CS4533 Lecture 9-10 Slides/Notes

Shading and Illumination (Notes, Ch 14)

By Prof. Yi-Jen Chiang
CSE Dept., Tandon School of Engineering
New York University

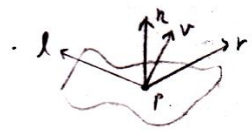
1

* **Phong Reflection Model** : 3 types of (material-light interactions) (reflections) : 1. ambient, 2. diffuse, 3. specular

(local ~~shading~~ lighting model)

(3 components of the model)

* **Vectors used by the Phong model** :



p : point on the surface we want to shade

n : surface normal at p

l : vector from p to the light source in question.

v : " p to the viewer

r : vector of a perfectly reflected ray from l.
(computed from l and n)

* Each light source L has 3 terms for light-material interactions:
  - ambient color $L_a = \{0, 0, 0, 0\}$ (R G B A)
  - diffuse color $L_d = \{0, 0, 0, 0\}$
  - specular : $L_s = \{0, 0, 0, 0\}$

* Surface material also has 3 terms
  - ambient reflection coefficient $k_a = \{0, 0, 0, 0\}$ (R G B A)
  - diffuse : k_d
  - specular : k_s

  (discussed later)

* Same computation applied to each different point p on the surface.

  (l, n, v, r are different for different points. (will simplify later))

3 reflection terms

1. **Ambient Reflection**: intensity of ambient light L_a is the same at every point on the surface

   ambient light: $L_a = \{0, 0, 0, 0\}$
   material: $k_a = \{0, 0, 0, 0 \neq 1.0\} \in [0,1]$

   (ambient reflection coefficient): $k_a = \{k_{ar}, k_{ag}, k_{ab}\}$ (surface property)

   $k_{ac} \in [0,1] \quad \forall c = r, g, b$

   $I_{ac} = k_{ac} \cdot L_{ac}$ for each $c = r, g, b$ — i.e. for each color c, how much fraction of light intensity L_{ac} is reflected

   ② $L_a = \{0, 0, 0, 0\}$ (R G B A = 1.0) ④ $\Rightarrow I_a = k_a \cdot L_a$ (Take component-wise multiplication)

   ③ $I_a = k_a \cdot L_a$ — 4 components each

   * $\because k_{ac} \in [0,1] \ \forall c = r, g, b$. (k_{ar}, k_{ag}, k_{ab}) also makes a (color), extended with $(\alpha (= A) = 1.0)$

   ① $k_a = \{\overset{R}{0}, \overset{G}{0}, \overset{B}{0}, \overset{A}{0}\}$. we also call k_a the (ambient color) of the surface

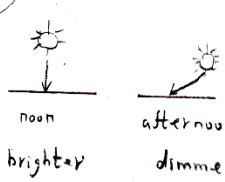
   * Note: In $I_a = k_a L_a$, L_a can be a (global ambient light) as well

2. **Diffuse Reflection**: rough surface, no prefered angle of reflection

   diffuse light: $L_d = \{0, 0, 0, 0\}$
   material: ~~$k_d \in [0,1]$~~
   $k_d = \{0, 0, 0, 0 \neq 1.0\} \in [0,1]$

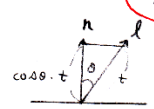
   Perfectly diffuse surface: Lambertian surface

   * Lambert's Law: we see only the vertical component of the incoming light

   brighter — dimmer

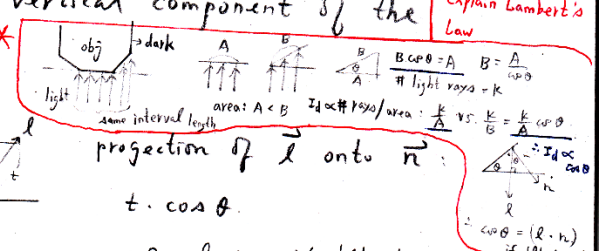
   $I_d \propto \cos\theta$.

   projection of $\vec{l}$ onto $\vec{n}$: $t \cdot \cos\theta$

   $\cos\theta = l \cdot n$ if $|l| = |n| = 1$

   * Better way to explain Lambert's Law

   $(\cos\theta) L_d = (l \cdot n) L_d$: vertical component of the light diffuse term L_d

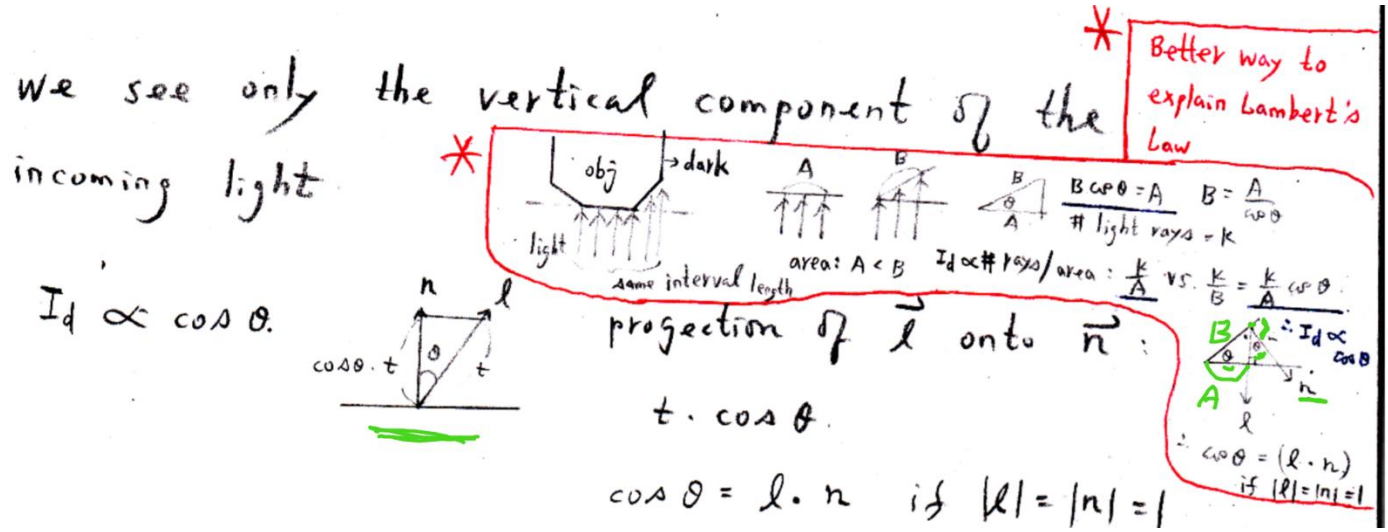
   $k_d = \{\overset{R}{0}, \overset{G}{0}, \overset{B}{0}, \overset{A}{0}\}$: (diffuse reflection coefficient) each component $\in [0,1]$ (Similar to k_a)

   → k_d is also called the (diffuse color) of the surface

   $\Rightarrow I_d = k_d (l \cdot n) L_d$

   ($k_d \cdot L_d$: component-wise multiplication)

Elaboration:



* If $l \cdot n < 0$ then no contribution. ie replace $(l \cdot n)$ with $\max\{(l \cdot n), 0\}$

* If a point source, need to account a distance attenuation

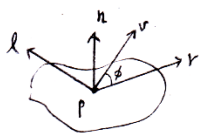
d: distance from the light source to the surface point p

$$\Rightarrow I_d = \frac{1}{a + bd + cd^2} K_d \cdot L_d \cdot \max\{(l \cdot n), 0\}$$

3. Specular Reflection: shiny, have highlight

(specular Light: $L_s = \{0,0,0,0\}$
material: ~~$K_s \in [0,1]$~~) $K_s = \{0,0,0,0\} \in [0,1]$

e.g. A red ball under white light will have a white highlight

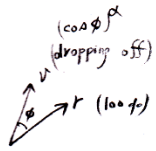


r: direction of perfect reflection.

v: " to the viewer

ϕ: angle between r & v

Approximation model: amount of specular reflection light seen by the viewer depends on ϕ:



$$I_s = K_s (\cos\phi)^\alpha L_s$$

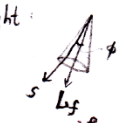
($K_s \cdot L_s$: component-wise multiplication)

(specular reflection coefficient) $\{0,0,0,0\}$ each component $\in [0,1]$

K_s is also called the (specular color) of surface

specular term of light

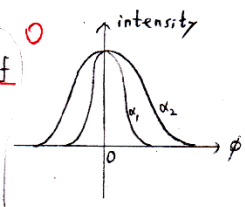
cf. spot light:



$I \propto (\cos\phi)^e$

α: shininess coefficient

(material property)



$\phi \neq 0 \Rightarrow \cos\phi < 1$

$\alpha \uparrow$ $(\cos\phi)^\alpha \downarrow$ $(\alpha > 1)$

$\alpha_1 > \alpha_2$.

$\Rightarrow$ Larger α, the reflected light is concentrated in a narrower region

$\alpha \to \infty$: mirror $\alpha \in [100, 500]$: metallic surface

$\alpha < 100$: broad highlights.

$\cos\phi = r \cdot v$ if $|r| = |v| = 1$, also no contribution if $r \cdot v < 0$ $\Rightarrow [\max(r \cdot v, 0)]^\alpha$

* If $l \cdot n < 0$ then no contribution.

$\Rightarrow$ Define $[\text{if } l \cdot n \geq 0] = \begin{cases} 1 & \text{if } l \cdot n \geq 0 \\ 0 & \text{else} \end{cases}$

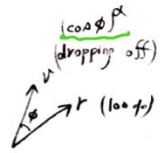
$$\Rightarrow I_s = [\text{if } l \cdot n \geq 0] \cdot K_s \cdot L_s [\max(r \cdot v, 0)]^\alpha$$

* Need to add distance attenuation if point source

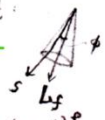
Elaboration:

approximation model : amount of specular reflection light seen by the viewer depends on ϕ :

$(\cos\phi)^{\alpha}$ (dropping off)



cf. spot light



$I \propto (\cos\phi)^{e}$

$$I_s = K_s (\cos\phi)^{\alpha} L_s$$

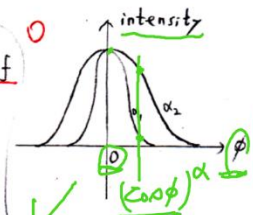
($K_s \cdot L_s$: component-wise multiplication)

(specular reflection coefficient) $\{0, 0, 0, 0\}$ each component $\in [0,1]$

K_s is also called the (specular color) of surface

specular term of light

α : shininess coefficient (material property)



$\phi \neq 0 \Rightarrow \cos\phi < 1$

$\alpha \uparrow$ $(\cos\phi)^{\alpha} \downarrow$ $(\alpha > 1)$

$\alpha_1 > \alpha_2$. $\cos\phi = \frac{1}{2}$

$(\cos\phi)^2 = \frac{1}{4}$

$(\cos\phi)^3 = 1/8$.

$\Rightarrow$ Larger α, the reflected light is concentrated in a narrower region

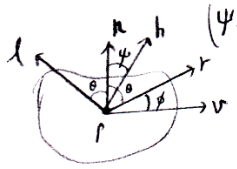
$\alpha \to \infty$: mirror $\alpha \in [100, 500]$: metallic surface

* r can be computed from l, n, but more complicated
* Avoid computing r $\Rightarrow$ Use half vector h

$h = \text{normalize}(l+v)$ $(|l| = |v| = 1)$

Replace $(r \cdot v)$ with $(n \cdot h)$

$(r \cdot v) = \cos\phi$, $(n \cdot h) = \cos\psi = \cos\frac{\phi}{2}$ $\left(\cos\phi \approx \cos\frac{\phi}{2}\right)$



$(\psi = \backslash psi)$

Take $h = \dfrac{l+v}{|l+v|} \Rightarrow \psi = \dfrac{\phi}{2}$

If v is on the same plane as l, n, r

[Pf: $\text{angle}(l,h) = \frac{1}{2}\text{angle}(l,v) = \frac{1}{2}(2\theta + \phi) = \theta + \frac{1}{2}\phi$

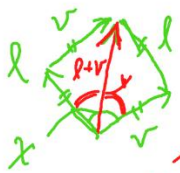
But $\text{angle}(l,h) = \theta + \psi$

$\therefore \psi = \frac{1}{2}\phi$]

[Goal: replace $(r \cdot v)$ with $(n \cdot h)$ $\Rightarrow$ avoid computing r.]

This is called (Blinn-Phong) or (Modified Phong) reflection model and is used typically.

Elaboration:



$x = \angle(l, v)$

$y = \angle(l, l+v) = \angle(v, l+v)$

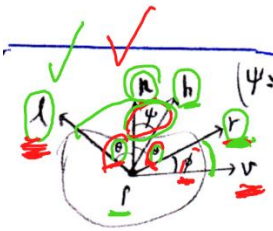
$\cancel{y = \frac{1}{2}x}$ $= \frac{1}{2}x$

* r can be computed from l, n, but more complicated
* Avoid computing r $\Rightarrow$ Use half vector h

$h = \text{normalize}(l+v)$ $(|l| = |v| = 1)$

Replace $(r \cdot v)$ with $(n \cdot h)$

$(r \cdot v) = \cos\phi$, $(n \cdot h) = \cos\psi = \cos\frac{\phi}{2}$ $\left(\cos\phi \approx \cos\frac{\phi}{2}\right)$



$(\psi = \backslash psi)$

Take $h = \dfrac{l+v}{|l+v|} \Rightarrow \psi = \dfrac{\phi}{2}$

If v is on the same plane as l, n, r

[Pf: $\text{angle}(l,h) = \frac{1}{2}\text{angle}(l,v) = \frac{1}{2}(2\theta + \phi) = \theta + \frac{1}{2}\phi$

But $\text{angle}(l,h) = \theta + \psi$

$\therefore \psi = \frac{1}{2}\phi$]

[Goal: replace $(r \cdot v)$ with $(n \cdot h)$ $\Rightarrow$ avoid computing r.]

This is called (Blinn-Phong) or (Modified Phong) reflection model and is used typically.

Note: component-wise multiplications:
$K_a \cdot L_a$, $K_d \cdot L_d$, $K_s \cdot L_s$
They are attenuated differently

Overall formula:

global ambient light

$$I = K_a \cdot L_{a_global} + \sum_{\text{light } i} (\text{Attenuation})_i \cdot \Big[K_a \cdot L_a + K_d \cdot L_d \cdot \max\{(\ell \cdot n), 0\} + [\text{if } \ell \cdot n \geq 0] \cdot K_s \cdot L_s \cdot \big(\max\{(r \cdot v), 0\}\big)^{\alpha} \Big]_i$$

replaced with $(n \cdot h)$

(1) If light i is a distant (directional) light, then

① $(\text{Attenuation})_i = 1$

② vector ℓ (from pt p to light source i = $-$(distant light direction L)) L $\ell = -L$

i.e. $\ell = -L$ (Also, use this ℓ to compute h = normalize$(\ell + v)$)

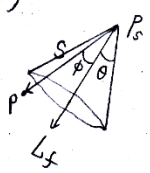
(2) If light i is a point source, then

① $(\text{Attenuation})_i = \dfrac{1}{a + bd + cd^2}$ where d = distance from pt p to the light source position.
a, b, c: constant, linear, quadratic attenuations

(3) If light i is a spotlight then

① $(\text{Attenuation})_i = \dfrac{1}{a + bd + cd^2} \cdot (\text{spotlight_attenuation})_i$ a, b, c, d are as in (2) point source.

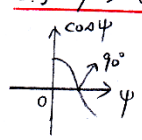
② $(\text{spotlight_attenuation})_i = ?$



(Let L_f = normalize(L_f))

θ: spotlight cut-off angle $\theta \in [0, 90°]$

(a) If $\phi > \theta$ then contribution $= 0$



In the range $[0, 90°]$

$\phi > \theta \iff \cos\phi < \cos\theta$

$\iff (L_f \cdot s) < \cos\theta$

$\iff L_f \cdot (-\ell) < \cos\theta$

In particular $d = |\overrightarrow{P_s P}|$

ℓ = normalize$(\overrightarrow{P P_s})$
$= -s$
$\therefore s = -\ell$

$\therefore$ If $L_f \cdot (-\ell) < \cos\theta$ then $(\text{spotlight_attenuation})_i = 0$

(b) Else,

$(\text{spotlight_attenuation})_i = (\cos\phi)^e$ e: spotlight exponent
$= [L_f \cdot (-\ell)]^e$

(Combining (a), (b): $(\text{spotlight_attenuation})_i = [\text{if } L_f \cdot (-\ell) \geq \cos\theta] \cdot [L_f \cdot (-\ell)]^e$)

Elaboration:

Overall formula:

Note: component-wise multiplications: $K_a \cdot L_a$, $K_d \cdot L_d$, $K_s \cdot L_s$. They are attenuated differently

$$I = \underbrace{K_a \cdot L_{a_global}}_{\text{global ambient light}} + \sum_{\text{light } i} (\text{Attenuation})_i \cdot \Big[K_a \cdot L_a + K_d \cdot L_d \cdot \max\{(\ell \cdot n), 0\} + [\text{if } \ell \cdot n \geq 0] \cdot K_s \cdot L_s \cdot \big(\max\{(r \cdot v), 0\}\big)^{\alpha} \Big]_i$$

replaced with $(n \cdot h)$

(1) If light i is a distant (directional) light, then

① $(\text{Attenuation})_i = 1$

② vector ℓ (from pt p to light source i) $= -$(distant light direction L)

L $\quad \ell = -L$

i.e. $\ell = -L$ (Also, use this ℓ to compute $h = \text{normalize}(\ell + v)$)

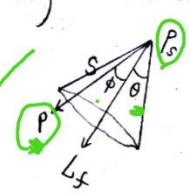
(2) If light i is a point source, then

① $(\text{Attenuation})_i = \dfrac{1}{a + bd + cd^2}$ where d = distance from pt p to the light source

(3) If light i is a spotlight then

① $(\text{Attenuation})_i = \dfrac{1}{a + bd + cd^2} \cdot (\text{spotlight_attenuation})_i$ $\quad$ a, b, c, d are as in (2) point source.

② $(\text{spotlight_attenuation})_i = ?$



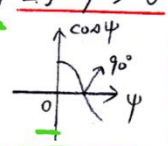
(Let $L_f = \text{normalize}(L_f)$)

θ: spotlight cut-off angle $\theta \in [0, 90°]$

In particular $d = |\overrightarrow{P_s P}|$

$\ell = \text{normalize}(\overrightarrow{P P_s})$
$= -s$
$\therefore s = -\ell$

(a) If $\phi > \theta$ then contribution $= 0$



In the range $[0, 90°]$

$\phi > \theta \iff \cos\phi < \cos\theta$

$\iff (L_f \cdot s) < \cos\theta$

$\iff L_f \cdot (-\ell) < \cos\theta$

If $L_f \cdot (-\ell) < \cos\theta$ then $(\text{spotlight_attenuation})_i = 0$

(b) Else,

$(\text{spotlight_attenuation})_i = (\cos\phi)^e$ $\quad$ e: spotlight exponent

$= [L_f \cdot (-\ell)]^e$

Polygonal shading : Phong reflection model : applied to every point on the surface

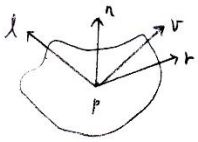
⇒ expensive

modeling curved surfaces in graphics : break curved surfaces into small (flat polygons)

⇒ simplify the shading calculation

* 3 ways of shading polygons :

1. Flat shading :



(r is defined from l, n)

l, n, v vary from point to point

① surface is a flat polygon ⇒ n is constant

② Assume a distant viewer ⇒ v "

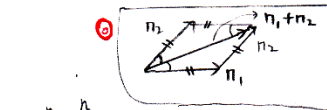
③ " distant light ⇒ l "

⇒ shading calculation is carried out once for each polygon

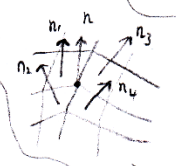
* <u>short coming</u>: small shading differences in adajent polygons will be seen with amplification by human eyes: overshooting the intensity on one side
(HW demo) under shooting " the other
of an edge

2. <u>Interpolative and Gouraud Shading</u>:

① average to get vertex normals:



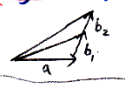
$\because |n_1| = |n_2| = 1$
$\therefore n_1+n_2$ evenly divides the angle between n_1 & n_2 $\Rightarrow n_1+n_2$ averaged the directions of n_1 & n_2.



$$\vec{n} = \frac{n_1+n_2+n_3+n_4}{|n_1+n_2+n_3+n_4|}$$

average over the normals of the polygons sharing the vertex

Note: Each n_i must be (unit length) so that each is (weighted equally) in the sum

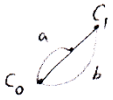
eg. 

$a+b_1 \neq a+b_2$

② compute shade at vertices. (using Phong reflection model)

③ interpolate using <u>vertex shade</u> to get the shade for each point

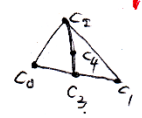
* <u>bilinear interpolation</u>:



$\alpha = \frac{a}{b}$

$C_{01}(\alpha) = C_0 + \alpha(C_1 - C_0)$
$= (1-\alpha)C_0 + \alpha C_1$

$C_{23}(\alpha) = (1-\alpha)C_2 + \alpha C_3$



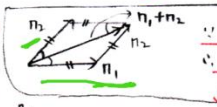
* associate a normal with each vertex: normal:
- averaged normal: step ①
- or
- true normal: if can get from analysis eg. sphere

need data structure for adjacent info.

Elaboration:

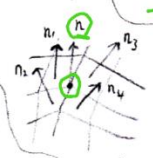
2. Interpolative and Gouraud Shading :

① average to get vertex normals :



$\because |n_1| = |n_2| = 1$

$\therefore n_1+n_2$ evenly divides the angle between n_1 & n_2 $\Rightarrow n_1+n_2$ averaged the directions of n_1 & n_2.

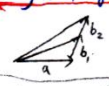


$$\vec{n} = \frac{n_1+n_2+n_3+n_4}{|n_1+n_2+n_3+n_4|}$$

average over the normals of the polygons sharing the vertex

Note : Each n_i must be unit length so that each is weighted equally in the sum

eg.

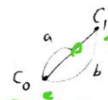


$a+b_1 \neq a+b_2$

② compute shade at vertices. (using Phong reflection model)

③ interpolate using vertex shade to get the shade for each point.

* bilinear interpolation :



$\alpha = \frac{a}{b}$

$C_{01}(\alpha) = C_0 + \alpha(C_1 - C_0)$

$= (1-\alpha)C_0 + \alpha C_1$

$\alpha = 0$

$\alpha = 1$

$C_{23}(\alpha) = (1-\alpha)C_2 + \alpha C_3$

* associate a normal with each vertex : normal :

need data structure for adjacent info.

averaged normal : step ①

or

true normal : if can get from analysis eg. sphere

$\vec{ov}$ is normal vector at pt v. on sphere.

3. Phong shading :

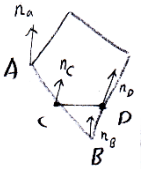
⓪ average to get vertex normals (as in step ① of Gourand shading)

① interpolate to get normals at all points

Note: flat polygon.
⊙ originally this polygon has its normal $\vec{n}$
Step ① may give each point in the interior a normal different from $\vec{n}$
Fine. the purpose is to smooth out the normal changes

Each vertex has a different normal by ⓪
⇒ want to have a smooth transition for normals

* use bi linear interpolation



$$n_{AB}(\alpha) = (1-\alpha)\, n_A + \alpha\, n_B$$

$$n_{CD}(\beta) = (1-\beta)\, n_C + \beta\, n_D$$

⊙ Note: by property of linear interpolation.
min, max only occurs at endpoints
∴ max/min shading only occurs at vertices in Gourand shading
⇒ Gourand : has no highlight in interior of a polygon.
But Phong shading can.

② compute the shade at each point using its normal

(independent computation at each point)

* Smoother than Gourand shading

* Now is supported by programmable GPU Done in fragment shader

* Normal Matrix

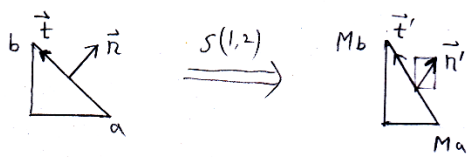
* Typically we perform shading computation in the (eye frame) (i.e. the right-handed eye frame where the eye/camera is at the origin looking at the $-z$ direction. This is the frame obtained by applying LookAt( ) to the world frame)

Let $\vec{t}$ be the tangent at pt p being shaded.
$\vec{n}$ " normal "

$\vec{n}, \vec{t}$ are in the model frame
M the model-view matrix
i.e. Mp puts p in the eye frame

(1) Suppose M involves (non-uniform scaling) (i.e. scaling factors in x-, y-, z-dimensions are different)

e.g. $S(1,2)$ in 2D



$\vec{t'} = Mb - Ma = M(b-a)$
$= M\vec{t}$ is the tangent after transformation

i.e. We can still apply M to $\vec{t}$ to obtain the new tangent $\vec{t'}$ correctly.
But applying M to $\vec{n}$ does NOT give the correct normal vector. (since $\vec{n'}$ is NOT perpendicular to $\vec{t'}$)

(2) Deriving the correct matrix for normal vector: the (normal matrix)

Let $\vec{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix}$ $\quad \vec{n} \cdot \vec{t} = 0$

The dot product can be expressed as matrix multiplication:

$\vec{n} \cdot \vec{t} = [\quad\quad][\quad] = (\vec{n})^t \vec{t}$ —— (*) $\quad$ ($(\vec{n})^t$: transpose of $\vec{n} = [\quad]$)
$= 0$

From (*) we have $0 = (\vec{n})^t \vec{t} = \underbrace{\left((\vec{n})^t M^{-1}\right)\left(M\vec{t}\right)}_{=I}$ (☆)

But $M = \begin{bmatrix} l & T \\ 0 & 1 \end{bmatrix}$ and the 4-th component of $\vec{t}$ is 0 $\Rightarrow$ We can ignore the 4th column of M. i.e. $\begin{bmatrix} T \\ 1 \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \\ 1 \end{bmatrix}$

$\Rightarrow$ Then the 4th row of the remaining columns are 0 and can be ignored

$\therefore$ In (☆) we can use l to replace M:

$$\underbrace{\left((\vec{n})^t l^{-1}\right)}_{=(\vec{x})^t}\underbrace{\left(l\vec{t}\right)}_{=\text{transformed tangent } \vec{t'}} = 0$$

$(\vec{x})^t$ where $\vec{x}$ is the transformed normal, in the form of (*):

cf: In (*): $(\vec{n})^t \vec{t} = 0$
Here: $(\vec{x})^t \vec{t'} = 0$

$\therefore (\vec{x})^t = (\vec{n})^t l^{-1} \Rightarrow \vec{x} = \left[(\vec{n})^t l^{-1}\right]^t = (l^{-1})^t (\vec{n})$

$\Rightarrow$ Desired normal $\vec{x}$ is obtained by $N \overset{\vec{n}}{\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}}$ where the 3×3 matrix N (normal matrix) is $\left((l^{-1})^t\right)$

Simplification:

(3) If M only involves translations, rotations, uniform scaling, and LookAt( )

(scaling factors in x-, y-, z-dim are all the same.)

then: translations have no effect on ℓ

LookAt( ) has translation and rotation

$\Rightarrow$ ℓ only involves rotations and uniform scaling

But uniform scaling has no effect after we normalize the transformed normal vector

$\Rightarrow$ $\ell \equiv R$. But $R^{-1} = R^t$

$\therefore$ $(\ell^{-1})^t \equiv (R^{-1})^t = (R^t)^t = R \equiv \ell$

ie ① We can use (ℓ) to replace $(\ell^{-1})^t$

(ie ② We can use the model-view matrix M (4x4) to apply to normal $\vec{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix}$ 4 components)

① $\equiv$ ②