Volumetric Image Visualization

Alexandre Xavier Falcão

LIDS - Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

Introduction

In this lecture, we will learn how to

- quickly estimate a normal vector during the surface rendering algorithm, and
- encode a normal vector $o.\mathbf{n}(p')$, $p' \in \mathcal{S}$, by using only two bytes per boundary voxel.

In this lecture, we will learn how to

- quickly estimate a normal vector during the surface rendering algorithm, and

- encode a normal vector $o.\mathbf{n}(p')$, $p' \in \mathcal{S}$, by using only two bytes per boundary voxel.

- The lecture will also cover the creation of stereo renditions by surface rendering.

Distance and index buffers

Instead of directly obtaining the Phong's shading $r(p')$ as a result of ray casting from each pixel $p \in D_J$, one can quickly generate distance $\hat{D} = (D_J, D)$ and index $\hat{V} = (D_J, V)$ buffers from ray casting and use them to compute $\hat{J} = (D_J, \mathbf{J})$, $\mathbf{J}(p) \leftarrow r(p')(o.r_1, o.r_2, o.r_3)$, afterwards.

$$r(p') \quad = \quad k_a r_a + r_d(p')\left(k_d \cos(\theta) + k_s \cos^{n_s}(2\theta)\right)$$

Distance and index buffers

Instead of directly obtaining the Phong's shading $r(p')$ as a result of ray casting from each pixel $p \in D_J$, one can quickly generate distance $\hat{D} = (D_J, D)$ and index $\hat{V} = (D_J, V)$ buffers from ray casting and use them to compute $\hat{J} = (D_J, \mathbf{J})$, $\mathbf{J}(p) \leftarrow r(p')(o.r_1, o.r_2, o.r_3)$, afterwards.

$$r(p') = k_a r_a + r_d(p')\left(k_d \cos(\theta) + k_s \cos^{n_s}(2\theta)\right)$$

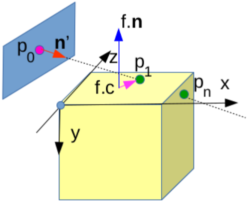
The distance map D can be used to estimate the depth shading $r_d(p')$ while the index map V can be used to estimate the normal vector at p' (i.e., to obtain θ).

Depth shading

The depth map stores the distance $D(p) = \|p' - p_0\|$, $p_0 = \phi^{-1}(p)$, between the viewing plane and the surface point p' that is reached from $p \in D_J$. The depth shading can be defined as

$$r_d(p') = H\frac{d_{\max} - \color{red}{D(p)}}{d_{\max} - d_{\min}},$$

where $d_{\min}$ and $d_{\max}$ are the minimum and maximum values in D.



Normal estimation from the index buffer

The index map stores the closest voxel
$V(p) = p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \in \mathcal{S} \subset D_I$ reached from $p \in D_J$.

Normal estimation from the index buffer

The index map stores the closest voxel $V(p) = p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \in \mathcal{S} \subset D_I$ reached from $p \in D_J$.

- This allows normal estimation in the case of opaque objects only.

Normal estimation from the index buffer

The index map stores the closest voxel $V(p) = p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \in \mathcal{S} \subset D_I$ reached from $p \in D_J$.

- This allows normal estimation in the case of opaque objects only.

- One has to be careful to avoid computations with voxels from distinct objects and distant parts of the same object.

Normal estimation from the index buffer

The index map stores the closest voxel $V(p) = p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \in \mathcal{S} \subset D_I$ reached from $p \in D_J$.

- This allows normal estimation in the case of opaque objects only.
- One has to be careful to avoid computations with voxels from distinct objects and distant parts of the same object.
- This technique might create artifacts, but it performs a good job in most cases.

Normal estimation from the index buffer

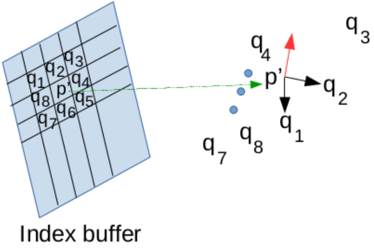
The index map stores the closest voxel $V(p) = p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \in \mathcal{S} \subset D_I$ reached from $p \in D_J$.

- This allows normal estimation in the case of opaque objects only.
- One has to be careful to avoid computations with voxels from distinct objects and distant parts of the same object.
- This technique might create artifacts, but it performs a good job in most cases.
- It also allows direct access to p' when the user clicks on a pixel p in the rendition $\hat{J} = (D_J, \mathbf{J})$.

Normal estimation from the index buffer

Let $\mathcal{A}_r(p') = \{q \in \mathcal{S} \mid \|q - p'\| \leq r, L(q) = L(p')\}$ be an ordered set obtained from the 8-neighbors of $p \in D_J$ in $\hat{V}$ as visited in a counter-clockwise order. The normal vector $\mathbf{n}(p')$ can be obtained from the subsequent spels $q, q' \in \mathcal{A}_r(p') \mid q \prec q'$, as

$$\mathbf{n}(p') = \frac{1}{|\mathcal{A}_r(p')|} \sum_{\forall q,q' \in \mathcal{A}_r(p') \mid q \prec q'} \frac{q - p'}{\|q - p'\|} \times \frac{q' - p'}{\|q' - p'\|}$$



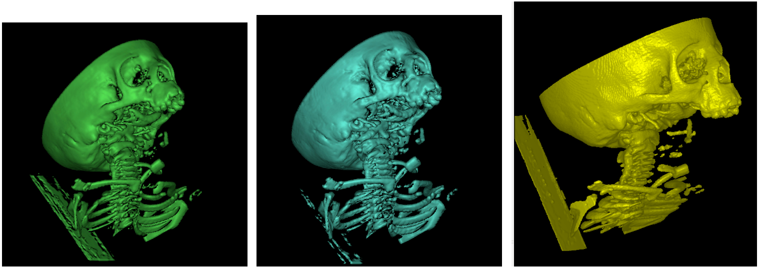
The distance $r > 0$ should be a small value.

Revisiting the surface rendering algorithm

$\mathbf{n}' \leftarrow \phi_r^{-1}(\mathbf{n})$, where $\mathbf{n} = (0, 0, 1, 0)$.
For each $p \in D_J$ do
    $p_0 \leftarrow \phi^{-1}(p)$.
    Find $\mathcal{P} = \{p_1, p_n\}$ by solving $\langle p_0 + \lambda \mathbf{n}' - f.c, f.\mathbf{n} \rangle = 0$ for each face $f \in \mathcal{F}$ of the scene, whenever they exist.
    if $\mathcal{P} \neq \emptyset$ then
        $p' \leftarrow \text{FindSurfacePoint}(\hat{L}, \mathcal{P})$.
        if $p' \neq nil$ then $D(p) \leftarrow \|p' - p_0\|$ and $V(p) \leftarrow p'$.
Compute $d_{\min} = \min_{p \in D_J}\{D(p) > 0\}$ and $d_{\max} = \max_{p \in D_J}\{D(p)\}$.
For each $p \in D_J$ do
    Find $o.\mathbf{n}(p')$, where $V(p) = p'$, and $r_d(p')$ from $D(p)$.
    $\mathbf{J}(p) \leftarrow \mathbf{c}(p')$ by Phong's model.

Comparison among the three approaches

Note that, for a same set $\mathcal{S}$, the index-based normal estimation (right) shows less holes than the scene-based (left) and object-based (center) approaches, because it might be using parts of the internal boundary near the external one.



Comparison among the three approaches

Note that, for a same set $\mathcal{S}$, the index-based normal estimation (right) shows less holes than the scene-based (left) and object-based (center) approaches, because it might be using parts of the internal boundary near the external one.

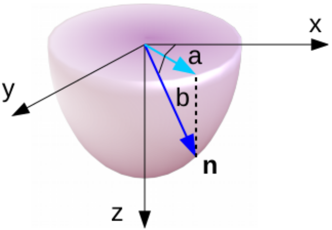
Comparison among the three approaches

Note that, for a same set $\mathcal{S}$, the index-based normal estimation (right) shows less holes than the scene-based (left) and object-based (center) approaches, because it might be using parts of the internal boundary near the external one.

Encoding the normal vectors

Let a unit vetor $\mathbf{n} = (n_x, n_y, n_z)$ be written in polar coordinates

$$
\begin{aligned}
n_x &= \cos(b)\cos(a) \\
n_y &= \cos(b)\sin(a) \\
n_z &= \sin(b)
\end{aligned}
$$



One may create a look-up table with unit vectors by varying $a = 0, 1, \ldots, 359$ and $b = -90, -89, \ldots, 90$, as follows.

Let $\mathbf{n}[i] = (n_x[i], n_y[i], n_z[i])$ be a look-up table of possible unit normal vectors.

$i \leftarrow 1$
For $b \leftarrow -90$ to 90 do
  $\gamma \leftarrow b\frac{\pi}{180.}$
  For $a \leftarrow 0$ to $a = 359$ do
    $\alpha \leftarrow a\frac{\pi}{180.}$
    $n_x[i] \leftarrow \cos(\gamma)\cos(\alpha)$
    $n_y[i] \leftarrow \cos(\gamma)\sin(\alpha)$
    $n_z[i] \leftarrow \sin(\gamma)$
    $i \leftarrow i + 1$

Encoding the normal vectors

Let $\mathbf{n}[i] = (n_x[i], n_y[i], n_z[i])$ be a look-up table of possible unit normal vectors.

$i \leftarrow 1$
For $b \leftarrow -90$ to 90 do
    $\gamma \leftarrow b\frac{\pi}{180.}$
    For $a \leftarrow 0$ to $a = 359$ do
        $\alpha \leftarrow a\frac{\pi}{180.}$
        $n_x[i] \leftarrow \cos(\gamma)\cos(\alpha)$
        $n_y[i] \leftarrow \cos(\gamma)\sin(\alpha)$
        $n_z[i] \leftarrow \sin(\gamma)$
        $i \leftarrow i + 1$

The idea is to store one index in 2 bytes per boundary voxel $p' \in \mathcal{S}$.

Encoding the normal vectors

Therefore, for each voxel $p' \in \mathcal{S}$ of each object o,

- one can estimate its unit normal vector $o.\mathbf{n}(p')$,
- find the index i^* of its closest unit normal $\mathbf{n}$ in the look-up table,

$$i^* \quad = \quad arg\max_i \{\langle o.\mathbf{n}(p'), \mathbf{n}[i] \rangle\},$$

- and store the index i^* to have fast access to its surrogate $\mathbf{n}[i^*]$ when needed for the Phong's illumination model.

Finally, given two viewing angles α (tilt) and β (spin). One can create a stereo rendition $\hat{J} = (D_J, \mathbf{J})$ as follows.

- The view of the left eye using $(\alpha, \beta - 5)$ is assigned to the red component of $\mathbf{J}$.
- The view of the right eye using $(\alpha, \beta + 5)$ is assigned to the blue component of $\mathbf{J}$.
- The average between the two views is assigned to the green component of $\mathbf{J}$.