The Visual Computer manuscript No.
(will be inserted by the editor)

Supplemental Document for adaptive sparse polynomial regression for camera lens simulation

Quan Zheng · Changwen Zheng

Abstract Our paper has presented an adaptive method for building a sparse and accurate multivariate polynomial system, to approximate the mapping of rays through a complex camera. In Section 7.1 of the main paper, we depict how our sparse polynomial systems can be integrated in a Monte Carlo rendering framework. In this supplemental document, we show the derivation of a new Monte Carlo estimator. In addition, a sample of a fitted polynomial system is also provided.

1 Derivation of Monte Carlo estimator

For path space rendering, we set the second vertex of a camera subpath in the scene. For bidirectional path tracing, the connection between camera subpath and light subpath is only implemented in the scene. In addition, light tracing technique is not used to construct a transport path. The reason is that light tracing involves constructing a "SDS" path, and the probability of sampling such paths is 0.

In Monte Carlo ray tracing, the radiance of a pixel j can be described as [1]:

$$I_j = \int_{\Omega} f(X) d\mu(X), \tag{1}$$

where X is a transport path, Ω is the space of paths of all lengths, $d\mu(X)$ is the differential product area measure, and f denotes a measurement contribution function:

$$f = L(s_1, -\omega_o)\bar{G}(s_1 \leftrightarrow s_0)W_e(s_0). \tag{2}$$

(✉)Quan Zheng[1,2] · Changwen Zheng[1]
quan.zheng@outlook.com; cwzheng@ieee.org
[1] Science and Technology on Integrated Information System Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China

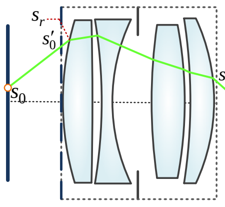
Here L is the incident radiance to s_1, $\bar{G}(s_1 \leftrightarrow s_0)$ is an extended factor and W_e denotes the visual importance. We define a Monte Carlo estimator as

$$\frac{1}{N}\sum \frac{L(s_1, -\omega_o)\bar{G}(s_1 \leftrightarrow s_0)W_e(s_0)}{p_A(s_0)p_A(s_1)}. \tag{3}$$

$L(s_1, -\omega_o)$ subsumes the incoming radiance carried by the light path (See Fig. 1 for the notations of vertices). It can be represented by

$$L(s_1, -\omega_o) = L_e(l_0) \cdot T(s_2 \leftrightarrow l_0) \cdot G(s_1 \leftrightarrow s_2). \tag{4}$$

Here $L_e(l_0) = L_e(l_0 \to l_1)$ is the emitted radiance from the light source. $T(\cdot)$ is called the path throughput. $G(s_1 \leftrightarrow s_2) = V(s_1 \leftrightarrow s_2)\frac{\cos\theta_{1,2}\cos\theta_{2,1}}{\|s_1 - s_2\|^2}$ is called a geometry factor.



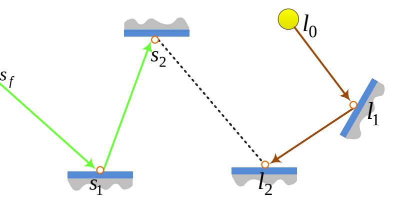


Fig. 1: Re-post of the Fig. 5 of the main paper for fast look-up. A camera subpath and a light subpath in the context of complex camera lenses. s_0 is on the sensor and s_0' is on a line pupil. s_r and s_f are on the rear lens and front lens respectively.

The remaining unknown factors are $\bar{G}$ and $p_A(s_1)$. In our method, we decide the initial camera ray via line pupils. The position of s_1 is indirectly affected by the s_0'. We thus transform sampling densities between the two points with

$$\begin{aligned} p_A(s_1) &= p_A(s_0') \cdot \left| \frac{\partial s_{1|W}}{\partial s_0'} \right| \\ &= p_A(s_0') \cdot \left| \frac{\partial s_{1|W}}{\partial s_{f|W}} \cdot \frac{\partial s_{f|W}}{\partial s_0'} \right| . \\ &= p_A(s_0') \cdot \frac{\cos \theta_{s_f}^2}{\cos \theta_{s_1}^2} \cdot \left| \frac{\partial s_{f|W}}{\partial s_0'} \right| \end{aligned} \tag{5}$$

The notation $s_{\cdot|W}$ represents a vertex in the world coordinate frame. We firstly transform $s_{\cdot|W}$ into the coordinate $s_{\cdot|C}$ in the camera coordinate frame, and then transform $s_{\cdot|C}$ into $s_{\cdot|S}$ in the local sphere frame of the front lens element. We use $\mathbf{A}$ to represent the submatrix $\frac{d\bar{x}d\bar{y}}{dxdy}$ of the Jacobian matrix.

$$\left| \frac{\partial s_{f|W}}{\partial s_0'} \right| = |J_{W2C}| \cdot |J_{C2S}| \cdot |J_H| \cdot \left| \frac{\partial s_{f|S}}{\partial s_0'} \right| . \tag{6}$$

$$\begin{aligned} \left| \frac{\partial s_{f|S}}{\partial s_0'} \right| &= \left| \frac{\partial s_{f|S}}{\partial s_0} \cdot \frac{\partial s_0}{\partial s_0'} \right| \\ &= \left| \frac{d\bar{x}d\bar{y}}{dxdy} \right| \cdot \frac{\cos \theta_{s_0}^2}{\cos \theta_{s_f}^2} \cdot \frac{\cos \theta_{s_0'}^2}{\cos \theta_{s_0}^2} . \\ &= \frac{|\mathbf{A}|}{\cos \theta_{s_0}^2} \cdot \frac{\cos \theta_{s_0'}^2}{\cos \theta_{s_f}^2} \end{aligned} \tag{7}$$

Here, $|J_{M2N}|$ stands for the Jacobian of the transformation from coordinate frame M to coordinate frame N. $|J_H|$ denotes the Jacobian of the transformation from the sphere frame of front lens to the outer hemisphere frame (see Fig. 1 of the main paper). For brevity, we denote $|J_{W2C}| \cdot |J_{C2S}| \cdot |J_H|$ as $|J_{all}|$. At this point, $p_A(s_1)$ can be simplified as:

$$p_A(s_1) = p_A(s_0') \cdot |J_{all}| \cdot \frac{|\mathbf{A}|}{\cos \theta_{s_0}^2} \cdot \frac{\cos \theta_{s_0'}^2}{\cos \theta_{s_1}^2} . \tag{8}$$

We define an extended factor $\bar{G}(s_1 \leftrightarrow s_0)$ as

$$\bar{G}(s_1 \leftrightarrow s_0) = G(s_1 \leftrightarrow s_0) \cdot P_a(s_1 \leftrightarrow s_0) \cdot T_r . \tag{9}$$

Here, P_a denotes a passage function, it is 1 if the ray can pass the lens system and 0 otherwise. T_r is the fresnel transmittance. s_1 and s_0 are not mutually visible, because there is an array of lens elements between s_1 and s_0. $G(s_1 \leftrightarrow s_0)$ is a generalized geometric term [2]:

$$\begin{aligned} G(s_1 \leftrightarrow s_0) &= \frac{\partial \omega_{s_1}^{\perp}}{\partial s_0} \\ &= \frac{\partial s_{f|W} \cos \theta_{s_f} / \|s_{f|W} - s_1\|^2}{\partial s_0} . \end{aligned} \tag{10}$$

$$\begin{aligned} \frac{\partial s_{f|W}}{\partial s_0} &= |J_{W2C}| \cdot |J_{C2S}| \cdot |J_H| \cdot \frac{\partial s_{f|S}}{dxdy / \cos \theta_{s_0}} \\ &= |J_{all}| \cdot \frac{|\mathbf{A}|}{\cos \theta_{s_0}} \end{aligned} . \tag{11}$$

Thus $\bar{G}(s_1 \leftrightarrow s_0)$ can be expressed as:

$$\bar{G}(s_1 \leftrightarrow s_0) = P_a \cdot T_r \cdot |J_{all}| \cdot \frac{|\mathbf{A}|}{\cos \theta_{s_0}} \cdot \frac{\cos \theta_{s_f}}{\|s_{f|W} - s_1\|^2} . \tag{12}$$

Finally, we substitute Eq. 8 and Eq. 12 into the Eq. 3 to obtain the estimator. The contents in the parentheses of L, G, W_e, T_r are omitted for brevity.

$$\begin{aligned} &\frac{1}{N} \sum \frac{L \cdot W_e \cdot P_a \cdot T_r}{p_A(s_0) p_A(s_0')} \cdot \frac{\frac{|J_{all}| \cdot |\mathbf{A}| \cdot \cos \theta(s_f)}{\left(\cos \theta(s_0) \cdot \|s_{f|W} - s_1\|^2\right)}}{\frac{|J_{all}| \cdot |\mathbf{A}| \cdot \cos \theta^2(s_0')}{\cos \theta^2(s_0) \cos \theta^2(s_1)}} = \\ &\frac{1}{N} \sum \frac{L \cdot W_e \cdot P_a \cdot T_r}{p_A(s_0) p_A(s_0')} \cdot \frac{\cos \theta_{s_0} \cos \theta_{s_1}^2 \cos \theta_{s_f}}{\cos \theta_{s_0'}^2 \cdot \|s_{f|W} - s_1\|^2} \end{aligned} . \tag{13}$$

2 Example of a sparse polynomial system

In this section, we show a fitted sparse polynomial system of a fisheye-ii lens. The polynomial system consists of 5 polynomials corresponding to 5 output variable: *out_x*, *out_y*, *out_u*, *out_v*, and *out_eta*. The input variables are x, y, u, v, and *lambda*. The training data set contains 4000 samples. The number of terms is limited to 40. The maximum degrees of the fitted polynomials are 5, 6, 5, 6, and 11.

References

1. Veach, E.: Robust monte carlo methods for light transport simulation. Ph.D. thesis, Stanford University (1997)
2. Jakob, W., Marschner, S.: Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. ACM Trans. Graph. **31**(4), 58:1–58:13 (2012)

```
    float out_x =  + 0.000107506  + 8.37381 *u + -1.16482 *x + -1.40838
          *x*lambda + -3.26043 *u*lambda + 0.000260936 *x*pow(y, 2) +
          3.48264 *x*pow(u, 2) + 59.8287 *u*pow(v, 2) + 2.03489
          *x*pow(lambda, 2) + 29.5743 *pow(u, 3) + 1.86916 *x*pow(v, 2)
          + 0.116914 *pow(x, 2)*u + 0.0993456 *x*y*v + 0.054114 *pow(y,
          2)*u + 3.46224 *y*u*v + 4.15285 *u*pow(lambda, 2) + -0.135439
          *pow(x, 2)*u*lambda + -0.00131725 *pow(x, 3)*lambda + -0.960116
          *x*pow(lambda, 3) + -3.82303 *y*u*v*lambda + -2.01202 *x*pow(v,
          2)*lambda + -0.00126578 *x*pow(y,2)* lambda + -0.0405873 *pow(y,
          2)*u*lambda + -84.5489 * u*pow(v, 2)*lambda + -0.0893615
          *x*y*v*lambda + -6.04454 *x*pow(u, 2)*lambda + -87.9182 *pow(u,
          3)*lambda + 3.37479e-005 *pow(x, 3)*y*v + 0.0439808 *pow(x,3)
          *pow(u, 2) + -0.00316706 *pow(y, 3)*u*v + 0.000526135 *pow(x,
          4)*u + 1.56569e-006 *pow(x, 5) + -9.71158e-007 *pow(x, 3)*pow(y,
          2) + 545.874 *pow(u, 5) + 53.9899 *x*pow(u, 4) + 9.21141e-005
          *pow(x, 2)*pow(y, 2)*u + 0.0363226 *pow(x, 2)*u*pow(v, 2) +
          -9.35092e-007 *x*pow(y, 4) + 0.0715244 *pow(y, 2)*pow(u, 3) +
          2.20053 *pow(x, 2)*pow(u, 3);
```

```
   float out_y = + 0.000184841  + 8.46678 *v + -1.16833 *y + -1.36683
         *y*lambda  +  -3.5678  *v*lambda  +  3.15864  *x*u*v  +  1.9379
         *y*pow(lambda, 2) + 54.41 *pow(u, 2)*v + 0.0505611 *pow(x, 2)*v
         +  29.9595  *pow(v,  3)  +  3.4686  *y*pow(v,  2)  +  4.33691
         *v*pow(lambda,2) + 1.66522 *y*pow(u, 2) + 0.0887244 *x*y*u +
         0.117847 *pow(y, 2)*v + -0.134595 *pow(y, 2)*v*lambda + -3.3116
         *x*u*v*lambda  +  -1.75644  *y*pow(u,  2)*lambda  +  -0.0755894
         *x*y*u*lambda + -0.898269 *y*pow(lambda, 3) + -0.0344533 *pow(x,
         2)*v*lambda  +  -0.001281  *pow(y,  3)*lambda  +  -74.2945  *pow(u,
         2)*v*lambda + -5.96681 *y*pow(v, 2)*lambda + -0.000877717 *pow(x,
         2)*y*lambda  +  -87.3494  *pow(v,  3)*lambda  +  0.0694095  *pow(x,
         2)*pow(v, 3) + 2.17107 *pow(y, 2)*pow(v, 3) + 1.43477e-006 *pow(y,
         5) + 537.058 *pow(v, 5) + -0.00313171 *pow(x, 3)*u*v + 4.30655e-005
         *pow(x, 3)*y*u + 0.0432255 *pow(y, 3)*pow(v, 2) + 0.000514786
         *pow(y, 4)*v + 9.73322e-005 *pow(x, 2)*pow(y, 2)*v + 53.3919
         *y*pow(v, 4) + 0.00134424 *pow(x, 2)*y*pow(u, 2) + 5.1155e-005
         *x*pow(y, 3)*u + -1.31486e-006 *pow(x, 2)*pow(y, 3)*lambda +
         -1.01892e-006 *pow(x, 4)*y*lambda;
```

```
    float out_u = + 6.66257e-007 + -0.235351 *u + -0.0857718 *x+
          0.00274023 *x*lambda + 0.123574 *x*pow(u, 2) + 0.000100477
          *pow(x,3) + 0.0638468 *x*pow(v, 2) + 0.00347409 *x*y*v +
          0.00418952 *pow(x, 2)*u + 0.000100678 *x*pow(y, 2) + 1.35096
          *pow(u, 3) + 0.00226813 *pow(y, 2)*u + 0.132925 *y*u*v + 2.55673
          *u*pow(v, 2) + -0.162989 *y*u*v*lambda + -3.58109e-005 *pow(x,
          3)*lambda + -3.80909 *pow(u, 3)*lambda + -3.68641 *u*pow(v,
          2)*lambda + -0.00516423 *pow(x, 2)*u*lambda + -0.00169049
          *pow(y, 2)*u*lambda + -0.0799597 *x*pow(v, 2)*lambda + 0.0602954
          *u*pow(lambda, 3) + -0.250181 *x*pow(u, 2)*lambda + -0.00330155
          *x*y*v*lambda + -3.71224e-005 *x*pow(y, 2)*lambda + 0.00465585
          *x*y*pow(u, 2)*v + 0.00198815 *pow(x, 3)*pow(u, 2) + 26.3507
          *pow(u, 5) + -9.65013e-008 *x*pow(y, 4) + -3.72203e-006 *pow(x,
          2)*pow(y, 2)*u + -1.82976e-007 *pow(x, 3)*pow(y, 2) + 0.00367081
          *pow(x, 2)*u*pow(v, 2) + 0.105703 *pow(x, 2)*pow(u, 3) +
          -3.20526e-006 *pow(x, 3)*y*v + 0.00485749 *pow(y, 2)*u*pow(v,
          2) + 0.154521 *y*pow(u, 3)*v + -2.44444e-006 *x*pow(y, 3)*v +
          0.00476159 *pow(y, 2)*pow(u, 3) + 2.62019 *x*pow(u, 4) +
          1.64867e-005 *pow(x, 4)*u;
```

```
    float out_v = + -1.58672e-006 + -0.239495 *v + -0.0856315 *y +
          0.00258882 *y*lambda + 1.35862 *pow(v, 3) + 2.59994 *pow(u, 2)*v
          + 0.0541402 *v*pow(lambda, 2) + 0.0615903 *y*pow(u, 2) +
          9.90232e-005 *pow(y, 3) + 0.00410881 *pow(y, 2)*v + 0.120617
          *y*pow(v, 2) + 0.130525 *x*u*v + 0.00204155 *pow(x, 2)*v +
          0.00329688 *x*y*u + 0.000118196 *pow(x, 2)*y + -0.00507045
          *pow(y,2)*v*lambda + -0.246231 *y*pow(v, 2)*lambda +
          -2.96221e-005 *pow(x, 2)*y*lambda + -3.40051e-005 *pow(y,
          3)*lambda + -0.00319774 *x*y*u*lambda + -0.0783419 *y*pow(u,
          2)*lambda + -0.00170867 *pow(x, 2)*v*lambda + -3.77346 *pow(u,
          2)*v*lambda + -3.83925 *pow(v, 3)*lambda + -0.163451
          *x*u*v*lambda + -6.92239e-008 *pow(x, 4)*y + -4.04945e-006
          *x*pow(y, 3)*u + -4.05089e-006 *pow(x, 3)*y*u + 0.104705 *pow(y,
          2)*pow(v, 3) + 0.0049727 *x*y*u*pow(v, 2) + 1.62885e-005 *pow(y,
          4)*v + 0.00196613 *pow(y, 3)*pow(v, 2) + 0.176651 *x*u*pow(v,
          3) + 0.00540087 *pow(x, 2)*pow(v, 3) + -5.2197e-006 *pow(x,
          2)*pow(y, 2)*v + 0.00417043 *pow(x, 2)*pow(u, 2)*v +
          -1.62266e-007 *pow(x, 2)*pow(y, 3) + 2.59293 *y*pow(v, 4) +
          25.9641 *pow(v, 5) + 0.00343233 *pow(y, 2)*pow(u, 2)*v;
    float out_eta =  + 0.145856  + 6.63004e-005 *y + 0.88191 *lambda +
         0.000273362 *pow(x, 2) + -0.0261669 *pow(u, 2) + -0.0161401
         *pow(v,2) + -0.00221 *x*u + 0.000343732 *pow(y, 2) + -1.20654
         *pow(lambda,2) + 0.578819 *pow(lambda, 3) + -1.1538e-005 *pow(x,
         4) + -6.92348e-006 *pow(x, 2)*pow(y, 2) + -1.33361e-005 *pow(y,
         4) + -7.74626e-005 *pow(y, 3)*u + -9.56815e-008 *pow(x, 3)*y +
         3.01117e-005 *pow(x, 3)*u + -1.31781e-007 *pow(y, 5) +
         1.49459e-007 *pow(y, 6) + 1.36809e-007 *pow(x, 6) + -4.14133e-008
         *pow(x,4)* pow(y, 2) + 3.57831e-007 *x*pow(y, 4)*u + 2.84773e-006
         *pow(y, 5)*u + 4.73212e-007 *pow(x, 2)*pow(y, 3)*pow(u, 2) +
         2.00941e-009 *pow(y, 7) + 9.76305e-008 *pow(x, 2)*pow(y,
         4)*pow(u, 2) + -3.13293e-008 *pow(y, 7)*u + -5.60404e-010 *pow(y,
         8) + -3.93646e-010 *pow(x, 2)*pow(y, 6) + 1.25312e-009 *pow(y,
         7)*v + -5.45484e-010 *pow(x, 8) + 6.60148e-009 *pow(x, 4)*pow(y,
         4) + -7.62066e-012 *pow(y, 9) + -2.32591e-010 *pow(x, 2)*pow(y,
         6)*u + -1.57219e-010 *pow(x, 3)*pow(y, 6) + 1.05084e-010 *pow(y,
         9)*u + -3.22853e-011 *pow(x, 4)*pow(y, 6) + -3.80463e-011 *pow(x,
         6)*pow(y, 4) + 5.25106e-010 *pow(x, 3)*pow(y, 6)*lambda +
         -1.70284e-011 *pow(y, 9)*u*v + -4.30621e-010
         *pow(x,3)*pow(y,6)*pow(lambda, 2);
```