基于多角度单光源图像的人脸三维重建与渲染

郝千越[1]，胡开哲[2]，刘雨玚[3]
（[1]无 85，2018011153；[2]无 84，2018013326；[3]无 81，2018010651）

1 原理与实现

1.1 根据光照角度的图像渲染

图像渲染与重建是计算机视觉中的重要问题，在人脸识别、游戏视觉等领域有重要应用。本实验根据已知光照角度下得到的渲染结果分析人脸表面特性，同时得到给定光照角度下的渲染结果。可用的光照模型主要有完全漫反射模型、Phong 模型等，本实验主要选用完全漫反射模型（关于 Phong 模型的部分讨论见“3 问题与讨论”）。

完全漫反射模型公式如下：

$$\boldsymbol{b}(x,y)=k_d(x,y)\frac{\left(z_x(x,y),z_y(x,y),-1\right)}{\sqrt{z_x^2(x,y)+z_y^2(x,y)+1}}$$

$$m(x,y)=max(\boldsymbol{b}^T(x,y)\boldsymbol{s},0)$$

其中人脸表面的每个点特性由归一化法向量$\frac{(z_x(x,y),z_y(x,y),-1)}{\sqrt{z_x^2(x,y)+z_x^2(x,y)+1}}$描述其方向，由表面参数$k_d(x,y)$描述其表面反射能力等特性。综合法向量以及表面参数得到该点表面特性向量$\boldsymbol{b}(x,y)$后，与光照方向向量$\boldsymbol{s}$做内积即可得到该点渲染后像素值。

进行给定光照角度的图像渲染时，并不需要分离表面参数与法向量，如果先分别解得表面参数与法向量，再将两者相乘得到$\boldsymbol{b}(x,y)$，会造成误差的累加。因此只需要直接解得$\boldsymbol{b}(x,y)$的三个分量即可。设$\boldsymbol{b}(x,y)=(b_x(x,y),b_y(x,y),b_z(x,y))$；设训练集的光照向量构成的矩阵为$\boldsymbol{L}=\begin{pmatrix}l_x^{(1)} & l_y^{(1)} & l_z^{(1)}\\ \dots & \dots & \dots\\ l_x^{(7)} & l_y^{(7)} & l_z^{(7)}\end{pmatrix}$，其中$l_m^{(i)}(m=x,y,z;i=1,2,\dots,7)$表示第$i$条光线的$m$分量；设训练集中像素值$\boldsymbol{M}(x,y)=(M_1(x,y),M_2(x,y),\dots M_7(x,y))$，其中$M_i(x,y)(i=1,2,\dots,7)$表示第$i$幅图片$(x,y)$处的像素值。

由此根据公式可以得到超定方程：

$$\boldsymbol{L}\cdot\boldsymbol{b}(x,y)^T=\boldsymbol{M}(x,y)^T$$

用最小二乘法拟合求解可以得到待求参数：

$$\boldsymbol{b}(x,y)^T=(\boldsymbol{L}^T\boldsymbol{L})^{-1}\boldsymbol{L}^T\boldsymbol{M}(x,y)^T$$

上述方法中并没有考虑$m(x,y)=max(\boldsymbol{b}^T(x,y)\boldsymbol{s},0)$中这一类似 ReLu 函数的非线性操作。这一非线性运算无法直接用矩阵相乘的线性运算求解优化，因此用上面求出的$\boldsymbol{b}$作为初值，定义损失函数如下：

$$Loss(x,y)=\frac{1}{7}\sum_i\left(M_i(x,y)-m_\theta(x,y)\right)^2$$

其中m_θ表示用当前$\boldsymbol{b}$恢复的像素值。以损失函数用梯度下降的方法继续微调优化$\boldsymbol{b}$，容易计算得到：

$$\frac{\partial Loss(x,y)}{\partial\boldsymbol{b}(x,y)}=\frac{2}{7}\sum_i\left(m_\theta(x,y)-M_i(x,y)\right)\boldsymbol{s}[i]\,\mathbb{I}(\boldsymbol{b}(x,y)\cdot\boldsymbol{s}[i]>0)$$

将此形式扩展为矩阵形式，具体实现见代码文件“rebuild.py”。对$\boldsymbol{b}$优化到梯度小于一定值后完成计算。

由优化得到的$\boldsymbol{b}$直接根据$m_\theta(x,y)=max(\boldsymbol{b}^T(x,y)\boldsymbol{s},0)$即可得到给定光照$\boldsymbol{s}$下渲染图像每个点的像素值$m_\theta(x,y)$。

1.2 人脸深度三维重建

完成人脸深度三维重建首先需要**得到各点的法向量**$\frac{(z_x(x,y),z_y(x,y),-1)}{\sqrt{z_x^2(x,y)+z_x^2(x,y)+1}}$，根据上节中的完全漫反射公式分离得到该法向量。该法向量可以直接由$\boldsymbol{b}(x,y)$推出。有：

$$\boldsymbol{b}(x,y)=\frac{1}{\sqrt{z_x^2(x,y)+z_y^2(x,y)+z_z^2(x,y)}}\big(k_d(x,y)z_x(x,y),k_d(x,y)z_y(x,y),k_d(x,y)z_z(x,y)\big)$$

因此可以得到：

$$k_d(x,y)=\|\boldsymbol{b}(x,y)\|$$

考虑到该法向量实际只有两个自由度，为保证与完全漫反射公式中定义一致，统一取$z_z(x,y)$为负值，即归一化法向量：

$$\boldsymbol{n}(x,y)=\frac{1}{\sqrt{z_x^2(x,y)+z_y^2(x,y)+1}}\big(z_x(x,y),z_y(x,y),-1\big)=-|z_z(x,y)|\frac{\boldsymbol{b}(x,y)}{k_d(x,y)}$$

下面由法向量**恢复人脸深度**，考虑对深度数据进行基函数分解：

$$\bar{z}\big(x,y;\bar{c}(\boldsymbol{w})\big)=\sum\bar{c}(w)\boldsymbol{\phi}(x,y;\boldsymbol{w})$$

其中$\boldsymbol{\phi}(x,y;\boldsymbol{w})$为二维基函数系，$\bar{c}(\boldsymbol{w})$为相应的展开系数。对深度求偏导得到：

$$\bar{z}_x\big(x,y;\bar{c}(\boldsymbol{w})\big)=\sum\bar{c}(w)\boldsymbol{\phi}_x(x,y;\boldsymbol{w}),\qquad \bar{z}_y\big(x,y;\bar{c}(\boldsymbol{w})\big)=\sum\bar{c}(w)\boldsymbol{\phi}_y(x,y;\boldsymbol{w})$$

由恢复的归一化法向量 n，可以得到对深度偏导数的估计：

$$z_x^*(x,y)=n_x(x,y),\qquad z_y^*(x,y)=n_y(x,y)$$

故在最小二乘的意义下，可以给出深度数据基函数系数的最优估计：

$$\min_{\bar{c}}\sum_{x,y}\big(\bar{z}_x(x,y;\bar{c})-z_x^*(x,y)\big)^2+\big(\bar{z}_y(x,y;\bar{c})-z_y^*(x,y)\big)^2$$

借助估计得到的展开系数$c(\boldsymbol{w})$，对基函数组进行反变换即可得到深度 z。

2 结果与分析

2.1 根据光照角度的图像渲染

通过 1.1 中所述方法在给定光照角度下进行人脸的渲染。首先对训练集样本进行反向重建，验证上述方法的可行性。针对第一个训练集的人脸结果如下（图 1），其余两个训练集结果类似。

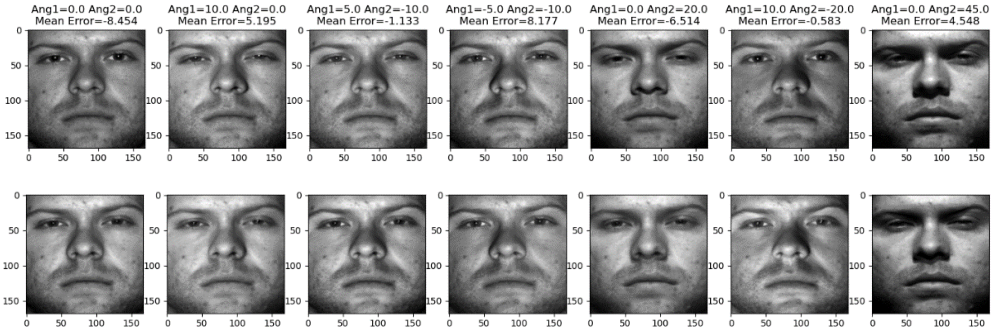

图 1 训练集上重建结果（第一行为原图，第二行为重建图）

可以看到，训练集上各光照角度下重建结果与原图在视觉上几乎没有差别。图中用所有像素值的平均偏差量化衡量误差（为表明相对明暗关系，未取绝对值），可见像素值范围为[0,255]时绝对误差较小，重建结果良好。

三个数据集上各 10 个光照方向渲染结果如下（图 2），可以看到，渲染结果在较小、较大的光照角度下均满足视觉效果的合理性，渲染结果良好。

图 2 测试集渲染结果

2.2 人脸深度三维重建

通过 1.2 中所述指导思想进行深度重建，初期采用正交余弦函数系进行展开，遇到了计算量过大、算法效果不佳等困难。通过网络资料查找，发现 MATLAB 库函数 grad2surf 通过离散正交多项式展开等数学工具较好地解决了这一问题。

为在 python 环境下调用该库函数，配置了 matlab. engine 库以实现统一运行环境，详细说明见附录 3。借助该库函数及估计得到的法向量获得深度图像。观察发现由于重建法向量、重建深度过程中的累计误差，人脸略有倾斜并存在少量表面毛刺，采取如下修正措施：在深度图中用高斯核进行滤波，使得深度结果平滑，从而更符合实际情况。

最终结果如下所示[1]：

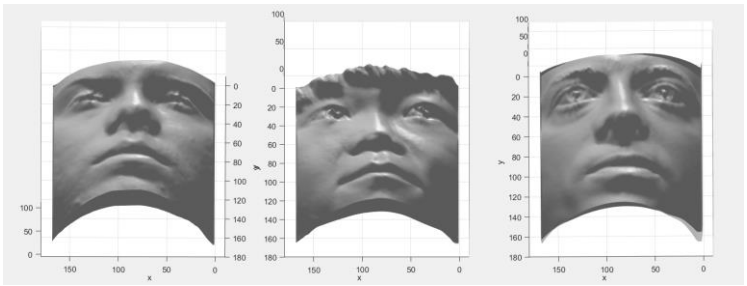

图 3 人脸深度重建结果

可以看出人脸表面的深度被准确恢复，第二幅图像中女士的头发也能够准确恢复。

3 问题与讨论

3.1 Phong 光照模型

本实验基于完全漫反射模型构建，实验中尝试了更加复杂的光照模型——Phong 模型。该模型将物体表面的光线分为环境光、漫反射光和镜面反射光，具体如下：

$$m(x,y) = m_e(x,y) + k_d(x,y)(\boldsymbol{n}(x,y)\cdot\boldsymbol{s}) + k_m(x,y)(\boldsymbol{v}(x,y)\cdot\boldsymbol{s})^{n_s}$$

其中$m_e(x,y)$为环境光项，$k_d(x,y)$为表面漫反射系数，$k_m(x,y)$为镜面反射系数，$\boldsymbol{s}$为光照方向，$\boldsymbol{n}(x,y)$为表面单位法向量，$\boldsymbol{v}(x,y)$为视角方向关于法向量的单位镜像向量，n_s为镜面理想系数。

该模型中$m_e(x,y)$提供了一定的环境光偏置，漫反射项与本实验中讨论基本一致。镜面反射项的引入是 Phong 模型主要特点，完全漫反射模型的渲染结果与视角无关，而镜面反射与视角相关联。视角镜像向量$\boldsymbol{v}(x,y)$与光线向量$\boldsymbol{s}$的点积反映了视角与表面镜面反射方向的重合程度；镜面理想系数n_s控制镜面反射高光的大小，当$n_s \to \infty$时为理想镜面，只有入射方向的镜像方向可以观察到反射光。

[1] 考虑视觉效果，此处 3D 重建结果使用 matlab 自动虚拟光源渲染，同时将图像四角置于同一水平面。结果中深度数据 z.npy 为直接重建保存得到，未做四角水平处理。

本实验中可以认为观察方向均为正面$(0,0,-1)$，则$\boldsymbol{v}(x,y)$可由$\boldsymbol{n}(x,y)$导出。因此模型共有 6 个自由参数：m_e、k_d、k_m、n_x、n_y、n_s，其中n_s可取整数值人为指定，其余参数由训练集拟合得到。由于该模型为非线性方程，难以得到类似最小二乘的解析解，故通过梯度下降的方法优化参数，具体过程见代码文件夹“Phong_code”。

结果显示该方法相比于完全漫反射模型并无明显优势，反而计算代价较大。分析原因主要有：

1） 观察典型的训练集样本，发现人脸图像表面并无大面积明显的高光点，因此 Phong 模型的优势难以发挥；

2） 模型中全部参数未知，梯度下降的初值难以确定，训练难度大。

因此，Phong 模型在该实验问题中并不适用，其更适用与已知部分表面参数数据库、渲染对象有较多光滑的、近似镜面的表面（如游戏人物装甲、车辆等）。

3.2 漫反射模型的不足

漫反射模型是本次重建过程的基本假设，但其仍存在一定的不足。在 2.2 所示深度恢复结果中，基于漫反射模型计算的法向量较好地反映了脸部区域的深度变化，但是在眼部区域出现了不合理的峰值，推测这是由于人眼处更加接近镜面反射导致的模型不适用所致。为了得到更合理的恢复结果，已经在深度图中用高斯核进行滤波，结果如下如上所述得到改善。如果可以应用更加复杂而真实的光照模型应当会从更根本的层面上改善这一问题。

4 附录

附录 1 成员名单及分工

姓名	班级	学号	分工
郝千越	无 85	2018011153	表面参数与法向量恢复、图像渲染、报告撰写
胡开哲	无 84	2018013326	深度恢复与三维重建、开发环境配置、报告撰写
刘雨玚	无 81	2018010651	文献调研、表面参数与法向量恢复、图像渲染

附录 2 文件清单

文件名	说明	文件名	说明
设计报告.pdf	本文件	setup.py	环境配置
demo.py	顶层文件	\dataset_offline	数据集、测试集结果
rebuild.py	由训练集重建法向量、表面描述		
depth.py	调用 depth.m 并返回深度	******以下为调试辅助函数******[2]	
depth.m	matlab 函数由法向量恢复深度	evaluate.py	评估训练集效果
g2s.m	用于恢复深度的 MATLAB 库函数	visualize.py	可视化结果
rendering.py	顶层文件，被 demo 调用	\Phong_code	Phong 模型试验代码
render.py	由表面描述及光照渲染图像		

附录 3 环境配置说明

本程序需通过 python 调用 MATLAB 引擎，配置方式为在文件夹根目录下命令行运行 python setup.py install，然后按照提示打开 MATLAB，在其命令行输入 matlabroot，获得返回的 MATLAB 根目录。在前述命令行界面按提示输入该根目录即可(去掉前后的单引号)(适用于 win10 系统)。

[2] rendering.py 中已注释 visualize 和 evaluate 代码，运行过程中无图像显示；取消注释可见阻塞输出的 evaluate 结果和最终渲染结果可视化。此处显示的深度图用 python 库函数直接画出，与 2.2 中效果有差异，使用 matlab 利用 3D 引擎画出的图像更加逼真，拖拽旋转也更加流畅。