能量项组合。与前两个不同，不需要用户涂鸦。
这篇论文和前边细读过且作过报告的：
3_MEKA2021_Real_Time_Global_Illumination_Decomposition_of_Videos
是同一作者，所以有很多思想上和公式构建上的沿用。

对色度的认识：

单色白光假设下，反射率的色度不会随着光照而改变，即：原图中像素的色度就是该点反射率三通道计算出的色度。但是在光照强的地方此结论才比较准确，在光照弱的地方不太成立。

公式：

- 数据保真项
- 本地约束
  - 反射率分段常数
  - 光照平滑：通过系数，在亮度大的地方、色度边界处强调此项
  - 色度近似先验：在亮度大的地方，估计出的反射率计算出的色度应当接近原像素色度
- 空间-时间 反射率一致性约束
  对于当前帧的某一点x，若从前面几帧中随机采样出的几个点y_i的色度和x相近（小于某阈值），则计算它们反射率之差的平方。
- 反射率聚类先验
  先用灰度直方图方法聚类，然后针对亮度比较小的区域的点，计算点x和它所在聚合的代表反射率之差的平方。此约束有助于提振亮度小的区域的点。

加速

- 针对lp项的求解，仍然使用了变权重最小二乘法IRLS，但是是在GPU上实现的
- 针对一些局部约束项，作者对图像进行了分割切块，在这些小块上进行局部约束项的GPU优化
- 为了抑制低频误差，采用了多层金字塔结构。不同于前一种方法采用金字塔主要是为了加速。