一种从物体表面法线估计高度信息的算法

虞 鸿 吴哲夫 刘 恺 何熊熊
(浙江工业大学信息工程学院 杭州 310000)

摘 要 针对从法向量平面恢复高度信息这个问题,提出了一种从法向量平面恢复高度信息的算法。该算法首先利用一个定义了三维向量的法向量平面构造一个保存表面高度差的平面,然后从像素格状图的某个位置开始向四周相邻的像素累加高度,在遍历所有像素后利用计算好的高度再次从某个位置开始向四周相隔一个像素的位置累加高度,并根据需要的精度进行迭代运算,最后对累加后的高度值求算术平均值,从而得到每个位置最终的高度值。实验结果表明,该算法可以产生精度较高的高度图,并且可以有效地用于视觉效果更好的浮雕贴图技术。

关键词 表面重构,高度估计,表面法线,极坐标求和
中图法分类号 TP391.9 **文献标识码** A

Height Estimation Algorithm Based on Surface Normal

YU Hong WU Zhe-fu LIU Kai HE Xiong-xiong
(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310000, China)

Abstract Aiming at the problem of achieving height from a normal map, this paper proposed an algorithm recovering height information from a normal map of surface normals which are defined as three-dimensional vectors. The algorithm starts from a plane storing surface normals to construct a plane storing height difference, then accumulate neighbor pixel height from a certain location in pixel lattice. After all pixel do the same calculation, it again proceeds to a certain location using already calculated heights. This iteration ends according to the accuracy needed, and the total accumulated height is averaged. The experiment result shows that this algorithm can generate a relative high resolution height map, and it can be used well in relief map which has a better bump effect.

Keywords Surface reconstruction, Height estimation, Surface normal, Polar coordinates summation

1 引言

如果一幅格状图上的每个格子表示一个法向量,可以称之为法线图。利用法线图可以实现凹凸贴图,该技术最早由Blinn提出[1]。每个法向量表示一个在三维空间表面的一个朝向。有时候需要根据一幅物体表面法向量的法线图估计出这个表面的高度信息。

表面法向量可以利用Shape-from-Shading算法根据一幅图的明暗流估计得到[2-4]。在这个算法中,每个指定的面积分配一个法向量。这个算法可以直接产生物体表面的法向量,得到的法向量可直接用于物体表面光照计算。为了获得模型表面更好的光照效果增强凹凸感,需要从表面法向量估计出高度信息。Gavin D. J. Smith等人通过一种迭代算法提取了高度信息[5]。但是该算法只是从每个点周围的4个像素获得高度,然后利用计算的高度做相同的迭代,却没有获得足够真实的高度信息,王松等人就是利用该算法获取高度信息,进而产生浮雕效果[6]。Rony Zatzarinni和王建国等人则是利用表面法向量求得表面高度函数,并根据该函数来计算表面高度[7,8],但是它的计算量相当大。

考虑到算法的效率和最终效果的真实程度,介绍了一个极坐标积分方法来估计这个高度值。由法线图产生的深度差值图按照一定规则的积分可以获得高度信息,这个高度算法针对一个指定区域,将其周围的区域作为该指定区域的高度参考标准。周围的区域按照所需的精度来确定大小。该算法是一个根据上一次计算获得高度的迭代运算,是以某区域为中心累加隔壁区域的高度,整个表面计算完毕后,再次用上次计算得到的高度值继续累加相隔一个像素区域的高度,这个过程就是对一个区域做极坐标运算。这样多次计算就能得到更加精确的高度信息。

本文实验结果展示了用此方法从一种保存法向量的图中获得高度信息的高度图,我们利用法线图和高度图实现浮雕贴图来模拟凹凸效果,验证了算法的正确性。

2 表面高度估计

本节提出了从一幅保存物体表面法线的二维图估计表面高度信息的问题,解决方法流程如图1所示。为了得到高度

到稿日期:2012-07-05 返修日期:2012-10-15 本文受浙江省科技厅重大专项(2011C13011),浙江省自然科学基金(Y1090539)资助。
虞 鸿(1988-),男,硕士生,主要研究方向为计算机图形,E-mail:yhzjut@gmail.com;吴哲夫(1971-),男,博士,副教授,主要研究方向为移动多媒体通信、IP网络;刘 恺(1961-),男,硕士,副教授,主要研究方向为自动化技术;何熊熊(1965-),男,博士,教授,主要研究方向为机器人鲁棒控制。

信息，在已有的法线图上计算每个像素与四周像素的高度近似差值[9]，由这些高度差值组成的二维图就是高度差值图。一旦高度差值图生成之后，独立地对每个像素计算高度值，计算过程中没有指定目的点或者方向。只是从起始像素开始沿四周从零开始做高度累加，为了提高精度，可以适当地增加极坐标积分的半径。最后，得到的值就是四周像素针对这个像素点的高度总和，可以除以参与运算点的个数来获得相对比较准确的平均值。

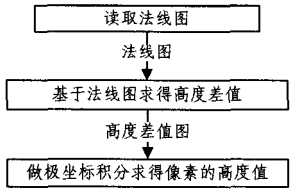

图 1　算法流程图

2.1　高度差值图的定义和获取

高度差值图定义为一个与表面法线图相对应的反映物体表面高度的图[10]。法线图每个像素存储的是该位置的法向量。高度差值图每个像素存储的是该位置与周围位置的高度差。

首先考虑一维的法向量图的一个像素（见图 2），S_{pixel} 表示像素的长度，用 $N(N_x,N_z)$ 表示这个表面的法向量，所有法向量都是归一化的，如式(1)所示：

$$N_x^2+N_z^2=1 \tag{1}$$

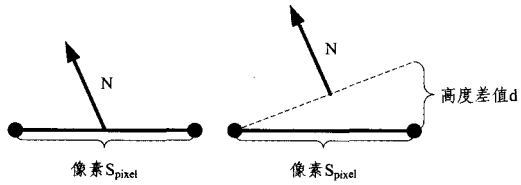

图 2　一维法向量图高度差值分析

我们把 d 认为是物体表面两个相邻点的高度差的估计值，N_x 和 N_z 分别表示法线量 N 沿着 X 和 Z 方向的分量，显然 d 的估计可以把平面抬高到与法向量垂直的位置，右侧点离开平台的长度就是该像素所处的位置与周围的高度差值，根据三角公式，式(2)给出了高度差值的计算方法。

$$d=\frac{N_x}{N_z}\cdot S_{pixel} \tag{2}$$

一幅一维的高度差值图就是在表面法线图上对每个像素做上述计算并保存的图片。以此类推，二维高度差值图的生成方法是一样的，先把需要计算的法向量分解成 X,Y 两个方向，再按照一维的计算方法得出二维的高度差值式(3)、式(4)，为了简化运算，可以把像素长度 S_{pixel} 设为 1。

$$d_x=\frac{N_x}{N_z}\cdot S_{pixel} \tag{3}$$

$$d_y=\frac{N_y}{N_z}\cdot S_{pixel} \tag{4}$$

2.2　基于高度差值图的高度获取

根据高度差值图的定义，为了获取某个像素的高度值，可以独立地从某个位置开始，累加深度差值，最后得到的就是该像素的高度值，所有位置高度初始化都为 0。我们先从一维法向量贴图考虑，如图 3 所示，假定第一个像素高度为 0，那么第二个像素的高度为 d_0，第三个像素的高度为 d_0+d_1，依此类推。因此可计算得一维法线贴图的第 n 个像素的高的计算公式为式(5)。

$$h_n=\sum_{i=0}^{i=n}d_i \tag{5}$$

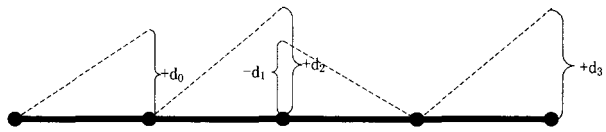

图 3　一维法向量图高度计算分析

理论上二维图高度获取和一维图高度获取计算方法是一样的，也可以选取某个点为起始点（高度为 0），然后选择合适的一条途径进行累加深度。如图 4 所示，选择 a 路径从起始点开始首先往 X 方向累加深度差值，然后再往 Y 方向累加深度差值，到达终点后就得到了终点的高度值，可以推出二维图表达式为式(6)。

$$h_n=\sum_{i=0}^{i=m}d_{xi}+\sum_{i=m}^{i=n}d_{yi} \tag{6}$$

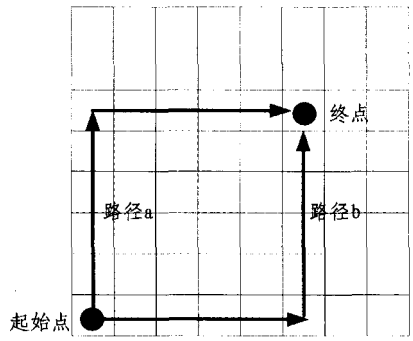

图 4　二维法向量图高度计算分析

但是法线贴图有以下几个因素将导致使用以上方法计算高度时会出现累计误差：

1. 法线贴图是有损压缩的。

2. 每个像素保存的是平均法向量，并不能提供实际每个点的法向量。

3. 每个像素保存的法向量之间是不连续的，缺少不连续的信息。

因此，图 4 中分别按照 a 路径和 b 路径求和，得到的高度是不一样的，即 $h_n(a)\neq h_n(b)$。

由于以上误差因素，希望构造出绝对完美的高度图，但尽可能使用一种方法逼近真实，使得最后的结果至少视觉上是可信的。

根据以上分析，为了减少上述因素带来的误差，最直接的方法就是多取几条路径累加，然后取平均值。但是这样又带来一个问题，若图 4 中多选取几条路径，那么起始点的高度几乎是由靠终点那一带像素高度来决定的。这样虽然能缓解误差，但是得到的高度图会有很明显的高低趋势。我们知道，一个实际的高度平面必定是连续的，一个像素点的高度几乎是由周围的像素高度决定，因此就应运而生了一种用周围像素的高度来估算中心像素的方法，本文称之为极坐标积分法。算法步骤如下：

1. 为了方便说明算法，图 5 中选择中间点为需要计算高度的像素。以该点为中心，画一个半径为 r 的圆，作为积分区域。选择 r 的大小对结果影响非常大，在下节的结果分析中将做出说明，这里选择的半径为像素长的两倍左右。

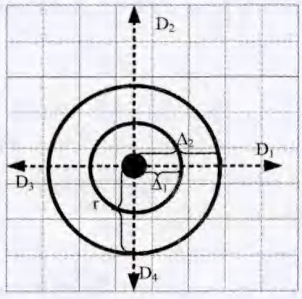

图 5　二维法向量图高度极坐标累加分析

2. 做积分的时候为了涉及到所有周围的像素点，可以把区域圆尽量分得密，这里为了考虑计算效率和清楚地表达算法，把区域分成了 4 份。

3. 接下来就把 4 个方向上基于中心点的高度值累加。由上节的结论可知，D_1 向上所有像素针对中心点的高度累加。d_1 是靠近中心的 D_1 方向第一个像素的深度差值，h_1 是这个像素基于中心点 D_1 方向第一个像素的初始高度，那么 $\Delta_1 d_1 + h_1$ 就是计算后的高度，同理 $\Delta_2 d_2 + h_2$ 就是基于中心点 D_2 方向第二个像素的计算后的高度。用相同的方法可以得到 4 周 4 个方向的高度：式(7)－式(10)。

$$h_{D_1} = \Delta_1 d_1 + d_1 + \Delta_2 d_2 + h_2 \tag{7}$$

$$h_{D_2} = \Delta_1 d_3 + h_3 + \Delta_2 d_4 + h_4 \tag{8}$$

$$h_{D_3} = \Delta_1 d_5 + h_5 + \Delta_2 d_6 + h_6 \tag{9}$$

$$h_{D_4} = \Delta_1 d_7 + h_7 + \Delta_2 d_8 + h_8 \tag{10}$$

4. 此算法是基于一个假设，选定的区域平均高度是零。这个假设也是非常合理的，因为在一个区域里，必然会有个水平基点，以这个基点，所有像素点的高度和为零。本文的算法使用的基点就是需要计算的这个中心像素点。根据假设可得，中心像素点的高度为式(11)。

$$h = \frac{1}{pixelnum} \int_0^r \int_{2\pi}^0 H(r,\sigma) \mathrm{d}\sigma r \mathrm{d}r \tag{11}$$

5. 对每个像素做上述相同的运算，就可以得到完整的高度图。

以上算法可以简化成一个极坐标积分表达式，如式(12)，$H(r,\sigma)$ 表示角度为 σ、离中心距离为 r 时的高度值。

$$h = -(h_{D_1} + h_{D_2} + h_{D_3} + h_{D_4}) / pixelnum \tag{12}$$

3　算法效果性能分析与结果验证

3.1　算法效果性能分析

实验在一台装有 2.27GHz CPU、2G 内存、GeforceGT 330M显卡、Windows 7 的 PC 机上进行。采用 C++编程语言实现了算法，实现了法线贴图转换到高度图的功能。表 1 给出了算法在指定的参数下运行的对应时间。算法有两个重要的参数，即积分区域大小和积分角度的步进，区域大小可以用积分半径 r 描述，积分角度步进可以用每次积分增加的角度 $step$ 描述，分别把这两个参数作为输入获得的高度图进行对比。图 6 为实验使用的大小为 256×256 的表面法线图，图 7 为在不同参数下生成的高度图。

图 6　表面法线图

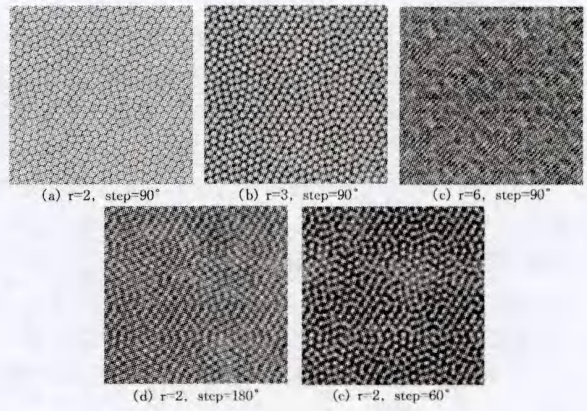

图 7　不同参数下生成的高度图

表 1　不同参数下算法运行的时间

积分半径	积分步进(rad)	运算时间(s)
2	π/3	8.2
2	π/2	6.1
2	π	3.3
3	π/2	10.3
6	π/2	24.6

如上述结果图所示，算法在积分半径为 3 个像素长度，积分步进为 π/2 的参数；积分半径为 2 个像素长度，积分步进为 π/3 下运行的高度图效果最令人满意，如图 7(b)和图 7(e)所示。它们呈现的高度明显，黑色的部分表示高度低，白色的部分表示高度高。由此可以得出，积分区域划定之后，区域内的像素的高度差值尽可能地都要进行累加，不能把区域内的像素忽略掉，那么得到的高度图效果是最好的。

针对以上几种参数设定，实验分别测量它们的执行时间，从而分析算法的性能。从表 1 的运算时间分析可以知道，该算法运行在秒的数量级上，因为这是对图片像素级别的运算，里面涉及到大量的乘法和加法。因此难以满足实时地按需渲染，只能提前把需要用高度图的模型生成高度图，再进行渲染。

为了进一步验证本方法的有效性，取 4 张大小不同的表面法线图，分别采用了 3 种不同算法进行比较。在相同条件下测得表 2 所列的运算时间。由表 1 分析，本文算法在积分半径为 2、积分步进为 π/3 的参数下运行。从表 2 所测的数据可见，在法线图比较小的时候，运算时间相差无几。但是在数据量增加时，本文的方法在运算时间上有一定优势。主要原因就是迭代算法需要用到比较大的存储空间，算法运行的过程中数据量太大。高度函数计算是通过不断迭代来减小邻域像素存在的误差。本文算法用了极坐标积分的方式来计算

高度，大大减少了计算复杂度，说明该算法是一种较好的算法。

表 2　3 种计算表面高度算法运行时间的比较

算法 / 法线图大小	本文算法	迭代算法	函数计算法
32 * 32	0.20s	0.18	0.23s
64 * 64	0.70s	0.84s	0.73s
128 * 128	2.42s	3.87	3.4s
256 * 256	8.20s	12.36s	11.23s

3.2　算法结果验证

在上述相同的 PC 上验证高度图的正确性，实验采用能体现高度效果的浮雕贴图技术，即利用一张标准的表面法线图和用算法恢复的高度图，在光照情况下产生凹凸更加明显的效果，此外也采用了没有高度信息的法线贴图技术作为对比。该验证实验利用 OpenGL 接口通过 CG 语言编程实现。浮雕贴图技术的实现是基于 Fabio 提出的二分搜索方法[11]，它的核心思想是除了利用表面法向量之外，还利用高度信息改变位移实现凹凸效果[12]。图 8 就是使用表面法向量图（见图 6）和本文算法生成的高度图（见图 7(e)）实现的凹凸效果，它的凹凸效果明显强于仅仅只用法线贴图的凹凸纹理映射（见图 9），尤其是离光照比较远的部分。

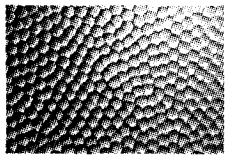

图 8　利用表面法线图和本文算法高度图的凹凸效果

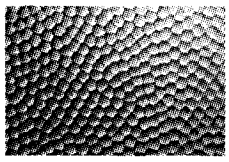

图 9　只利用表面法线图的凹凸效果

结束语　本文提出了一种从物体表面法向量估计高度信息的新颖算法，为了实现它，首先根据表面法线图计算出高度差值，并且利用高度差值图来保存物体表面像素与像素之间的高度差；然后，选定极坐标积分的半径以及步进角度，分别对每个像素进行极坐标积分运算。实验结果表明，本文的算法具有较高的准确度，生成的高度图完全可以用来实现浮雕贴图技术。由于算法计算量较大，本文方法仍然不适合实时渲染高度信息，只能对有需求的物体模型事先做好高度图，这是本算法需要改进之处。

参 考 文 献

[1] Blinn J F. Simulation of wrinkled surfaces[C]// Proceedings of SIGGRAPH. 1978:286-292

[2] Zhang R, Tsai P-S, Cryer J E, et al. Shape from shading: A survey[J]. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1999, 8(21): 690-706

[3] Feng Qiao-sheng, Wang Yun-qiong, Tian Jie. Low Dimensional and Easily Interactive Method of 3D Shape from Shading[C]// International Conference on Wireless Communications, Networking and Mobile Computing. 2010

[4] Ren Guo-quan, Li Wen-zhao, Chen Liang, et al. Application of Shape from Shading Algorithm in Wear Debris 3D Surface Shape Recovery[C]// International Conference on Measuring Technology and Mechatronics Automation. 2010: 1: 586-589

[5] Smith G D J, Bors A G. Height Estimation From Vector Fields of Surface Normals[C]// Digital Signal Processing. 2002: 1031-1034

[6] 王松，李著文，于金辉. 利用拓片恢复汉画像的浮雕效果[J]. 计算机辅助设计与图形学学报，2011，23(5)：784-789

[7] Zatzarinni R, Tal A, Shamir A. Relief Analysis and Extraction [J]. ACM Transactions on Graphics, 2009, 5(28)

[8] Wang Jian-guo, Wang Xiao-tong, Xu Xiao-gang. The Algorithm of Extraction Base Surface from Mesh[C]// International Conference on Information, Electronic and Computer Science. 2010: 1467-1470

[9] Tasdizen T, Whitaker R, Burchard P, et al. Geometric Surface Processing via Normal Maps [J]. Transactions on Graphics, 2003, 4(22): 1012-1033

[10] Kelly A R, Hancock E R. Surface Hegiht Recovery From Surface Normals Using Manifold Embedding [C]// International Conference on Image Processing. 2004: 2107-2110

[11] Policarpo, Fabio. Relief Mapping in a Pixel Shader Using Binary Search[EB/OL]. http://www. Paraleelo. com. br/A rquivos/ReliefMapping. pdf, 2012-07-04

[12] Miyazaki R, Harada K. Creating the Displacement Mapped Low-Level Mesh and its Application for CG Software[J]. International Journal of Image and Graphics, 2010, 3(10): 467-480

（上接第 302 页）

[8] Veksler O, Boykov Y, Mehrani P. Superpixels and Supervoxels in an Energy Optimization Framework[C]// Preceedings of European Conference on Computer Vision (ECCV'10). 2010: 211-224

[9] Heikkiä M, Pietikäinen M, Schmid C. Description of Interest Regions with Center-Symmetric Local Binary Patterns [C]// Preceedings of ICVGIP. 2006: 58-69

[10] Comaniciu D, Meer P. Mean Shift: A Robust Approach Toward Feature Space Analysis[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(5): 603-619

[11] Stricker M A, Orengo M. Similarity of Color Images[C]// Preceedings of Storage and Retrieval for Image and Video Databases (SPIE). 1995: 381-392

[12] Ojala T, Pietikainen M, Harwood D. A comparative study of texture measures with classification based on feature distributions [J]. Pattern Recognition, 1996, 1(29): 51-59

[13] Heikkila M, Pietikainen M. A texture based method for modeling the background and detecting moving objects [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(4): 657-662

[14] Candemir S, Akgül Y S. Adaptive Regularization Parameter for Graph Cut Segmentation [C] // Preceedings of ICIAR. 2010 (6111): 117-126

[15] Rother C, Kolmogorov V, Blake A. Grabcut: Interactive foreground extraction using iterated graph cuts[J]. Preceedings of SIGGRAPH, 2004, 23(3): 309-314