Contents

I Overview of Computer Graphics

1. Visual Effects

- ACM SIGGRAPH
- GDC

2. The Graphics Process

3. Geometric Modeling

- Represent 3D object
- Construct 3D object
- Manipulate 3D object

3.1 Represent 3D object

1) Triangular Mesh: 效率非常高, 虽然近距离很粗糙.
   游戏里用的多
2) Subdivision Surface(细分曲面):
   coarse mesh and subdivision rule
   电影里用的多
3) Parametric Surface(参数曲面): $z = f(x, y)$
   人造物体
4) Implicit Surface(隐式曲面): $f(x, y, z) = 0$
   把 $f(x, y, z)$ 的值存在体素的角点上.
   擅长动态的拓扑, 或者神经网络

3.2 Construct 3D object

1) 3D Scanning
2) Displacement Map

3.3 Manipulate 3D object

1) 3D Mesh Deformation

4. Texturing

5. Animation

6. Rendering

II Differential Geometry Primer

网格参数化

1. Parameterization

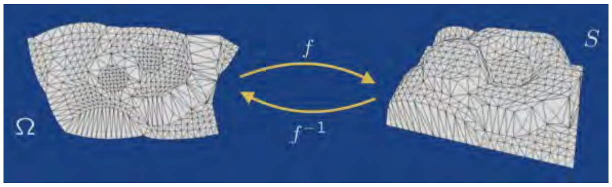

Figure II.1: Parameterization

- Surface $S \subset \mathbb{R}^3$ (开放曲面, 有边界)
- Parameter domain $\Omega \subset \mathbb{R}^2$
- mapping $f : \Omega \to S$ and $f^{-1} : S \to \Omega$

2. Example

Cylindrical Coordinates:

- $S : \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 = 1, z \in [0, 1]\}$
- $\Omega = \{(\phi, h) \in \mathbb{R}^2 : \phi \in [0, 2\pi), h \in [0, 1]\}$
- $f(\phi, h) = (\sin\phi, \cos\phi, h)$

Orthographic Projection:

- $S : \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 1, z \geq 0\}$
- $\Omega = \{(u, v) \in \mathbb{R}^2 : u^2 + v^2 \leq 1\}$
- $f^{-1}(x, y, z) = (x, y)$
- $f(u, v) = (u, v, \sqrt{1 - u^2 - v^2})$

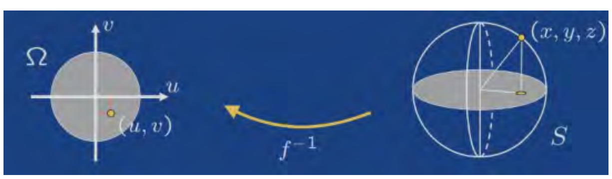

Figure II.2: Orthographic Projection

Stereographic Projection:

- $S : \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 1, z \geq 0\}$
- $\Omega = \{(u, v) \in \mathbb{R}^2 : u^2 + v^2 \leq 1\}$
- $f^{-1}(x, y, z) = \left(\frac{x}{1+z}, \frac{y}{1+z}\right)$
- $f(u, v) = \left(\frac{2u}{1+u^2+v^2}, \frac{2v}{1+u^2+v^2}, \frac{1-u^2-v^2}{1+u^2+v^2}\right)$

Mapping of the Earth

3. Distortion is almost inevitable

Theorem II.1 (Theorema Egregium) *A general surface cannot be parameterized without distortion.*

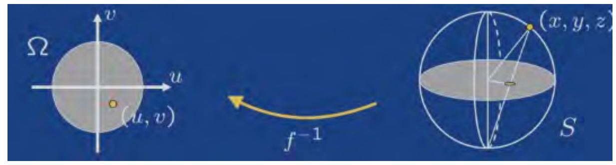

Figure II.3: Stereographic Projection

- no distortion = conformal(保角) + equiareal(等面积) = isometric(等测度)
- requires surface to be developable(可伸展)

  planes

  cones

  cylindrical

4. Distortion

- parameter point $x = (u, v) \in \Omega$
- surface point $p = f(x) \in S$
- small disk $D(x, r)$ around x

$$D = D(x, r) = \{y \in \Omega : \|x - y\| \leq r\}$$

- image(像) of D under f

$$f(D) = \{f(y) : y \in D\} \subset S$$

- shape of $f(D)$

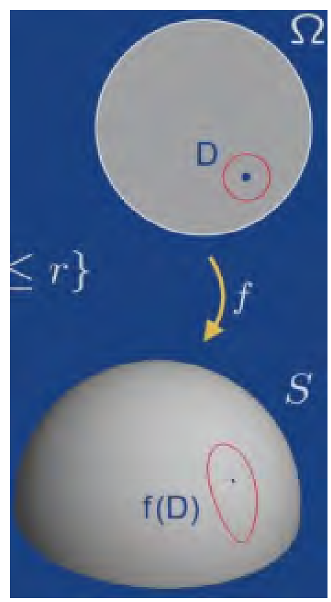

Figure II.4: Distortion

5. Linearization

- f_u, f_v 分别是 f 对 u, v 的偏导

- Jacobian of f, 这里直接取在 x 的值

$$J_f = [f_u, f_v] \in \mathbb{R}^{3\times 2}$$

- tangent plane at p

$$T_p = \{p + \alpha f_u + \beta f_v : \alpha, \beta \in \mathbb{R}\}$$

- Taylor expansion of f

$$f(y) = f(x) + J_f(y - x) + \cdots$$

- first order approximation of f

$$g(y) = p + J_f(y - x) \in T_p$$

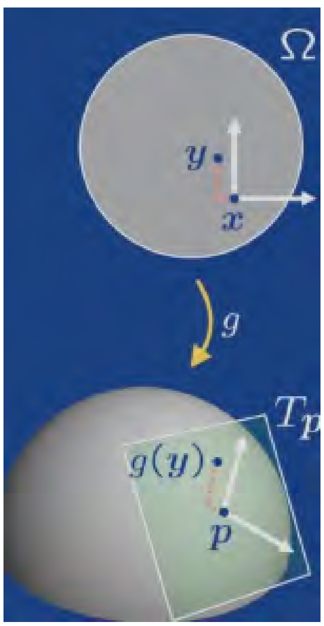

Figure II.5: Linearization

6. Infinitesimal Dis(k)tortion

- small disk $D(x, r)$ around x
- image of D under g

$$g(D) = \{g(y) : y \in D\} \subset T_p$$

- shape of $g(D)$
  ellipse
  semi-axes(半轴) $r\sigma_1$ and $r\sigma_2$
- behavior in the limit

$$\lim_{r\to 0} g(D) = f(D)$$

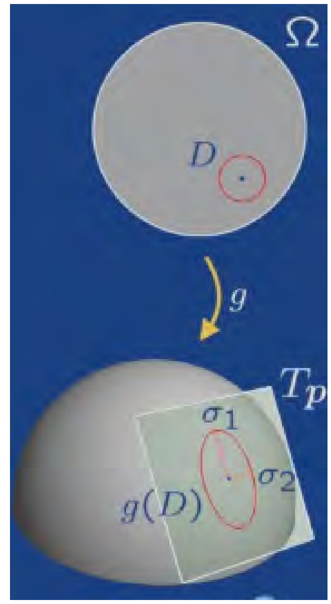

Figure II.6: Infinitesimal Dis(k)tortion

7. Linear Map Surgery

- Singular Value Decomposition (SVD) of J_f

$$J_f = U\Sigma V^\top = U \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{pmatrix} V^\top$$

with rotations $U \in \mathbb{R}^{3\times 3}$ and $V \in \mathbb{R}^{2\times 2}$ and scale factors (singular value) $\sigma_1 \geq \sigma_2 > 0$

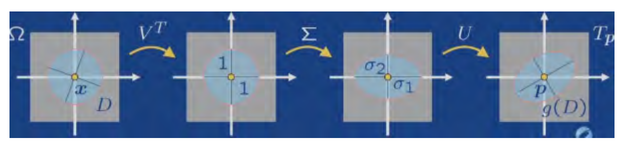

Figure II.7: Linear Map Surgery

8. Notion of Distortion

- isometric or length-preserving
  $\sigma_1 = \sigma_2 = 1$
- conformal or angle-preserving
  $\sigma_1 = \sigma_2$
- equiareal or area-preserving
  $\sigma_1 \cdot \sigma_2 = 1$

everything defined pointwise on Ω

9. Example

Cylindrical Coordinates:

- $f(\phi, h) = (\sin\phi, \cos\phi, h)$
- $J_f = \begin{pmatrix} \cos\phi & 0 \\ -\sin\phi & 0 \\ 0 & 1 \end{pmatrix}, \sigma_1 = \sigma_2 = 1$ (isometric)

Orthographic Projection

- $f(u, v) = (u, v, \sqrt{1 - u^2 - v^2})$
- $J_f = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -ud & -vd \end{pmatrix}$ with ...
- $\sigma_1 = 1, \sigma_2 = d$ neither conformal nor equiareal

Stereographic Projection:

- $\sigma_1 = \sigma_2$ conformal

10. Computing the Stretch Factors

- first fundamental form I_f
- eigenvalues of I_f
- singular values of J_f

11. Measuring Distortion

- loacl distortion measure

$$E : (\mathbb{R}_+ \times \mathbb{R}_+) \to \mathbb{R}$$
$$(\sigma_1, \sigma_2) \mapsto E(\sigma_1, \sigma_2)$$

- E has minimum at
  - $(\sigma_1, \sigma_2) = (1, 1)$ isometric measure
  - $(\sigma_1, \sigma_2) = (x, x)$ conformal measure
- overall distortion

$$E(f) = \int_\Omega E(\sigma_1(u, v), \sigma_2(u, v)) du\, dv \Big/ Area(\Omega)$$

12. Example

Conformal Measures

- Conformal energy

$$E_C = (\sigma_1 - \sigma_2)^2/2$$

- MIPS energy

$$E_M = \kappa_F(J_f) = \|J_f\|_F \left\|J_f^{-1}\right\|_F = \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}$$

isometric Measures

- Green-Lagrange deformation tensor
- Combined energy

Other Measures

- Dirichlet energy
- Stretch energies

13. Piecewise Linear Parameterizations

- piecewise linear atomic maps $f|_t : t \to T$
- distortion constant per triangle (因为映射是线性的，求偏导得 J_f 得 distortion 就是常数)
- overall distortion

$$E(f) = \sum_{t\in\Omega} E(t)A(t) \Big/ \sum_{t\in\Omega} A(t)$$

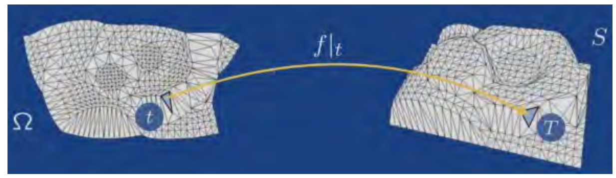

Figure II.8: Piecewise Linear Parameterizations

14. Beyond Distortion

- surface normal
- surface area
- independent of the particular parameterization
- intrinsic surface properties

15. Curvature

- second fundamental form
- Gaussian curvature
- mean curvature

16. Triangle Mesh Parameterization

- triangle mesh $S \subset \mathbb{R}^3$
  vertices $p_1, \ldots, p_{n+b}$ (n 个内部点, b 个边界点)
  triangles $T_1, \ldots, T_m$
- parameter mesh $\Omega \subset \mathbb{R}^2$
  parameter points $u_1, \ldots, u_{n+b}$
  parameter triangles $t_1, \ldots, t_m$
- parameterization $f : \Omega \to S$
  piecewise linear map $f(t_j) = T_j$

17. The Spring Model

- replace edges by springs
- fix boundary vertices
- relaxation process
- energy of spring between p_i and p_j:

$$\frac{1}{2} D_{ij} s_{ij}^2$$

where spring constant $D_{ij} > 0$, spring length $s_{ij} = \|u_i - u_j\|$

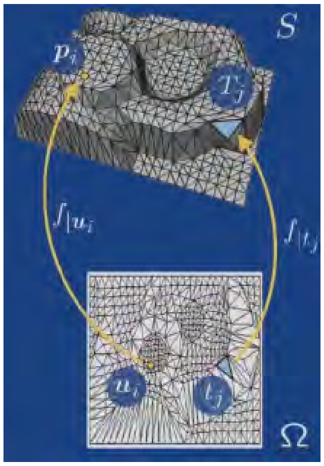

Figure II.9: Triangle Mesh Parameterization

- total energy

$$E = \sum_{(i,j)\in\epsilon} \frac{1}{2} D_{ij} \left\| u_i - u_j \right\|^2$$

18. Energy Minimization

- interior vertices $p_1, \ldots, p_n$
- p_i's neighbours p_j, $j \in N_i$
- overall spring energy

$$E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in N_i} \frac{1}{2} D_{ij} \left\| u_i - u_j \right\|^2$$

- partial derivative

$$\frac{\partial E}{\partial u_i} = \sum_{j \in N_i} D_{ij}(u_i - u_j)$$

- minimum of spring energy E

$$\sum_{j \in N_i} D_{ij} u_i = \sum_{j \in N_i} D_{ij} u_j$$

for all interior points $u_i, i = 1, \ldots, n$

- u_i is a convex combination of its neighbours u_j

$$u_i = \sum_{j \in N_i} \lambda_{ij} u_j$$

with weights $\lambda_{ij} = D_{ij} \Big/ \sum_{k \in N_i} D_{ik}$

19. The Linear System

- separation of vriables

$$u_i - \sum_{j \in N_i, j \le n} \lambda_{ij} u_j = \sum_{j \in N_i, j > n} \lambda_{ij} u_j = \bar{u}_i$$

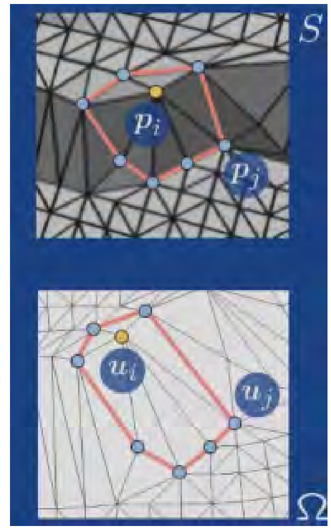

Figure II.10: Energy Minimization

- linear system

$$\begin{pmatrix} 1 & * & \cdots & -\lambda_{ij} \\ * & 1 & * & \vdots \\ \vdots & * & \ddots & * \\ -\lambda_{ji} & \cdots & * & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \vdots \\ \bar{u}_n \end{pmatrix}$$

- solve system twice

$$AU = \bar{U}$$
$$AV = \bar{V}$$

for u and v coordinates of interior parameter points

- matrix A is
  - sparse
  - diagonally dominant
  - nonsingular

20. Choice of Weights

- uniform spring constants
  $D_{ij} = 1, \lambda_{ij} = \frac{1}{\#N_i}$
- chordal spring constants
  $D_{ij} = \frac{1}{\|p_i - p_j\|}, \lambda_{ij} = \frac{D_{ij}}{\sum_{k=N_i} D_{ik}}$
- no fold-overs for convex boundary
- no linear reproduction
  planar meshes are distorted
- suppose S is a planar mesh
- specify weights λ_{ij} such taht

$$p_i = \sum_{j \in N_i} \lambda_{ij} p_i$$

- barycentric coordinates(质心坐标) of p_i
- then solving

$$u_i = \sum_{j \in N_i} \lambda_{ij} u_j$$

reproduces S

21. Barycentric Coordinates

22. Example

参数化后会折叠, 不好.

23. The Boundary Mapping

24. Segmentation and Constraints

24.1 Segmentation

Necessary for closed and high genus(高规格) meshes.
分片来做参数化
Goals: Large Charts → Low Distortion

- Single Charts
- Multiple Charts

e.g. Iso-charts(使用谱分析算法)

24.2 Constraints

Enforce specific point-to-point correspondences.
3D mesh and 2D mesh → constrained texture mapping
e.g. Texture Montage

III Example-Based Texture Synthesis

摸了

IV Differentiable Appearance Acquisition

可微分采集材质?

1. Introduction

渲染材质需要知道颜色, 反射率, 反射类型 (漫, 全), 反射函数 (输入观察角度, 输出颜色)

Light/material interaction: 吸收, 反射, 穿透

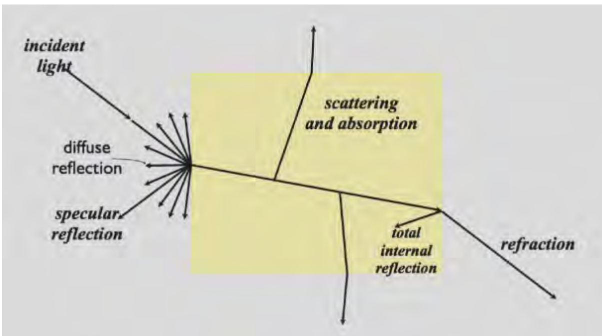

Figure IV.1: Light/material interaction

要数字化光的特性.

2. Reflection Models

2.1 BRDF

Bidirectional Reflection Distribution Function(双向反射率分布函数) $f(i, o)$, 四维函数 (只要方向)

若固定 i(入射), 函数退化为描述出射光的分布.

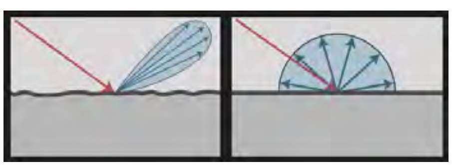

Figure IV.2: BRDF fixing i

2.2 Perfect Mirror Reflection

完美镜面反射 光的方向相反是为了计算少一个负号, 方

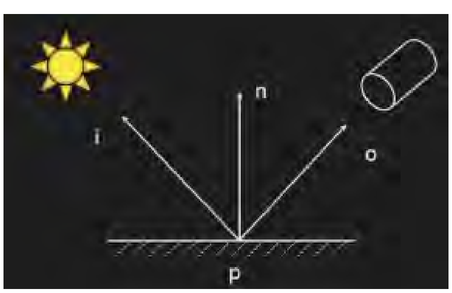

Figure IV.3: Perfect Mirror Reflection

便, 算是某种约定.

计算出射光方向, 再比较是否一致.

$$-r = -(i - \langle i, n\rangle n) + \langle i, n\rangle n = 2\langle i, n\rangle n - i$$

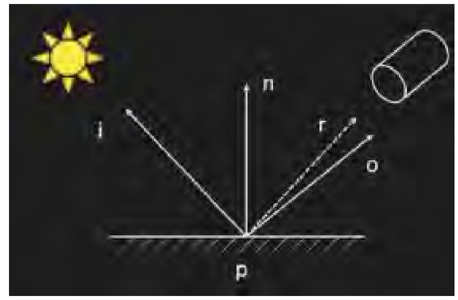

Figure IV.4: naive

计算半向量 (Half Vector),

$$h = \frac{i + o}{\|i + o\|}$$

判断其是否与 n 共线.

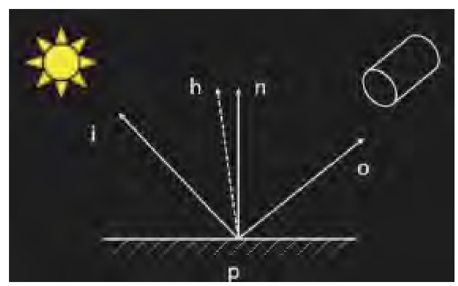

Figure IV.5: Half Vector

Fresnel Reflection Term 反射依赖于观察角度.

解麦克斯韦方程组求反射率, 也可以用逼近

Schilick's Approximation

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5$$

$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$

3. Microfacet-Based Models

所有真实世界的材料都可以由小镜子建模而成. 通过控制小镜子 (microfacet) 法向量的朝向控制材质反射.

$$f(i, o) = \frac{F(i, h)G(i, o, h)D(h)}{4(n, i)(n, o)}$$

- $F(i,h)$ Fresnel Reflection Term
- $D(h)$ 小镜子法向量分布函数, 发生 $n=h$ 镜面反射的小镜子的概率.
- $G(i,o,h)$ 考虑小镜子之间的遮挡 (Shadowing and Masking).

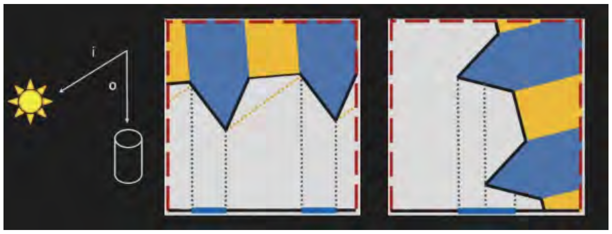

Figure IV.6: Shadowing and Masking

- 分母是依据视角的衰减项

3.1 *Example: Cook-Torrance Model*

$$D = \frac{e^{\frac{-\tan^2(\alpha)}{m^2}}}{\pi m^2 \cos^4(\alpha)}, \alpha = \arccos(n \cdot h)$$
$$G = \min\left(1, \frac{2(h \cdot n)(o \cdot n)}{o \cdot h}, \frac{2(h \cdot n)(i \cdot n)}{o \cdot h}\right)$$

仅需要一个 m 参数即可控制高光.

4. Anisotropic BRDF

各向异性的高光. 因为小尺度上材质的方向性.

5. Diffuse Reflection

漫反射. 光在各个方向上反射相同.

Lambertian model

$$f(i,o) = constant$$

这里仅仅是为了描述方便, 物理上难以实现.

$$f(i,o)(i,n) = \left[\frac{\rho_d}{\pi} + \rho_s \cdot f_{spec}(i,o)\right](i,n)$$

ρ_d, ρ_s 是 diffuse/specular coefficient

6. Measured Data

使用函数列表存储. 优点: 真实, 缺点: 开销大.

BRDF Explorer

7. Differentiable Acquisition of Visual Information

end-to-end(端到端)

深度学习的端到端需要物理信息到最后的结果. 软硬件一起考虑. 所以说是可微分的数据获取 (Differentiable Acquisition), 可以通过梯度优化.

数据的信息量很多, 想少采集必须使用强先验补足信息.

用神经网络的参数控制现实世界.(这里是控制灯的亮度) 具体来说就是优化参数, 也等价于优化控制.

用已有数学模型生成训练数据.

7.1 *Efficient Reflectance Capture Using an Autoencoder*

一作是本科生, 应该大三才开始做的, 这篇文章拿到了一个很好的奖, 听下来确实非常精妙, 很有收获.

文章的目的是为了快速重建材质的 BRDF 函数.

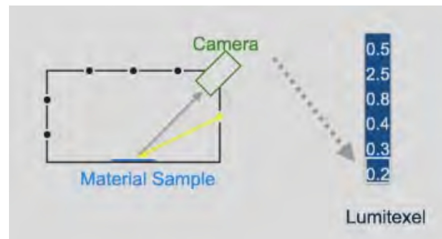

Figure IV.7: device

Lumitexel 是一个像素点, 每个灯的角度获得的 BRDF 输出的集合, 包含了最多的信息.

naive 方法 就是使用打表的方法记录现实的 BRDF, 即记录每个像素的 Lumitexel. 开启一个灯珠, 拍一张照, 获得 BRDF 的一个角度的输出, 换一个灯珠如此重复, 直到获得所有灯珠角度的输出, 就算记录了 BRDF 函数.

缺点: 开销大, 时间长. 灯珠有近 10000 个.

论文方法 以某种方式组合灯珠, 获得一些观测值 (32 个左右), 再通过这些值重建出原本的 Lumitexel. 组合方式与重建方式通过神经网络训练而来.

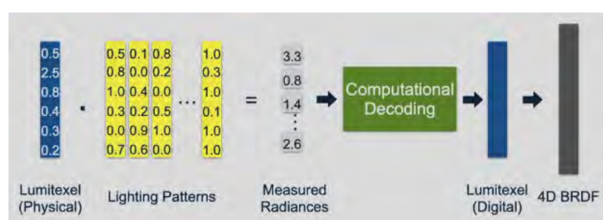

Figure IV.8: Pipeline

原理是灯亮度与那个像素 Lumitexel 的点积就是观测值, 这样求 Lumitexel 就相当于解线性方程组, 但是妙就妙在方程组的系数是通过学习而来的. 他用第一层参数控制了灯组合方式. 这样训练时就可以调整组合, 获得更好的数据.

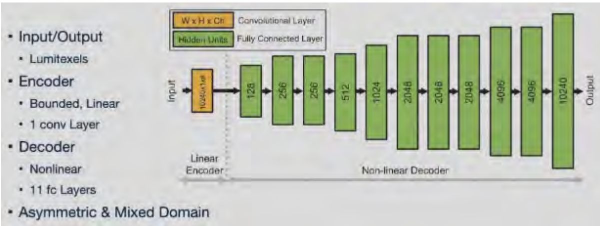

Figure IV.9: L-DAE

训练数据的来源于成熟的 BRDF 函数估计与拟合公式, 从经典材质上训练的网络参数可以去采集其他材质, 做到很好的重建.

V Introduction to 3D Printing in Computer Graphics

3D Printing is a type of manufacturing technologies.
等材制造. 就是开模具, 然后铸造.
减材制造. 切削加工.
增材制造. 3D 打印.

1. 类型

- FDM
- SLA
- SLS
- 4D Printing

VI Differential Domain Shape Deformation

1. Shape Deformation

对三维内容创造的 useful tool

- FFD: freeform deformation (自由形变)
  最早的 deformation

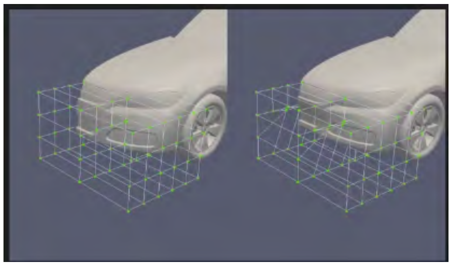

Figure VI.1: FFD

- Multi-resolution editing (多分辨率编辑)

Figure VI.2: Multi-resolution editing

- Differential domain methods (微分域方法)

2. Differential domain methods

(做研究首先要确定目标)

2.1 Goal

- High quality: smooth deformation, detail preservation
- Easy manipulation: anchor-based, sketch-based
- Useful constraints: volume, skeleton, projection, ...
- Popular inputs: subdivsion, skinned mesh, man-made
- Interactive speed: nonlinear, optimization, GPU

2.2 Detail preservation

Detail:

gradients:

$$\nabla F = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right)$$
$$\nabla F = F_1\psi_1 + F_2\psi_2 + F_3\psi_3$$

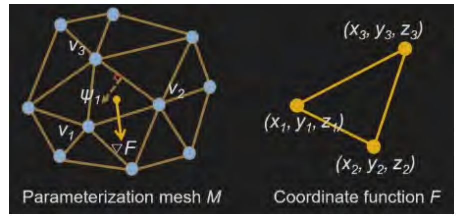

Figure VI.3: gradients

Manipulate gradients:

$$G = F_1'\psi_1 + F_2'\psi_2 + F_3'\psi_3$$
$$\min_F \int_M \|\nabla F - G\|^2$$

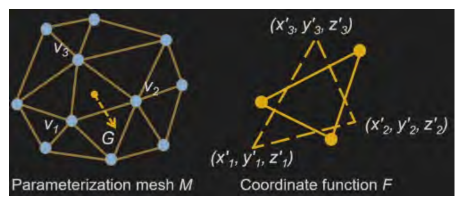

Figure VI.4: Manipulate gradients

Poisson equation:

$$\Delta F = \mathrm{div} G \text{ with } F|_\Omega = F^*|_\Omega$$
$$\Delta F(v_i) = v_i - \sum_{j=1}^{n_i} w_{ij} v_{i,j}$$
$$\mathrm{div} G(v_i) = \sum_{t_k \in N(v_i)} \rho_{ki} \cdot G(t_k)|t_k|$$
$$AF = b$$

Δ is Laplacian, div is Divergence.

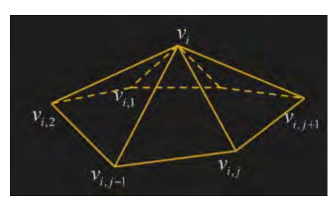

Figure VI.5: Poisson equation

Poisson mesh editing:

- Sommthly changing gradients: 因为从梯度到坐标点, 可以保证坐标变换的光滑
- Global optimization: 平均分布的误差

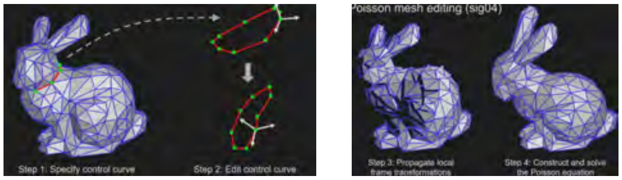

Figure VI.6: Poisson mesh editing

Laplacian surface editing:

$$\delta_i = L(v_i) = v_i - \sum_{j=1}^{n_i} w_{ij} v_{i,j}$$

$$L(v_i) = \delta_i$$

$$AV = b$$

Preserving surface details is not enough: Bending and Twisting

已经昏了, 不想记了, 开摆!

非线性能量函数优化:

1) 子空间: 在在空间求解
2) 分段迭代: 先迭代易收敛的, 再以其为初始值迭代其他的
3) 瀑布求解: 先加基本约束, 再逐渐加其他约束.

VII GPU 并行计算

1. GPU: Graphics Processing Unit

Originally designed for games.

From simple, fixed, pipline to complex, highly, programmable.

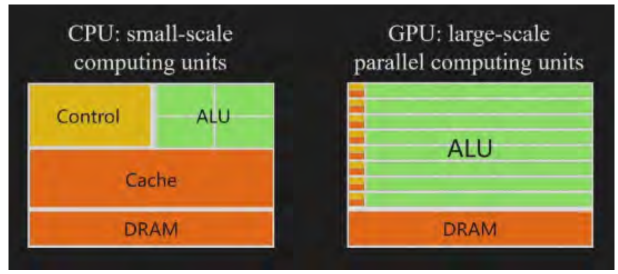

Figure VII.1: GPU v.s. CPU

2. BSGP: bulk-synchronous GPU programming

批量同步 GPU 编程语言.

2.1 GPGPU Programming Languages

Stream processing 流式处理.

- CUDA
- OpenCL
- DirectX11, Compute Shader

2.2 Stream Processing Model

数据就是流 (stream), 流式处理就是以数据为中心.

核函数 (kernel), 以流为输入, 核中的指令会并行的作用在流中每一个元素上, 然后形成输出流.

每一个核函数, 会被装载到 GPU 芯片上.

GPU 本质就是以数据为中心的流式处理器.

Problems 支持高性能, 但是需要好的代码, 让码农头秃. 主要体现在以下方面:

- 影响可读性与可维护性.

  会把语义不相关的代码放在一起, 进行优化, 让代码难以修改.
- 需要手动管理数据流.

  即需要循环利用流, 容易出错.
- 代码重用性低

  因为核函数是高度特化的, 不能拿过来直接用.

2.3 BSGP

基于 BSP. 可以以类序列的方式编程.

Example one-ring neighborhood

计算每个顶点所共享的三角形. 假设有 n 个三角形.

1) 对 3n 个整数排序, 每个三角形会生成三个 32bits 整数, 每个整数对应一个顶点. 整数高 16 位就是顶点 index, 低 16 位就是三角形 index.
2) 排序后, 求出顶点的 index 分隔.

Code 1: BSGP

```
findFaces(int* pf, int* hd, int* ib, int n){
    spawn(n*3){
        rk = thread.rank;   //face id
        f = rk/3;           // vertex id
        v = ib[rk];
        thread.sortby(v);
        // allocate a temp list
        require
            owner = detmpnew[n]int;
        rk = thread.rank;
        pf[rk] = f;
        owner[rk] = v;
        barrier;
        if(rk == 0 || owner[rk-1] != v)
            hd[v] = rk;
    }
}
```

CUDA version 略过.

2.4 BSGP Language Constructs

结构

关键字

会写 complier 会越来越有用, 会写的人也越来越少.

3. BSGP debuger & GPU interrupt

以数据流为中心的调试系统.

提出 GPU 中断, 让 GPU 可以调用 CPU 函数, 以支持数据流的存储.

4. Data structures

- Octrees 八叉树, 用以并行的曲面重建, 用点构造 mesh
- KD-Trees
- 6D Spatial Hierarchies

5. Applications

摸了

CG 技术简介可微渲染技术, 从真实世界的图像反推参数. mocap 动作捕捉技术.

VIII Rendering

1. Introduction

摸了

IX Determinative vs Probabilisitc

以抠图与去模糊为例.

1. Matting

1.1 Compositing

抠图的目的就是合成.

- F 前景
- α 透明度
- B 背景
- C 合成的图像

合成公式:

$$C = \alpha F + (1-\alpha)B$$

1.2 Matting

从 C 分解出 F, α, B. 不是唯一解, 病态问题.

Three approaches:

1) 减少未知数
2) 增加观测 (约束)
3) 增加先验 (priors)

reduce # unknowns difference matting, i.e. known B. 用 $C-B$ 得到一个背景的 α. 但背景复杂, 所以使用简单的 B, 保障 α 的干净. 但纯色的 B 难以获得.

add observations 增加不同颜色的 B.

add priors 若是自然图像, 只能使用这种方法.

1) 用轮廓线算法分割前景与背景, 但轮廓线周围的像素比较 ambiguous. 把图像分成三个区域 (trimap), 必定前景/背景, 与轮廓线周围的模糊区域.
2) Bayesian framework

1.3 Bayesian framework

最大化后验概率.

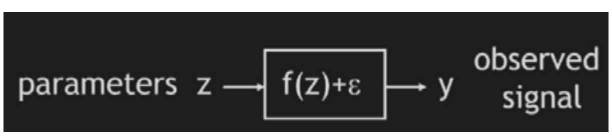

Figure IX.1: Bayesian framework

这里 ϵ 为噪声.

复习下贝叶斯定理:

$$P(z|y)P(y) = P(y|z)P(z)$$

$$\begin{aligned} z^* &= \max_z P(z|y) \\ &= \max_z \frac{P(y|z)P(z)}{P(y)} \\ &= \max_z L(y|z) + L(z) \end{aligned}$$

这里 $L(z) = \log(P(z))$. 因为 $L(y)$ 不影响 z^*, 所以舍弃了.

假设 $P(y|z)$ 满足正态分布:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

那么

$$L(y|z) = \frac{\|y - f(z)\|^2}{\sigma^2}$$

表示 data evidence(数据置信度). $L(z)$ 需要使用另一个先验.

目标问题:

$$\begin{aligned} &\underset{F,B,\alpha}{\operatorname{argmax}} P(F,B,\alpha|C) \\ &\quad = \underset{F,B,\alpha}{\operatorname{argmax}} \frac{P(C|F,B,\alpha)P(F)P(B)P(\alpha)}{P(C)} \\ &\quad = \underset{F,B,\alpha}{\operatorname{argmax}} L(C|F,B,\alpha) + L(F) + L(B) + L(\alpha) \end{aligned}$$

这里 $P(C|F,B,\alpha)$ 为 likehood, $P(F)P(B)P(\alpha)$ 为 priors, $P(F,B,\alpha|C)$ 为 posterior proability.

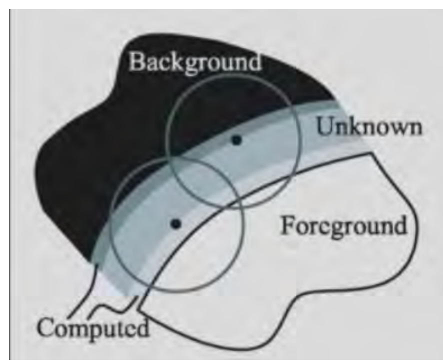

Figure IX.2: $L(F)$ 与 $L(B)$

且

$$\begin{aligned} L(C|F,B,\alpha) &= -\frac{\|C - \alpha F - (1-\alpha)B\|^2}{2\sigma_C^2} \\ L(F) &= -(F-\bar{F})^\top \Sigma_F^{-1}(F-\bar{F})/2 \\ L(B) &= -(B-\bar{B})^\top \Sigma_B^{-1}(B-\bar{B})/2 \end{aligned}$$

Σ_F, Σ_B 为协方差矩阵. 忽略 $L(\alpha)$.

求解:

1) 先固定 α 求解 F, B
2) 再固定 F, B 求解 α

迭代直到收敛.

最后相当于用概率论, 结合已知的先验, 对 F, B, α 进行约束, 让其结果符合先验.

2. Image Deblurring

2.1 Different Types of Blur

三种类型:

1) 运动模糊 (Scene motion)
2) 景深 (Defocus blur)
3) 相机抖动 (Camera shake)

对于相机抖动, 定义 Convolution 模型.

Figure IX.3: Convolution

目标: 给予 Blurry image, 获得 Sharp image. 假设静态的场景.

基本上解法分两类: blind v.s. non-blind deconvolution.

- non-blind: 已知 blur kernel
- blind: 都未知

前人的工作有很多的前提假设:

- 需要多个图像
- 相机运动是简单的

即使是 non-blind, 问题仍是病态的.

2.2 Natural Images Priors

用梯度分布作为先验, 在自然界中, 图像的梯度是稀疏的.

$$x = \arg\min \underbrace{\lfloor f \otimes x - y \rfloor^2}_{\text{Convolution error}} + \lambda \underbrace{\sum_i \rho(\nabla x_i)}_{\text{Derivatives prior}}$$

Figure IX.4: deconvolution with priors

然后就是讲论文了

2.3 用一张图消除相机抖动

需要三种信息:

1) 重建约束
2) 图像先验 (图片梯度的分布)
3) 模糊先验 (正 + 稀疏)

然后就是推公式, 每个值是怎么来的. 摸了!
有问题, 因为先验还是不好, 去模糊后会有噪声.

2.4 如何做高质量的去模糊

框架无改变. 引入了图像局部约束, 在模糊图像中, 小梯度的区域, 在清晰图像中, 梯度应该也很小.

优化的公式涉及卷积, 求解时开销巨大. 使用变量替换的技巧把卷积独立. 还做了个傅里叶变换, 把卷积操作变为乘法操作.

X Digital Avatars for All: Interactive Face and Hair

如何做选题.

1. Overview

需要关注:

- 人脸
- 表情
- 头发
- 逼真的人脸和头发 (电影中)
- 游戏中的人类和头发

2. Hair Modeling and Animation

2.1 Single-view Hair Modeling

From Single-view image to 3D hair model.
Visual constrains:

- 与原图片匹配
- 逼真的新视角

Physical constrains:

- 头发是从头皮上张出来的
- 光滑连续

3. Face tracking and Animation

XI Research as a Career