Volumetric Image Visualization

Alexandre Xavier Falcão

LIDS - Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

Phong's illumination model and surface rendering

- After 3D object segmentation, one can simulate the visualization of 3D objects in the scene from any viewing point.

Phong's illumination model and surface rendering

- After 3D object segmentation, one can simulate the visualization of 3D objects in the scene from any viewing point.

- We will assume that the observer is distant enough and behind the viewing plane such that the rendition technique, called surface rendering, is obtained by orthogonal projection.

Phong's illumination model and surface rendering

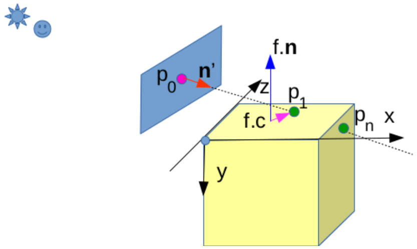
- After 3D object segmentation, one can simulate the visualization of 3D objects in the scene from any viewing point.
- We will assume that the observer is distant enough and behind the viewing plane such that the rendition technique, called surface rendering, is obtained by orthogonal projection.
- Surface rendering usually adopts the Phong's illumination model and, to avoid undesirable shadows, we will assume a single light source at the same position of the observer.

Phong's illumination model and surface rendering

- After 3D object segmentation, one can simulate the visualization of 3D objects in the scene from any viewing point.
- We will assume that the observer is distant enough and behind the viewing plane such that the rendition technique, called surface rendering, is obtained by orthogonal projection.
- Surface rendering usually adopts the Phong's illumination model and, to avoid undesirable shadows, we will assume a single light source at the same position of the observer.
- This lecture covers the surface rendering algorithm used to obtain color renditions $\hat{J} = (D_J, \mathrm{J})$.

Phong's illumination model and surface rendering

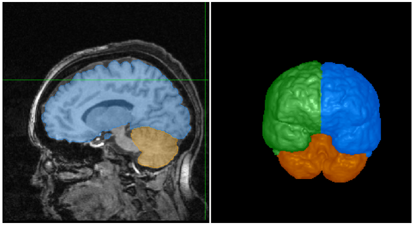
For each pixel $p \in D_J$, the algorithm applies a viewing transformation followed by ray casting.



Assuming that the objects are opaques in a label scene $\hat{L} = (D_I, L)$ from image segmentation, each ray will stop at a first voxel $p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \in D_I$, such that $L(p') \neq 0$ is an object label in $\{1, 2, \ldots, c\}$. The Phong's model then computes the reflected color $\mathrm{J}(p) \leftarrow \mathrm{c}(p')$ at the surface point $p' = (x_{p'}, y_{p'}, z_{p'})$.

Phong's illumination model and surface rendering

For each pixel $p \in D_J$, the algorithm applies a viewing transformation followed by ray casting.



Assuming that the objects are opaques in a label scene $\hat{L} = (D_I, L)$ from image segmentation, each ray will stop at a first voxel $p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \in D_I$, such that $L(p') \neq 0$ is an object label in $\{1, 2, \ldots, c\}$. The Phong's model then computes the reflected color $\mathrm{J}(p) \leftarrow \mathrm{c}(p')$ at the surface point $p' = (x_{p'}, y_{p'}, z_{p'})$.

Phong's illumination model and surface rendering

In addition to the transformations $\phi_r^{-1} = \mathrm{R}_x(-\alpha)\mathrm{R}_y(-\beta)$,

$$\phi^{-1} \quad = \quad \mathrm{T}(x_c, y_c, z_c)\phi_r^{-1}\mathrm{T}(\frac{-d}{2}, \frac{-d}{2}, \frac{-d}{2}),$$

and the scene's face set $\mathcal{F}$, with normal $f.\mathrm{n}$ and center $f.c$ for each face $f \in \mathcal{F}$, the graphical context must include

Phong's illumination model and surface rendering

In addition to the transformations $\phi_r^{-1} = \mathrm{R}_x(-\alpha)\mathrm{R}_y(-\beta)$,

$$\phi^{-1} \quad = \quad \mathrm{T}(x_c, y_c, z_c)\phi_r^{-1}\mathrm{T}(\frac{-d}{2}, \frac{-d}{2}, \frac{-d}{2}),$$

and the scene's face set $\mathcal{F}$, with normal $f.\mathrm{n}$ and center $f.c$ for each face $f \in \mathcal{F}$, the graphical context must include

- the color reflectance $o.\mathrm{r} = (o.R, o.G, o.B)$ of each object o in the RGB color space, and
- the parameters of the Phong's model.

Phong's illumination model and surface rendering

In addition to the transformations $\phi_r^{-1} = \mathrm{R}_x(-\alpha)\mathrm{R}_y(-\beta)$,

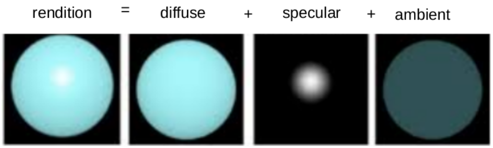
$$\phi^{-1} \quad = \quad \mathrm{T}(x_c, y_c, z_c)\phi_r^{-1}\mathrm{T}(\frac{-d}{2}, \frac{-d}{2}, \frac{-d}{2}),$$

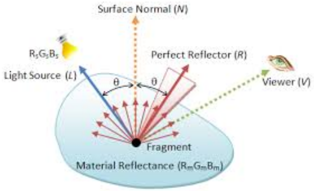
and the scene's face set $\mathcal{F}$, with normal $f.\mathrm{n}$ and center $f.c$ for each face $f \in \mathcal{F}$, the graphical context must include

- the color reflectance $o.\mathrm{r} = (o.R, o.G, o.B)$ of each object o in the RGB color space, and
- the parameters of the Phong's model.
- Another information that can be stored in the graphical context is the normal vector at the surface points of each object.

Phong's illumination model

The light reflected towards the observer results from the combination of uniform ambient light reflection with the diffuse and specular reflections at each visible surface point.



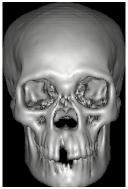


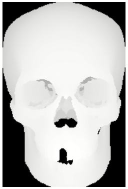
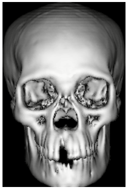
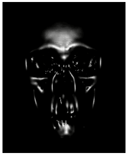
Recall that we have simplified this model by considering observer and light source at the same position, as follows.

Phong's illumination model

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

where $k_a + k_d + k_s = 1$ are ambient, diffuse, and specular constants, n_s depends on the material, $r_d(p')$ is a depth shading at p', and θ is the angle between $-\mathrm{n}' = -\phi_r^{-1}(\mathrm{n})$ and the normal vector $o.\mathrm{n}(p')$ at the surface point p'.


(a)

(b)

(c)

(d)

(e)

(a) $r(p')$, (b) $k_a r_a$, (c) $r_d(p')$, (d) $r_d(p')k_d \cos(\theta)$, and (e) $r_d(p')k_s \cos^{n_s}(2\theta)$.

Implementation issues

- A reasonable parameter choice is $k_a = 0.1$, $k_s = 0.2$, $k_d = 0.7$, $n_s = 5$, and $r_a = H$ (the maximum intensity value).

Implementation issues

- A reasonable parameter choice is $k_a = 0.1$, $k_s = 0.2$, $k_d = 0.7$, $n_s = 5$, and $r_a = H$ (the maximum intensity value).
- The reflected light $r(p')$ is computed only when $0 \leq \theta < \frac{\pi}{2}$.

Implementation issues

- A reasonable parameter choice is $k_a = 0.1$, $k_s = 0.2$, $k_d = 0.7$, $n_s = 5$, and $r_a = H$ (the maximum intensity value).
- The reflected light $r(p')$ is computed only when $0 \leq \theta < \frac{\pi}{2}$.
- The specular component is computed only when $0 \leq \theta < \frac{\pi}{4}$.

Implementation issues

- The depth shading is $r_d(p') = H(1.0 - \frac{\|p'-p_0\|-d_{\min}}{(d_{\max}-d_{\min})})$, where $p_0 = \phi^{-1}(p)$, and $d_{\min}$ and $d_{\max}$ are the minimum and maximum distances between boundary voxels and the center of the viewing plane after transformation.

Implementation issues

- The depth shading is $r_d(p') = H(1.0 - \frac{\|p'-p_0\|-d_{\min}}{(d_{\max}-d_{\min})})$, where $p_0 = \phi^{-1}(p)$, and $d_{\min}$ and $d_{\max}$ are the minimum and maximum distances between boundary voxels and the center of the viewing plane after transformation.

- Techniques to estimate the normal vector $o.\mathrm{n}(p')$ at each surface point p' of an object o will be covered in the next lecture.

Implementation issues

- The depth shading is $r_d(p') = H(1.0 - \frac{\|p'-p_0\|-d_{\min}}{(d_{\max}-d_{\min})})$, where $p_0 = \phi^{-1}(p)$, and $d_{\min}$ and $d_{\max}$ are the minimum and maximum distances between boundary voxels and the center of the viewing plane after transformation.

- Techniques to estimate the normal vector $o.\mathrm{n}(p')$ at each surface point p' of an object o will be covered in the next lecture.

- The assigned color $\mathrm{J}(p) \leftarrow \mathrm{c}(p') = r(p') \times o.\mathrm{r}$, where $o.\mathrm{r} = (o.R, o.G, o.B)$ is the color reflectance of object o.

Surface rendering algorithm

Input : $\hat{L} = (D_I, L)$, and angles α and β.
Output: $\hat{J} = (D_J, \mathrm{J})$.

```
1 $\mathrm{n}' \leftarrow \phi_r^{-1}(\mathrm{n})$, where $\mathrm{n} = (0,0,1,0)$.
2 For each $p \in D_J$ do
3     $p_0 \leftarrow \phi^{-1}(p)$.
4     Find $\mathcal{P} = \{p_1, p_n\}$ by solving $\langle p_0 + \lambda \mathrm{n}' - f.c, f.\mathrm{n} \rangle = 0$
      for each face $f \in \mathcal{F}$ of the scene, whenever they exist.
5     if $\mathcal{P} \neq \emptyset$ then
6         $p' \leftarrow$ FindSurfacePoint$(\hat{L}, \mathcal{P})$.
7         if $p' \neq nil$ then $\mathrm{J}(p) \leftarrow \mathrm{c}(p')$ by Phong's model.
```

Algorithm to find a surface point p'

Input : $\hat{L} = (D_I, L)$ and $\mathcal{P} = \{p_1, p_n\}$.
Output: Point p' or *nil* for no intersection.

```
1 If $p_1 = p_n$ then set $n \leftarrow 1$.
2 Else
3     Set $D_x \leftarrow x_{p_n} - x_{p_1}$, $D_y \leftarrow y_{p_n} - y_{p_1}$, $D_z \leftarrow z_{p_n} - z_{p_1}$.
4     If $|D_x| \geq |D_y|$ and $|D_x| \geq |D_z|$ then
5         Set $n \leftarrow |D_x| + 1$, $d_x \leftarrow sign(D_x)$, $d_y \leftarrow \frac{d_x D_y}{D_x}$, and
          $d_z \leftarrow \frac{d_x D_z}{D_x}$.
6     Else
7         If $|D_y| \geq |D_x|$ and $|D_y| \geq |D_z|$ then
```

Algorithm to find a surface point p'

```
8        Set $n \leftarrow |D_y| + 1$, $d_y \leftarrow sign(D_y)$, $d_x \leftarrow \frac{d_y D_x}{D_y}$, and
         $d_z \leftarrow \frac{d_y D_z}{D_y}$.
9      Else
10       Set $n \leftarrow |D_z| + 1$, $d_z \leftarrow sign(D_z)$, $d_x \leftarrow \frac{d_z D_x}{D_z}$, and
         $d_y \leftarrow \frac{d_z D_y}{D_z}$.
11 Set $p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \leftarrow (x_{p_1}, y_{p_1}, z_{p_1})$.
12 If $L(p') \neq 0$, then return $p' \leftarrow (x'_p, y'_p, z'_p)$.
13 For each $k = 2$ to $n$, do
14     $p' = (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil) \leftarrow (x_{p'}, y_{p'}, z_{p'}) + (d_x, d_y, d_z)$
15     If $L(p') \neq 0$, then
           return $p' \leftarrow (x'_p, y'_p, z'_p)$.
16 return $p' \leftarrow nil$.
```

An additional implementation issue

- If you have pre-stored the normal vectors for boundary voxels only and do not want to miss the boundary during ray casting (avoid holes in the rendition), voxel p' can be detected as the closest to that boundary.

An additional implementation issue

- If you have pre-stored the normal vectors for boundary voxels only and do not want to miss the boundary during ray casting (avoid holes in the rendition), voxel p' can be detected as the closest to that boundary.

- This requires to replace Lines 12 and 15 by

```
For each $q \in \mathcal{A}_1(p')$, do
   If $L(q) \neq 0$, then
      return $p'$.
```