トーンマップ関数

Kohei Ishiyama

September 20, 2017

1 トーンマップ関数

次のトーンマップ関数は Toe $f_t(x)$, Mid $f_m(x)$, Shoulder $f_s(x)$ の各部分を持ち、$(x_1, y_1), (x_2, y_2), (x_3, y_3)$ の 3 点を指定することで決まる, 滑らかな関数である (図 1).

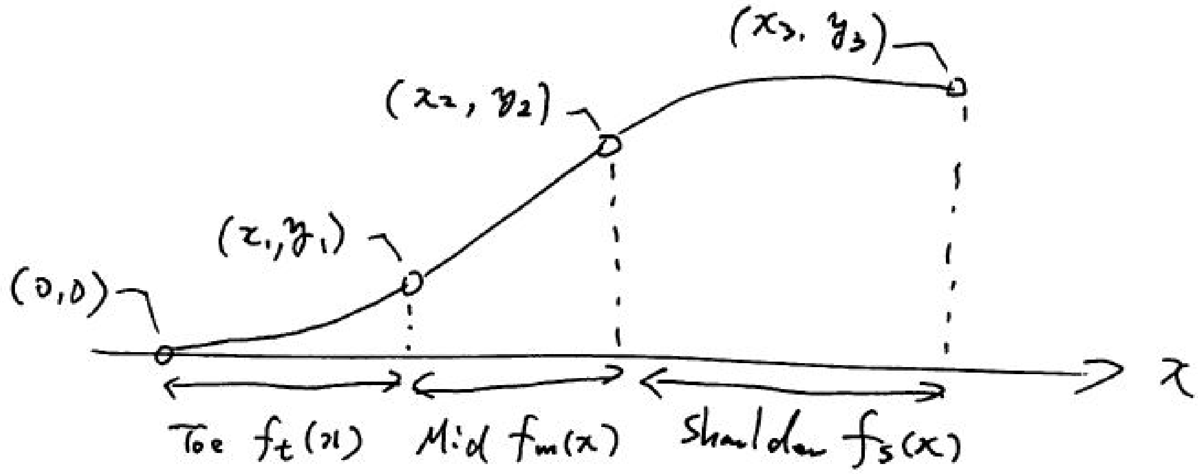

Figure 1: トーンマップ関数

$$f(x) = \begin{cases} f_t(x) = -\frac{a_t}{x+b_t} + c_t & 0 \le x < x_1, \\ f_m(x) = s_m x + b_m & x_1 \le x < x_2, \\ f_s(x) = -\frac{a_t}{x+b_s} + c_s & x_2 \le x < x_3. \end{cases}$$

$$a_t = \frac{s_m x_1^2 y_1^2}{(y_1 - s_m x_1)^2},\ b_t = \frac{s_m x_1^2}{y_1 - s_m x_1},\ c_t = \frac{y_1^2}{y_1 - s_m x_1}$$

$$s_m = \frac{y_2 - y_1}{x_2 - x_1}, \quad b_m = y_1 - s_m x_1 (= y_2 - s_m x_2)$$

$$a_s = \frac{s_m (x_2 - x_3)^2 (y_2 - y_3)^2}{s_m (x_2 - x_3) - y_2 + y_3},\ b_s = \frac{s_m x_2 (x_3 - x_2) + x_3 (y_2 - y_3)}{s_m (x_2 - x_3) - y_2 + y_3},\ c_s = \frac{y_3 (s_m (x_2 - x_3) + y_2) - y_2^2}{s_m (x_2 - x_3) - y_2 + y_3}$$

この関数の導出を示す.

2　導出

次のような Toe f_t, Mid f_m, Shoulder f_s 関数を考える.

$$f_t(x) = -\frac{a_t}{x+b_t} + c_t \tag{1}$$
$$f_m(x) = a_m x + b_m \tag{2}$$
$$f_s(x) = -\frac{a_s}{x+b_s} + c_s \tag{3}$$

これらの未知定数 $a_t, b_t, c_t, a_m, b_m, a_s, b_s, c_s$ を, 位置 $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ から求めよう。まず後のために, 各関数の一次微分 (傾き) を求めておく。

$$\frac{df_t(x)}{dx} = \frac{a_t}{(x+b_t)^2} \tag{4}$$
$$\frac{df_m(x)}{dx} = a_m \tag{5}$$
$$\frac{df_s(x)}{dx} = \frac{a_s}{(x+b_s)^2} \tag{6}$$

次に, それぞれの関数の端点に課される条件について考える。

　トーンマップ関数は $(0,0)$ から出発し, (x_1, y_1) で $f_t(x)$ が (x_2, y_2) に繋がり, (x_2, y_2) で $f_s(x)$ に繋がったのち, (x_3, y_3) に至る。(x_1, y_1) と (x_2, y_2) の間の傾きを s_m とすれば, 各関数の境界条件は次のようになる。

$$f_t(x):\ f_t(0) = 0,\ f_t(x_1) = y_1,\ \left.\frac{df_t(x)}{dx}\right|_{x=x_1} = s_m \tag{7}$$
$$f_m(x):\ f_t(x_1) = y_1,\ f_t(x_2) = y_2,\ \left.\frac{df_m(x)}{dx}\right|_{x=x_1} = \left.\frac{df_m(x)}{dx}\right|_{x=x_2} = s_m \tag{8}$$
$$f_s(x):\ f_s(x_2) = y_2,\ f_s(x_3) = y_3,\ \left.\frac{df_s(x)}{dx}\right|_{x=x_2} = s_m \tag{9}$$

これらの条件を使えば, 各関数の未知定数を求めることが出来る。

2.1　Toe 関数

Toe 関数 (1) とその一次微分 (4) に条件 (7) を課すと,

$$f_t(0) : -\frac{a_t}{b_t} + c_t = 0$$
$$f_t(x_1) : -\frac{a_t}{x_1+b_t} + c_t = y_1$$
$$\left.\frac{df_t(x)}{dx}\right|_{x=0} : \frac{a_t}{(x_1+b_t)^2} = s_m$$

となる。これらを a_t, b_t, c_t について解けば,

$$a_t = \frac{s_m x_1^2 y_1^2}{(y_1 - s_m x_1)^2},\ b_t = \frac{s_m x_1^2}{y_1 - s_m x_1},\ c_t = \frac{y_1^2}{y_1 - s_m x_1} \tag{10}$$

と, Toe 関数の未知定数が求まる.

2.2 Mid 関数

Mid 関数 (2) とその一次微分 (5) に条件 (8) を課すと,

$$
\begin{aligned}
f_m(x_1) &: a_m x_1 + b_m = y_1 \\
f_m(x_2) &: a_m x_2 + b_m = y_2 \\
\frac{df_m(x)}{dx} &: a_m = s_m
\end{aligned}
$$

これらを a_m, b_m について解けば,

$$
a_m = s_m = \frac{y_2 - y_1}{x_2 - x_1}, \quad b_m = y_1 - s_m x_1 = y_2 - s_m x_2 \tag{11}
$$

と, Mid 関数の未知定数が求まる.

2.3 Shoulder 関数

Shoulder 関数 (3) とその一次微分 (6) に条件 (9) を課すと,

$$
\begin{aligned}
f_s(x_2) &: -\frac{a_s}{x_2 + b_s} + c_s = y_2 \\
f_s(x_3) &: -\frac{a_s}{x_3 + b_s} + c_s = y_3 \\
\left.\frac{df_s(x)}{dx}\right|_{x=x_2} &: \frac{a_s}{(x_2 + b_s)^2} = s_m
\end{aligned}
$$

となる。これらを a_s, b_s, c_s について解けば,

$$
a_s = \frac{s_m(x_2 - x_3)^2(y_2 - y_3)^2}{s_m(x_2 - x_3) - y_2 + y_3}, \; b_s = \frac{s_m x_2(x_3 - x_2) + x_3(y_2 - y_3)}{s_m(x_2 - x_3) - y_2 + y_3}, \; c_s = \frac{y_3(s_m(x_2 - x_3) + y_2) - y_2^2}{s_m(x_2 - x_3) - y_2 + y_3} \tag{12}
$$

と Shoulder 関数の未知定数が求まる。

2.4 トーンマップ関数

以上で, 位置 $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ と傾き s_m から, $f_t(x)$ (1) の未知定数 (10), $f_m(x)$ (2) の未知定数 (11), そして $f_s(x)$ (3) の未知定数 (12) が求まった。これらを組み合わせると, 求めるトーンマップ関数を得ることが出来る.

$$
f(x) = \begin{cases} f_t(x) & 0 \le x < x_1, \\ f_m(x) & x_1 \le x < x_2, \\ f_s(x) & x_2 \le x < x_3. \end{cases} \tag{13}
$$

2.5 逆トーンマップ関数

f_t (1), f_m (2), f_s (3) の逆関数は, 次のように得られる.

$$
x = f_t^{-1}(y) = -\frac{a_t}{y - c_t} - b_t \tag{14}
$$

$$
x = f_m^{-1}(y) = \frac{y - b_m}{a_m} \tag{15}
$$

$$
x = f_s^{-1}(y) = -\frac{a_s}{y - c_s} - b_s \tag{16}
$$

これらを使うと, 関数 (13) の逆関数が得られる.

$$
f^{-1}(y) = \begin{cases} f_t^{-1}(y) & 0 \le f_t^{-1}(y) < x_1, \\ f_m^{-1}(y) & x_1 \le f_m^{-1}(y) < x_2, \\ f_s^{-1}(x) & x_2 \le f_s^{-1}(x) < x_3. \end{cases} \tag{17}
$$

2.6 関数の定義域

双曲線関数は $f = -\frac{a}{x+b} + c$ は $x = b$ に特異点を持つ. この特異点の回避措置には条件分岐が必要なため, 処理負荷がかかる. ここに, トーンマップ関数の定義域に特異点が存在する条件をまとめておく.

関数	特異点が存在する領域
Toe	$0 \leq -b_t < x_1$
逆 Toe	$0 \leq c_t < y_1$
Shoulder	$x_2 \leq -b_s < x_3$
逆 Shoulder	$y_2 \leq c_t < y_3$

2.7 サンプルコード

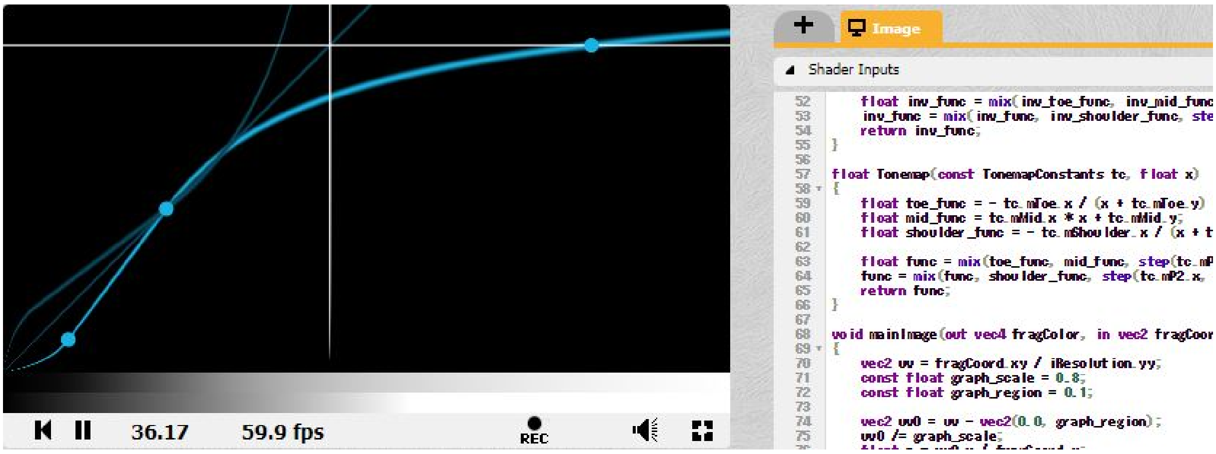

Figure 2: shadertoy

下記サンプルコードは, shadertoy(https://www.shadertoy.com/new) にコピペすると実行できる。

```
struct TonemapConstants
{
	vec3 mToe;
	vec2 mMid;
	vec3 mShoulder;
	vec2 mP1;
	vec2 mP2;
};

bool CalculateTonemapConstants(vec2 p1, vec2 p2, vec2 p3, out TonemapConstants tc)
{
	// slope
	float denom = p2.x - p1.x;
	denom = abs(denom) > 1e-5 ? denom : 1e-5;
	float sl = (p2.y - p1.y) / denom;

	// tonemap constants
	float td = p1.y - sl * p1.x;
	td = abs(td) > 1e-5 ? td : 1e-5;
	tc.mToe.x = sl * p1.x * p1.x * p1.y * p1.y / (td * td);
	tc.mToe.y = sl * p1.x * p1.x / td;
	tc.mToe.z = p1.y * p1.y / td;

	tc.mMid.x = sl;
	tc.mMid.y = p1.y - sl * p1.x;

	float sd = sl * (p2.x - p3.x) - p2.y + p3.y;
	sd = abs(sd) > 1e-5 ? sd : 1e-5;
	tc.mShoulder.x = sl * pow(p2.x - p3.x, 2.0) * pow(p2.y - p3.y, 2.0) / (sd * sd);
	tc.mShoulder.y = (sl * p2.x * (p3.x - p2.x) + p3.x * (p2.y - p3.y) ) / sd;
	tc.mShoulder.z = (-p2.y * p2.y + p3.y * (sl * (p2.x - p3.x) + p2.y) ) / sd;

	tc.mP1 = p1;
	tc.mP2 = p2;

	// Check the domain where contains any undefined regions
	bool isDomain = true;
	if (0.0 <= -tc.mToe.y && -tc.mToe.y < p1.x) { isDomain = false; }
	if (0.0 <= tc.mToe.z && tc.mToe.z < p1.y) { isDomain = false; }
	if (p2.x <= -tc.mShoulder.y && -tc.mShoulder.y < p3.x) { isDomain = false; }
	if (p2.y <= tc.mShoulder.z && tc.mShoulder.z < p3.y) { isDomain = false; }

	return isDomain;
}
```

```
float InvTonemap(const TonemapConstants tc, float y)
{
	float inv_toe_func = - tc.mToe.x / (y - tc.mToe.z) - tc.mToe.y;
	float inv_mid_func = (y - tc.mMid.y) / tc.mMid.x;
	float inv_shoulder_func = - tc.mShoulder.x / (y - tc.mShoulder.z) - tc.mShoulder.y;

	float inv_func = mix(inv_toe_func, inv_mid_func, step(tc.mP1.y, y));
	inv_func = mix(inv_func, inv_shoulder_func, step(tc.mP2.y, y));
	return inv_func;
}

float Tonemap(const TonemapConstants tc, float x)
{
	float toe_func = - tc.mToe.x / (x + tc.mToe.y) + tc.mToe.z;
	float mid_func = tc.mMid.x * x + tc.mMid.y;
	float shoulder_func = - tc.mShoulder.x / (x + tc.mShoulder.y) + tc.mShoulder.z;

	float func = mix(toe_func, mid_func, step(tc.mP1.x, x));
	func = mix(func, shoulder_func, step(tc.mP2.x, x));
	return func;
}

void mainImage(out vec4 fragColor, in vec2 fragCoord)
{
	vec2 uv = fragCoord.xy / iResolution.yy;
	const float graph_scale = 0.8;
	const float graph_region = 0.1;

	vec2 uv0 = uv - vec2(0.0, graph_region);
	uv0 /= graph_scale;
	float p = uv0.y / fragCoord.y;

	vec2 p1 = vec2(0.2, 0.1);
	vec2 p2 = vec2(0.5, 0.5);
	vec2 p3 = vec2(1.8, 1.0);
	vec4 m = iMouse / (iResolution.yyyy * graph_scale);
	m.yw -= graph_region;
	if (m.z > 0.0)
	{
		if (length(m.zw - p1) < 0.05) { p1 = m.xy; }
		if (length(m.zw - p2) < 0.05) { p2 = m.xy; }
		if (length(m.zw - p3) < 0.05) { p3 = m.xy; }
	}

	TonemapConstants tc;
	bool isDomain = CalculateTonemapConstants(p1, p2, p3, tc);
	vec3 lineColor = isDomain ? vec3(0.1, 0.7, 0.9) : vec3(0.8, 0.1, 0.1);

	float y = Tonemap(tc, uv0.x);
	fragColor.rgb = mix(fragColor.rgb, lineColor, 1.0 - smoothstep(0.0, 5.0 * p, abs(uv0.y - y)));

	y = InvTonemap(tc, uv0.x);
	fragColor.rgb = mix(fragColor.rgb, lineColor * 0.4, 1.0 - smoothstep(0.0, 5.0 * p, abs(uv0.y - y)));

	y = InvTonemap(tc, Tonemap(tc, uv0.x));
	fragColor.rgb = mix(fragColor.rgb, lineColor * 0.4, 1.0 - smoothstep(0.0, 3.0 * p, abs(uv0.y - y)));

	// horizontal line
	fragColor.rgb = mix(fragColor.rgb, vec3(1.0, 1.0, 1.0), 1.0 - smoothstep(0.0, 1.5 * p, abs(uv0.y - 1.0)));

    // vertical line
	fragColor.rgb = mix(fragColor.rgb, vec3(1.0, 1.0, 1.0), 1.0 - smoothstep(0.0, 1.5 * p, abs(uv0.x - 1.0)));

	fragColor.rgb = mix( fragColor.rgb, lineColor, 1.0-smoothstep(0.02,0.025, length(uv0-p1)) );
	fragColor.rgb = mix( fragColor.rgb, lineColor, 1.0-smoothstep(0.02,0.025, length(uv0-p2)) );
	fragColor.rgb = mix( fragColor.rgb, lineColor, 1.0-smoothstep(0.02,0.025, length(uv0-p3)) );

	fragColor.rgb = uv.y < 0.1 ? vec3(Tonemap(tc, uv.x)) : fragColor.rgb;
	fragColor.rgb = uv.y < 0.05 ? vec3(uv.x) : fragColor.rgb;
}
```

3 応用

この Toe, Mid, Shoulder 関数の導出過程は, 未知定数が 3 つで一次微分可能であれば, 任意の関数に応用することができる。