CG2023 HW2 Report

PB20111686 黄瑞轩

目标

本次实验的目的是根据 RBF 算法实现 image warping。加分项是解决白条纹问题。

原理

假设输入 n 个点对，则 $psrc, pdst \in \mathbb{R}^{n\times 2}, psrc = [\mathbf{p_1} \quad \dots \quad \mathbf{p_n}]^T, pdst = [\mathbf{q_1} \quad \dots \quad \mathbf{q_n}]^T$。

原来的点 $\mathbf{x} \in \mathbf{int}^{1\times 2}$ 将映射到 $f(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{a}_j b_j(\mathbf{x})$，这里 $\mathbf{a}_j, j = 1, \dots, n$ 是待定变量，$\mathbf{a}_j \in \mathbb{R}^{1\times 2}$。

约束条件需要改成 $f(\mathbf{p}_i) = \mathbf{p}_i - \mathbf{q}_i, i = 1, \dots, n$（现在的映射是起始差值），利用条件可解所有 $\mathbf{a}_j$，然后将每个坐标按 f 映射即可。

其中，b_j 是一个 $\mathbb{R}^{1\times 2} \to \mathbb{R}$ 的映射，在这选择 $b_j(\mathbf{x}) = \dfrac{1}{|\mathbf{x} - \mathbf{p}_j|^2 + d}$，$d$ 是防止除 0 而引入的常数。

对白色条纹的处理

如果直接计算 f 之后，再将原图片的所有像素映射到新像素，即对遍历变量 i, j，新图片是这样产生的：

$$im2(f(i,j)) = im(i,j) \tag{1}$$

这里假设不越界。这可能会导致 $im2$ 中出现白色条纹，因为 $f(i,j)$ 不一定会覆盖图片的所有像素。

如果我们记录的是一个逆映射 g^{-1}，也就是说：

$$im2(i,j) = im(g^{-1}(i,j)) \tag{2}$$

这里假设不越界。这样 $im2$ 中每一个像素都可以从 im 中确定地找到一个来源（越界额外处理），这样就可以避免白色条纹的出现。为此，需要翻转上面 $\mathbf{p}_i$ 和 $\mathbf{q}_i$ 的定义。

实现细节

下面会逐一讲解一下实现中的细节。

首先，对于 monalisa 这张图片，我选择 $d = 3000$。

> 经粗略的观察和测试，d 貌似与一对 $\mathbf{p,q}$ 之间的距离有关，如果 d 太小但 $\mathbf{p,q}$ 之间距离太大，可能无法正确处理。经过数次调节，选择 $d = 3000$ 可以达到比较好的测试效果。

关于 $im2$ 的初始化：

```
im2 = zeros(h, w, dim, 'uint8') + 255;  % 将目标图像初始化成全白
psrc(:,[1,2]) = psrc(:,[2,1]);          % 将输入点对的 x,y 转换，方便下面处理
pdst(:,[1,2]) = pdst(:,[2,1]);
```

使用 Matlab 可以方便地用矩阵表示所有 $b_j(\mathbf{p}_i)$，并计算出所有 $\mathbf{a}_i$：

```
dists = pdist2(pdst, pdst).^2;
bp_map = 1 ./ (dists + d);        % b 映射矩阵
A = bp_map \ (psrc - pdst);       % a 矩阵，解线性方程组可得
```

现在我们计算上面所说的逆映射：

```
src = transpose(meshgrid(1:h,1:w));      % 创建描述当前坐标位置的矩阵，方便计算
dst = meshgrid(1:w,1:h);

rever_f = zeros(h, w, 2);
for k = 1:n
    q = 1 ./ ((src - pdst(k,1)).^2 + (dst - pdst(k,2)).^2 + d);
    rever_f(:,:,1) = rever_f(:,:,1) + A(k,1) * q;
    rever_f(:,:,2) = rever_f(:,:,2) + A(k,2) * q;  % 按求和式计算逆映射
end
rever_f = round(rever_f + cat(3, src, dst));  % 因为改过的映射是增量，需要加上原来的坐标后取整
```

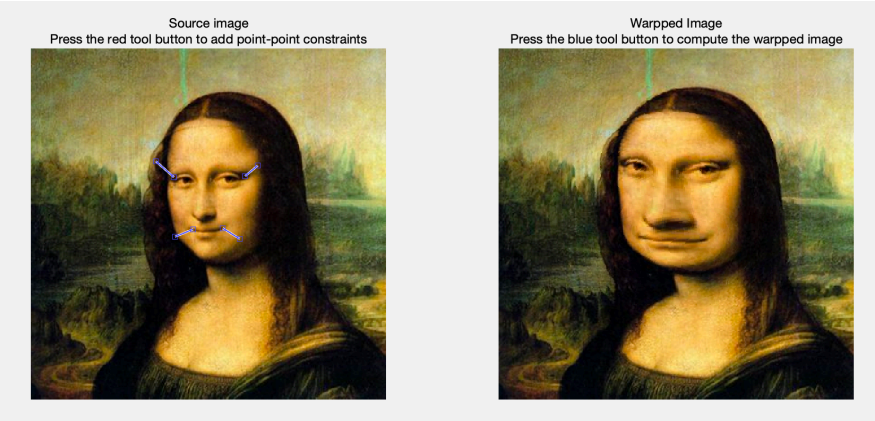
求出逆映射之后，只需要按上面公式（2）计算新图像各个像素：

```
for i = 1:h
    for j = 1:w
        x = rever_f(i, j, 1);
        y = rever_f(i, j, 2);
        if x < 1 || x > h || y < 1 || y > w
            im2(i,j,:) = [0,0,0];
        else
            im2(i,j,:) = im(x, y, :);
        end
    end
end
```

越界的像素用黑色填充，事实上这种越界只会发生在图像边界处，表示此处无法从原图像中寻得像素来源。为了优化，也可以将这些黑色像素填成附近有定义像素的平均值，不过这超出本次作业的要求了，就不再做这种优化。

结果

$d = 3000$ 可以做到 pdf 要求的微调：



$d = 30000$ 可以实现较大尺度的调整：