Mean-value Seamless Cloning 实验报告

2015011004 鞠家兴

1. 算法原理

MVC是一种 seamless cloning 算法，可以将一块区域复制到另一张图片上，并保证patch可以和目标图片较好的融合在一起。与简单的抠图（边缘精确的patch复制）相比，seamless cloning 算法可以处理抠图无法处理的例子，比如飞舞的毛发抠图过于复杂，又比如水中物体没有办法抠出确定的轮廓，抠图后边缘会明显的不舒服的感觉。

在此之前的类似算法是通过解泊松方程，计算复杂且耗费时间和内存，这篇论文提出了使用 mean-value coordinates 来插值，并使用三角化和边缘采样的方法加速。

1.1 算法流程

MVC 本质上是插值过程，对于区域中的每一个点，可以计算出边界上各个点对这个点的“贡献值” $\lambda_i(x)$，计算方法如下。

$$\lambda_i(\mathbf{x}) = \frac{w_i}{\sum_{j=0}^{m-1} w_j}, \quad i = 0, \ldots, m-1, \tag{1}$$

where

$$w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{\|\mathbf{p}_i - \mathbf{x}\|}, \tag{2}$$

其中的角度 α_i 这样计算：

$$\tilde{f}(\mathbf{x}) = \sum_{i=0}^{m-1} \lambda_i(\mathbf{x}) f(\mathbf{p}_i). \tag{3}$$

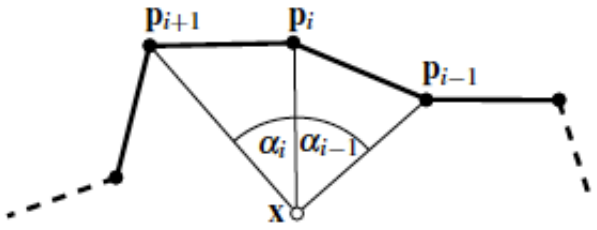

Figure 2: *Angle definitions for mean-value coordinates.*

根据上面的公式（3），看出这实质上是插值。

在此基础上，下面是大概的流程。首先，无论 patch 会在哪个地方，边界对内部点的插值系数只与 patch 本身有关，所以要先计算插值系数。然后对于 patch 内部的每一个点，计算 r(x)。

Algorithm 1 MVC Seamless Cloning

{Preprocessing stage}
for each pixel $\mathbf{x} \in P_s$ **do**
  {Compute the mean-value coordinates of $\mathbf{x}$ w.r.t. ∂P_s}
  $\lambda_0(\mathbf{x}), \ldots, \lambda_{m-1}(\mathbf{x}) = MVC(x, y, \partial P_s)$
end for
for each new P_t **do**
  {Compute the differences along the boundary}
  for each vertex $\mathbf{p}_i$ of ∂P_t **do**
    $diff_i = f^*(\mathbf{p_i}) - g(\mathbf{p}_i)$
  end for
  for each pixel $\mathbf{x} \in P_t$ **do**
    {Evaluate the mean-value interpolant at $\mathbf{x}$}
    $r(\mathbf{x}) = \sum_{i=0}^{m-1} \lambda_i(\mathbf{x}) \cdot diff_i$
    $f(\mathbf{x}) = g(\mathbf{x}) + r(\mathbf{x})$
  end for
end for

实际上为了运行速度，可以不必对内部的每个点都单独计算插值 $\lambda_i(x)$，而是可以网格化，只计算三角形顶点的插值系数；对于三角形内部的点，依据点到三个顶点距离做线性插值。这样能够显著提高准备阶段的速度。

2. 效果

2.1官方例子

2.2 更多例子

原始图片：

合成的新图片：

3. 思考与改进

运行速度方面应用了三角化网格，显著加快了速度。

另外，如论文中所述，源图片和目标图片应该相似，但是尽管相似，也可能造成 patch 内部被新背景同化，改变过大，损害 patch 内部的本来面目，比如下面就是一个例子，可以看到人物的身体颜色过浅，几乎看不清：

考虑一下改进措施，离边界越远的点应该更多的保留原有的颜色，离边界近的点应该与边界尽可能融合。于是相比于简单插值，引入顶点 x 处的衰减因子 $\lambda_x = exp(-1.0/MinDist)$， 其中 $MinDist$ 为 x 到边界的最小距离，如果小于等于1，那么 λ_x 直接取1。其实也不只这一个函数，只要满足距离大于1的时候函数取值 (0,1)，且最好为二阶导小于0的减函数即可。于是改进后的效果如下，稍微好了一些。