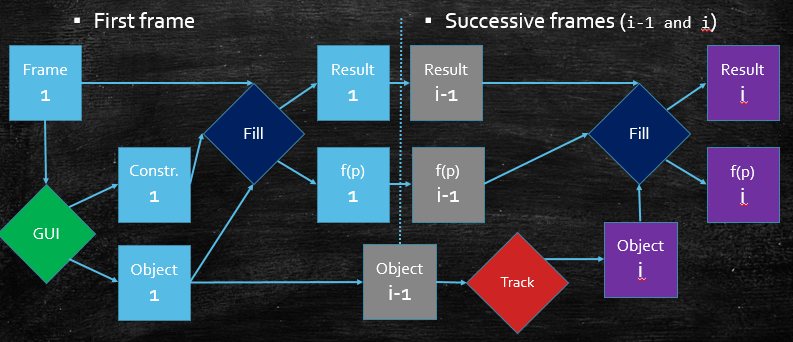
图像处理课程项目

**第二部分：基于像素映射的视频填补算法论文实现**

小组成员：张先耀、张宇、童羽强

一、算法目的及应用

二、算法原理



上图是我们的视频填补算法的流程。首先对第一帧，用上述物体选择算法经GUI

1.物体选择（童羽强实现）

1）算法流程

在物体选择这一部分，用户先看见视频的第一帧，并用鼠标粗略地圈出他想要消去的物体，随后算法开始执行，根据用户提供的 fingerprint 向内部寻找准确的表示物体的像素点，并计算边缘提供给物体追踪算法使用。

2）算法细节

物体选择算法主要思想为用户选取的 fingerprint 是图片背景中的一系列像素点，而需要消去的物体与背景由较大差异，因此如果能准确定义前景与背景的“差异”，物体选择就变得很容易。

在纯色背景下，背景点的RGB像素高度一致，因此前景的判别只需要对比像素点与背景点的RGB差异即可。由此，论文使用RGB三色通道作为图像特征，对 fingerprint 进行了聚类处理，得到C = {C\_1,C\_2, … C\_b} b个背景类别，（用来表示复杂背景），对每个背景类别的像素点RGB值进行方差计算，并以所有类别中方差的最大值为阈值来检测前景点。如果 fingerprint 内部某个像素点在RGB三色通道上与某个背景点的差异都不大于最大方差，则认为这个点与当前计算的背景点“相似”，如果一个内部点与95%的 fingerprint “不相似”，那么这个内部点被判断为需要被消去的“前景点”。

在算法实现中，由于Matlab的循环速度太慢，采用了Mex方法将内部像素点的判断写在了C++中，由于在Mex中调用matlab函数过于繁琐，手动完成了 inpolygon 函数的c++ 实现，随后上诉算法的效率（对于2000x1000像素的图片）从数分钟降至即时。并且得到的前景点与边界对于后续算法都是可用的。

2.物体追踪（张宇实现）S

1）算法流程

在物体追踪这一部分，输入视频在第1帧中由物体选择部分求出的边缘以及视频流，输出从第2帧开始，物体在每一帧之间位置的变换H，以及物体在每一帧的边缘。这一部分的算法主要分为4步：特征检测、特征追踪、求解变换、边界修正。

2）算法细节

a.特征检测

论文中选择用Harris角点作为待检测和追踪的特征。

Harris角点的思想是，当一个滑窗处在没有角点也没有边界的地方时，沿任意方向滑动时，滑窗内的灰度不会出现较大的变化；当一个划窗处在边界时，只有沿一个方向滑动时，滑窗内的灰度值会出现较大的变化；当一个划窗处在角点是，沿两个方向滑动时，滑窗内的灰度值都会出现比较大的变换。由此就能对角点进行判定。

具体度量时，先求，其中是图像在x方向的差分，是图像在y方向的差分。然后计算响应函数R = det(M) – k\*，其中k是需要调节的参数。

由此，给定了一帧图像以后，我们可以检测物体边缘上的特征。实现中，我们先计算图像各点处的响应R(x,y)。然后使用一个8×8的全1模板对上一帧得到的图像边缘进行膨胀，得到一组candidate点。物体的边缘上的角点都会在这组candidate中。然后我们对这组candidate点进行排序，越符合角点定义的排名越高。排序准则是，首先希望这个角点比其8邻域内尽量多的点的响应更大，其次希望这个点的响应的数值越大越好。我们选取排名中前n个作为检测出来待追踪的特征点，其中n是需要调节的参数。

b.特征追踪

检测完n个待追踪的特征点以后，需要对特征点进行追踪。论文中没有明确指出追踪使用的方法，我们选择使用LK光流法对特征点进行追踪。

LK光流法的思想是，对于一帧图像中的一组像素点，将这组点的灰度矩阵在这一帧图像对应位置的一定范围内进行移动，检查能够对应位置灰度的均方误差，误差最小的地方即为匹配的位置。两个匹配位置之差即为该点的光流向量。光流向量可以认为是对一个像素的运动进行了估计。但是实验中发现直接使用原始的LK光流法，跑10帧左右，轮廓就会开始出现偏差。主要原因可能在于LK光流法中搜索范围这一参数的选取与图像的分辨率有较大关系，参数不好调，追踪结果也不够健壮。搜索范围过小，无法追踪高速运动的物体，搜索范围过大，匹配错误的可能性会增加，并且时间开销会增大。

因此最终我们选用了LK金字塔进行特征追踪。LK金字塔对图像金字塔的每一层进行光流计算，上一层计算出来的残差交由下一层计算，这样每一层的搜索范围都不需要很大，也能达到追踪高速物体的效果，并且追踪的准确率较高。

c.求解变换

获得了前一帧中物体的n个特征点的坐标，以及这一帧中物体的n个特征点的坐标，就可以用来估计物体在两帧之间的变换。n越大，能够估计的变换的自由度越大。当n大于等于4时，可以去估计自由度最多的投影变换。由于求解变换时除了满足n的底线要求，还要求估计出的变换H的inlier不太少，因此实验中n一般取8或更大。

d.边界修正

估计出变换H以后，就已经能通过上一帧物体的轮廓点，获得这一帧物体的轮廓点。但是由于inpainting时对于轮廓点的准确度有较高的要求，至少要求物体的实际轮廓完全落在求得的轮廓内，否则没有落在求得的轮廓内的那部分物体最终就无法删去，会带来明显的瑕疵。因此，为了得到更加精确的边界，还要对轮廓进行修正。

边界修正的思路是对获得的原始边界进行扩展以后再收缩，具体有两种做法。

一种是沿原始轮廓上个点的法线向外、向内各延伸一定距离，沿法线从外到内，根据物体选择步骤中的聚类结果，依次检查每个像素属于背景类还是属于物体类，如果属于物体类，则重新划定这一点处的边界位置。

另一种是对原始轮廓直接进行膨胀，求膨胀以后的凸包，实现中使用8×8的全1矩阵进行膨胀。对这个凸包调用物体选择部分的整个算法，从而获得这个凸包内物体的轮廓。

3.逐帧图像填补（张先耀实现）

首先，对于一幅要填补的图像I，将其分为源区域S和目标区域T，我们要利用S中的信息来填补T。

在图像/视频填补（inpainting）领域中，较常见的模式是基于图像块（patch）的方法[3]，即在S中寻找与T边缘在某种意义下相似的图像块并逐渐填充T。本文中图像填补的算法则是基于像素映射（pixel mapping）的方法，即对任一像素点，要找到一映射，。

采用逐像素的能量函数来度量当前映射的好坏。如此，算法的目标可表示成下式：

优化该函数的方法为迭代法，与PatchMatch[4]中的方法相似，分为扩散（propagation）和随机搜索（random search）两步。在扩散（以从左上到右下为例）这一步中，对每个像素，我们试图用其左边一个像素和上边一个像素的映射来优化该像素的能量值。具体做法（以左边像素为例）是，取并计算对应的能量值，如果低于当前的对应的能量值，则更新。接下来对于进一步优化能量值：给定初始搜索半径和每次搜索次数，对于每个搜索半径，尝试次搜索，每次令

是上的随机向量。在第一次能够找到降低能量值的映射的时候便结束整个搜索。若某一半径无法找到合适的搜索结果，则半径减半进行下一次搜索。初始搜索半径为图像长宽较大值的一半。

搜索采用mex实现，matlab循环太慢。在每一次迭代时，采用如下策略来加快搜索：计算上次能量函数的平均值，能量值大于平均值的1.2倍的像素才进行随机搜索，其它像素只进行扩散操作。这样是由于能量函数大部分由少部分像素贡献，大部分像素能量值较低，故只需对这些不理想的像素进行搜索优化。

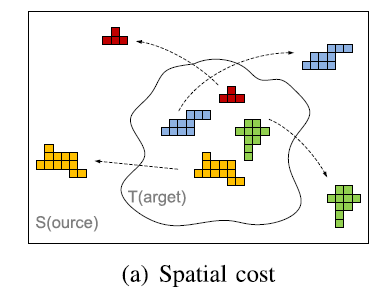
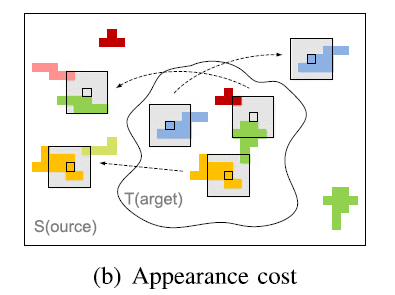
同时，采用多分辨率处理。多分辨率的最粗糙层满足T中每个点到S的距离不超过3。从最粗糙层获得的映射通过简单的插值作为下一层的映射的初始化。即（以0开始索引）

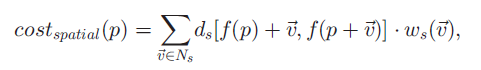
在实现这个算法的时候，采用了向量化，在整个matlab函数中没有使用循环，算法速度提高了约200倍。

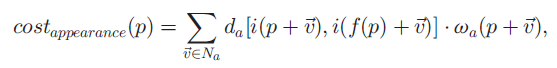
a. 第一帧填补

在填补第一帧时，没有任何之前帧的信息，输入为图片I、被选中的物体区域M、不应被用来填补的区域，以及一些用户输入的线段信息。

其中，。

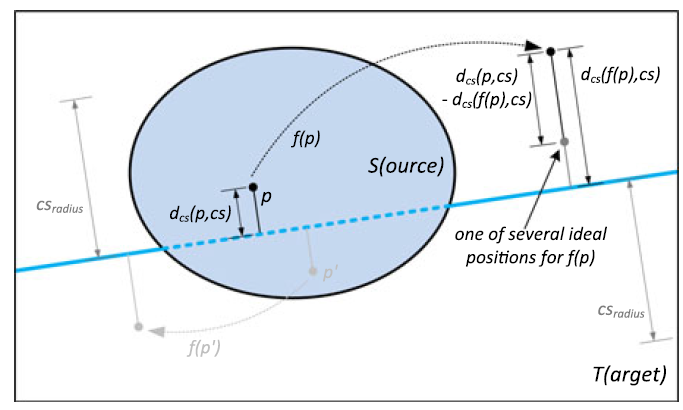
 





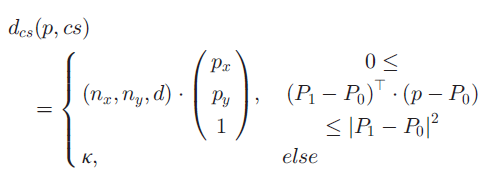
Spatial cost表示像素映射的连续性，即相邻的像素应当映射到相邻的像素。为被截平的欧氏距离，即距离过大时设为常值；为邻域，取8邻域以及自身，为。

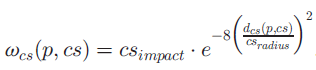
Appearance cost表示像素映射后不仅要考虑位置的连续性，也要考虑颜色值的一致性，即如果相邻的像素不映射到相邻的位置，那映射前后邻域的灰度值至少要较为相似。为5x5的patch，为。



Line constraint cost使得像素映射与原来的像素连线关于某一直线或线段平行，这样再加上spatial和appearance costs就可以在一定程度上保持直线结构。定义如下：



其中，

。

如果有多条直线约束，最大的一条直线（也就是最近的）的约束。在实现中，先利用Hough变换检测图中的直线段，再选出距离T较近的线段，加上用户声明的线段，共同组成第一帧的线段约束组。

另外，首帧填补保证。

b. 有前一帧的后一帧填补

在填补第帧时，利用及，输入还有跟踪后的物体区域，两帧之间估计的物体运动变换。由于背景直线运动可能较复杂，故不利用直线信息。首先，利用构造的初始化。

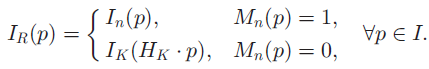


这里为在前一帧的对应像素坐标。由于这个初始化被认为比较可靠，不采用多分辨率处理，直接对原图进行迭代，这个迭代也很快收敛（到一个较好的局部极小值）。

同时，能量函数略有不同：

其中，与之前相同。定义与前一帧有关，即要保证这一帧的结果与前一帧的时间连续性。

首先，定义参考图像：



这里。

然后定义cost：



即新图像与参考图像对应邻域的像素值应当相近。论文中认为，要使得这个cost更好地起作用，应当首先对两幅图的亮度差进行补偿，也提出了通过插值的简单亮度补偿算法。我实现了这个算法，但这要求有对应的contour点，由我们的tracking算法不易得到，故没有应用该算法。另一种补偿办法是在计算时先对两个邻域中的像素值减掉均值（zero-mean SSD）。

三、实验结果及分析

四、结论

参考文献：

[1] J. Herling and W. Broll, "High-Quality Real-Time Video Inpaintingwith PixMix," in IEEE Transactions on Visualization and Computer Graphics, vol. 20, no. 6, pp. 866-879, June 2014.

[2] J. Herling and W. Broll, “PixMix: A Real-Time Approach to High-Quality Diminished Reality”, in ISMAR 2012.

[3] Y. Wexler, E. Shechtman and M. Irani, “Space-Time Completion of Video”, in TPAMI, vol. 29, no.3, 2007.

[4] C. Barnes, E. Shechtman, A. Finkelstein and D. B. Goldman, “PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing”, in CVPR 2009.

提交文件包括： 课程设计报告； 算法源代码和原图像； 阅读的论文； 所有文件压缩后使用“DIP+学号.zip"命名上载。