数学实验第五次实验报告 video stablilation

PB20000264 韩昊羽

一.实验要求

- 阅读论文《Bundled Camera Paths for Video Stabilization》，了解相机稳定的原理，理解文章中新提出的划分cell，对每个cell作稳定的思想。
- 根据助教提供的框架，结合文章中求相机path的部分补全代码
- 更改参数，测试
- 用测试样例和自己录制的程序体现效果

环境配置

- matlab R2021a

二.实验原理

step 1

捕捉特征点，需要得到特征点对　　。

step 2

有了特征点之后，我们将 表达为cell内四个顶点的加权和，设　　是第 时刻cell的四个顶点，那么 可以被表达为　　。我们总是希望当到了第　时刻的时候， 的对应点 依然可以被表述为加权和。设第　时刻的cell变成了$\hat{\quad}\ \hat{\quad}\ \hat{\quad}\ \hat{\quad}\ \hat{\quad}$
据此我们定义能量：

$\hat{\quad}\qquad\hat{\quad}$

本来这样就已经足够了，但文章中提出了一种改进方法：我不只希望它们尽量接近，我还希望cell本身不发生一个比较大的形变，于是引入另一种能量:

$$E_s(\hat{V})=\sum_{\hat{v}}\|\hat{v}-\hat{v}_1-sR_{90}(\hat{v}_0-\hat{v}_1)\|^2,\ R_{90}=\begin{bmatrix}0&1\\-1&0\end{bmatrix}$$

最后我们得到了

$\widehat{\quad}$ $\widehat{\quad}$ $\widehat{\quad}$

目的是让 最小：这是一个二次最优化问题，只需要求解一个稀疏线性方程组就好了。所以本质上我们得到了一个从 到$\widehat{\quad}$的homography变换，我们可以把它表示为

$\widehat{\quad}$

step 3

有了这些homography变换，我们就可以把每个cell里面相机是怎么运动的求出来了。我们有相机的位置公式：

（tip：和助教讨论了一下，似乎应该是左乘）,于是我们就可以对相机的路径进行稳定和优化。对于一条单独的路径有能量：

$$\mathcal{O}\left(\{P(t)\}\right) = \sum_t \left(\|P(t) - C(t)\|^2 + \lambda_t \sum_{r \in \Omega_t} \omega_{t,r}\left(\mathbf{C}\right) \cdot \|P(t) - P(r)\|^2 \right)$$

作者提出了对于整张图片有能量：

$$\sum_i \mathcal{O}\left(\{P_i(t)\}\right) + \sum_t \sum_{j \in N(i)} \|P_i(t) - P_j(t)\|^2$$

于是我们可以根据能量求出optimize之后的相机路径。

step 4

有了路径之后，只需要根据路径进行新的渲染就可以了。

三.编程实现

编程管线

整个视频稳定分为四个部分，就如前面step 1-4所说，第一步是加载图片，捕捉特征点。第二步是根据特征点求出相机的路径，用RANSAC来获取好的匹配特征点对，通过最小化能量求出homography变换，再累积求出路径。第三步是根据求出的路径对路径进行优化。第四步是渲染新的图像。但在处理之前，还要先通过　　把视频转化成图片，最后得到的图片也需要转化回视频。

基础功能

基本的实现助教已经帮我们写好了框架，需要我们做的只有通过特征点求出变换，并进一步求出相机路径。代码如下：

```
[f1,f2] = getF(tracks,frameIndex-1);
[homos] = NewWarping(f1,f2,tracks.videoHeight,tracks.videoWidth,tracks.videoHeight/MeshSize,trac
    for row = 1:MeshSize
        for col = 1:MeshSize
            A = zeros(3,3);
            B = zeros(3,3);
            A(:,:) = path(frameIndex-1, row, col, :,:);
            B(:,:) = homos(row, col, :, :);
            C = B*A;
%           C = C ./ C(3,3);  并没有必要，返回的时候一定是1
            path(frameIndex, row, col, :, :) = C(:,:);
        end
    end
```

首先通过 `getF` 函数获取好的匹配点对，然后通过 `NewWarping` 函数求出homography变换，这里在 `NewWarping` 函数中最后一个参数按照论文中的说法建议取3，这里取2，效果好一些。有了homography变换之后我们对每一个cell，左乘一个对应cell的homography矩阵，即公式中的　　，得到新的path值。

其他功能

从调试方面来看，最后输出的视频是有边框的可以让我们比较直观的看到图像的运动和扭曲，但从最终效果上来看，我们还是想要一个完整的成果，可以通过裁剪图像来实现：

```
imwarp1 = zeros(obj.videoHeight-obj.retraction*2,obj.videoWidth-obj.retraction*2,3);
for i = 1 : obj.videoHeight-obj.retraction*2
    for j = 1 : obj.videoWidth-obj.retraction*2
        imwarp1(i,j,:) = imwarp(i+obj.retraction+obj.gap,j+obj.retraction+obj.gap,:);
    end
end
imwarp = imwarp1;
```

在 `buldle.m` 文件里有render函数，可以对返回之前的img进行裁剪，具体的操作是设置一个 `obj.retraction` ，相当于相对原图像向里裁剪的宽度，再加上 `obj.gap` ，对整体做一个平移就行。

四.结果展示

视频见文件，这里展示改变参数之后的变化

```
% -------TRACK-------
TracksPerFrame = 256;           % number of trajectories in a frame, 200 - 2000 is OK
% -------STABLE------
MeshSize = 8;                   % The mesh size of bundled camera path, 6 - 12 is OK
Smoothness = 1;                 % Adjust how stable the output is, 0.5 - 3 is OK
Span = 30;                      % Omega_t the window span of smoothing camera path, usually set
Cropping = 1;                   % adjust how similar the result to the original video, usually s
Rigidity = 2;                   % adjust the rigidity of the output mesh, consider set it larger
iteration = 20;                 % number of iterations when optimizing the camera path [10 - 20]
% -------OUTPUT------
OutputPadding = 200;            % the padding around the video, should be large enough.
Retraction = 20;
```

以下均采用150张图片测试。

Meshsize = 12

path：历时 4.374496 秒
bundle：历时 72.837752 秒
render：历时 169.810494 秒
效果上并无明显不同。

Smoothness = 3

path：历时 3.957499 秒
bundle：历时 33.767324 秒
render：历时 88.977991 秒
发现为了让两帧之间尽量接近，不产生太大抖动，增大smoothness明显让变形变得更加柔性。

Rigidity = 4

path：历时 3.052340 秒
bundle：历时 30.532353 秒
render：历时 102.949171 秒
和smoothness正好相反，会让整个区域在抖动的时候更倾向于刚性变化，但会带来效果的减弱，看起来依然很抖。

iteration = 10;

path:历时 3.363078 秒
bundle：历时 16.499635 秒
render：历时 101.788055 秒
可以感受出来稳定的效果还没有那么好，迭代次数少了之后，路径也变得不精确了。
从时间上来看，增大meshsize会让时间显著增加，增加smoothness会减少渲染时间，减少迭代次数会让bundle时间减少，不过本来就不多。其余相差都不大。

视频结果有example跑出来的局部视频和全局视频，还有自己拍摄视频的局部视频和原视频。