Sparse high-degree polynomials for wide-angle lenses

Emanuel Schrade, Johannes Hanika
and Carsten Dachsbacher
computer graphics lab
Karlsruhe Institute of Technology

motivation

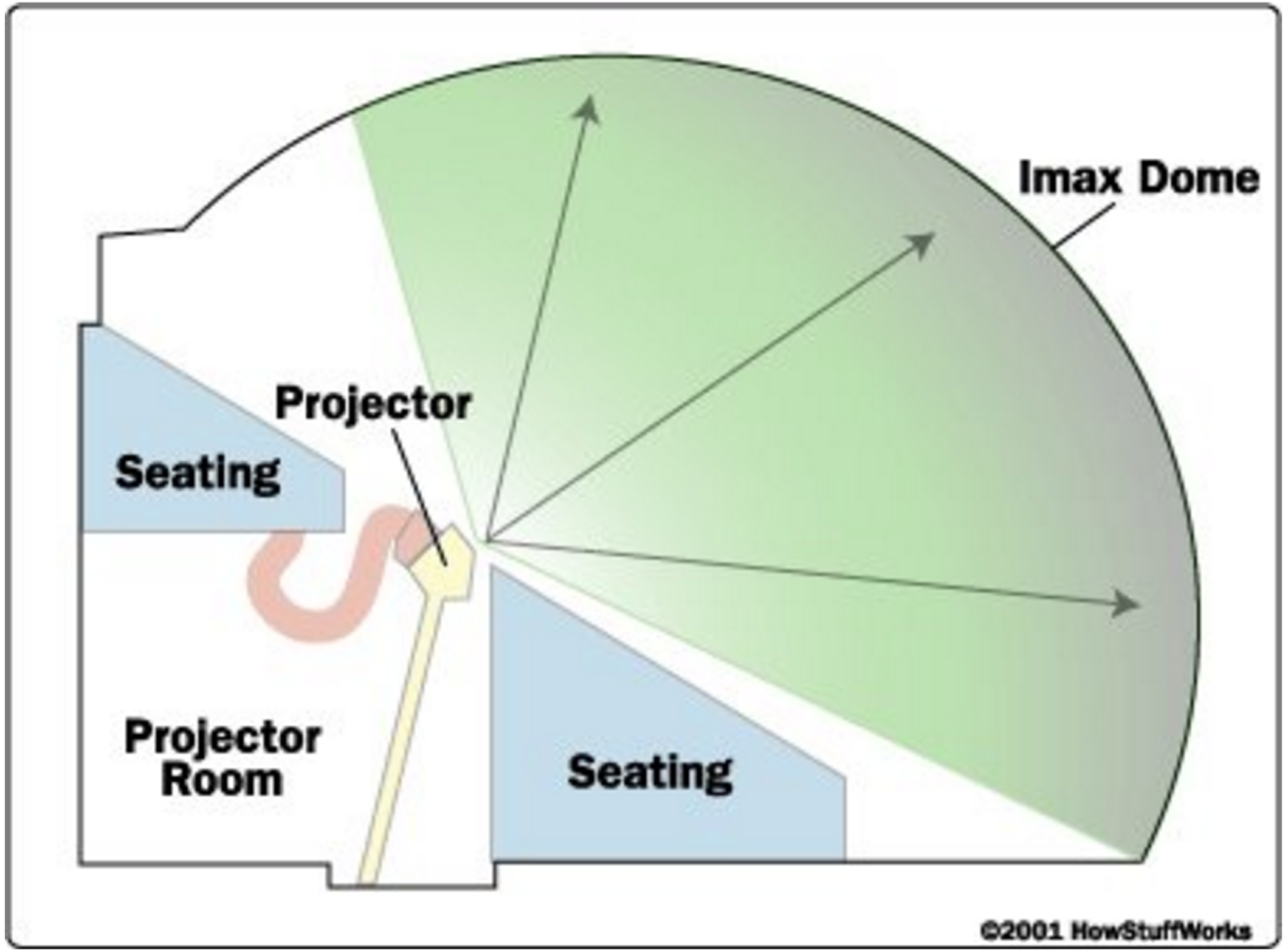
fisheye lenses

motivation

panoramic cameras

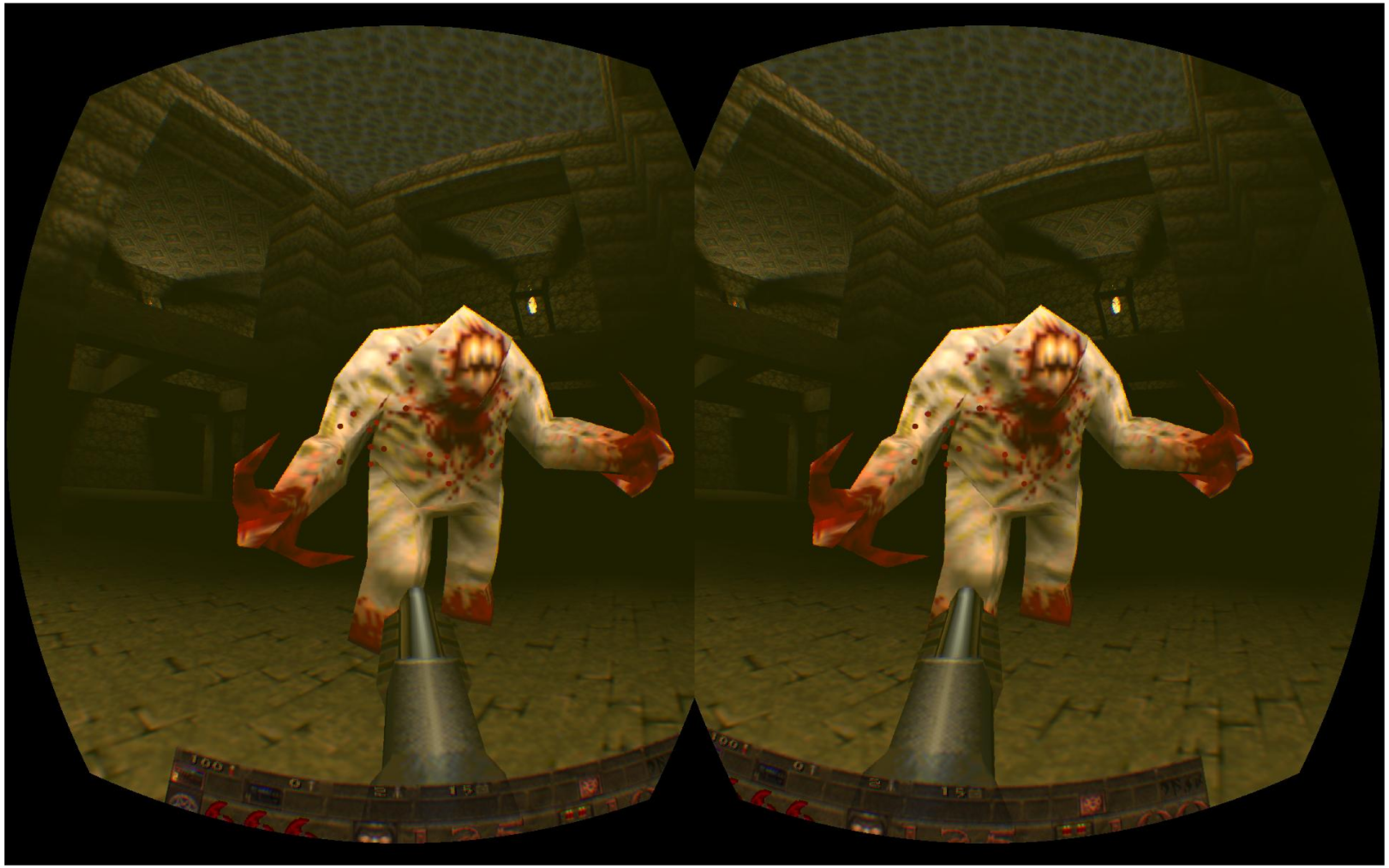
motivation

rendering for the imax dome



motivation

rendering for virtual reality



motivation

photorealistic rendering

- flat and boring bokeh

motivation

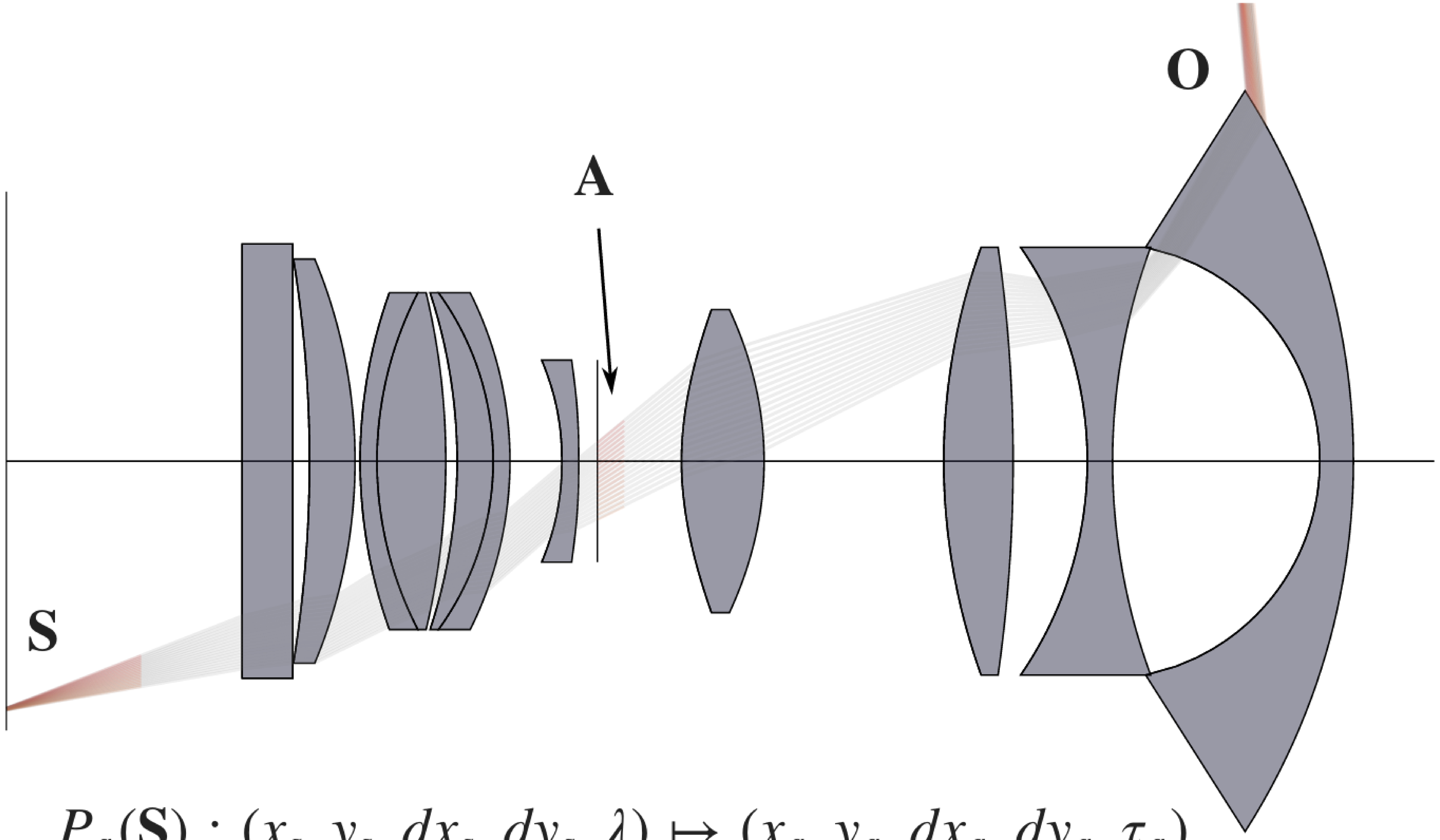
photorealistic rendering

- interesting bokeh, distortion, and vignetting to match plate

state of the art

approximate lens systems with simple polynomial

- collapse complicated ray tracing
- simple function evaluations $\mathbf{A} = P_a(\mathbf{S})$ and $\mathbf{O} = P_o(\mathbf{S})$

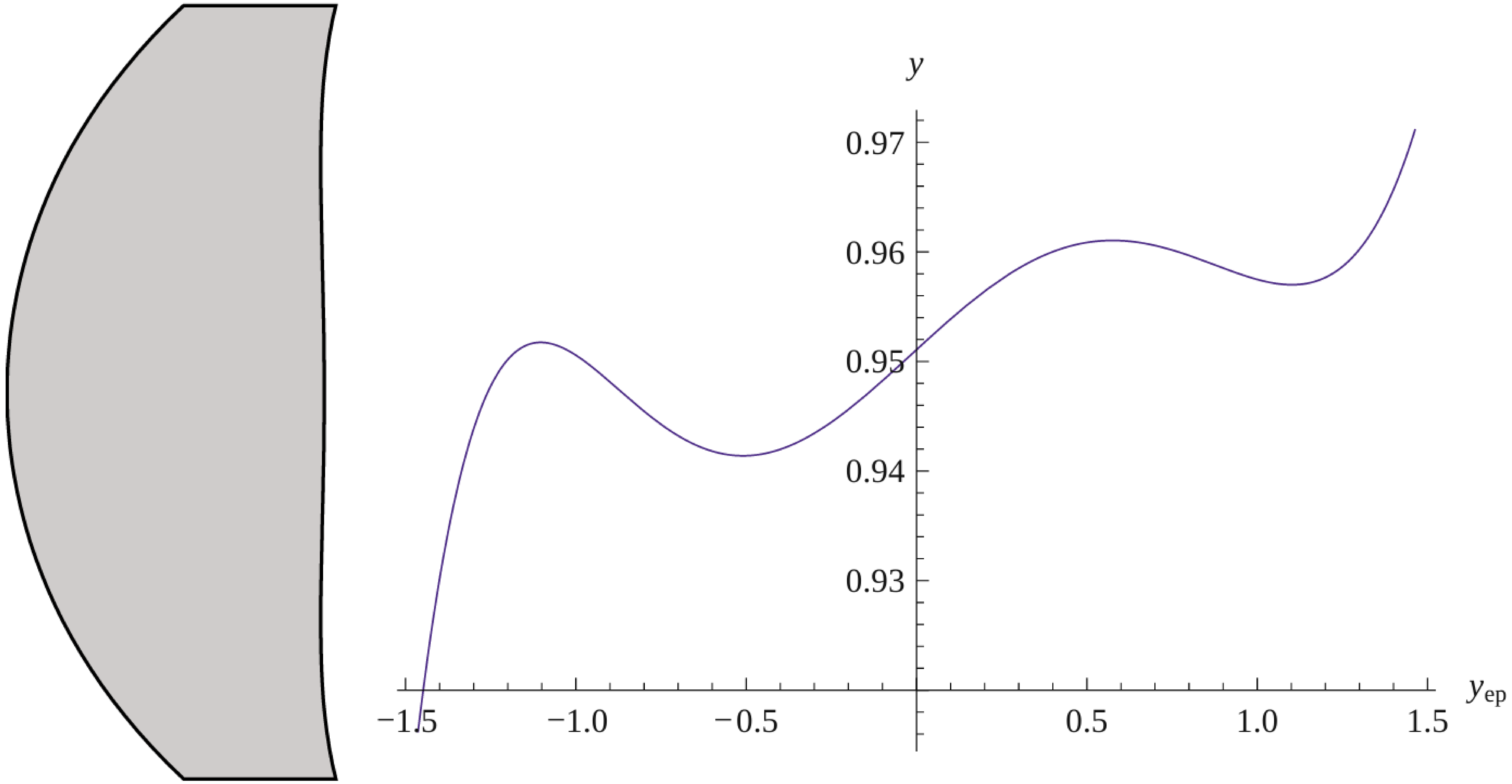


$$P_a(\mathbf{S}) : (x_s, y_s, dx_s, dy_s, \lambda) \mapsto (x_a, y_a, dx_a, dy_a, \tau_a)$$
$$P_o(\mathbf{S}) : (x_s, y_s, dx_s, dy_s, \lambda) \mapsto (x_o, y_o, dx_o, dy_o, \tau_o)$$

state of the art

optics use polynomials to ray trace

- approximate ray tracing using polynomials for lens designers [ZHB10]
- and analyse error in that domain directly

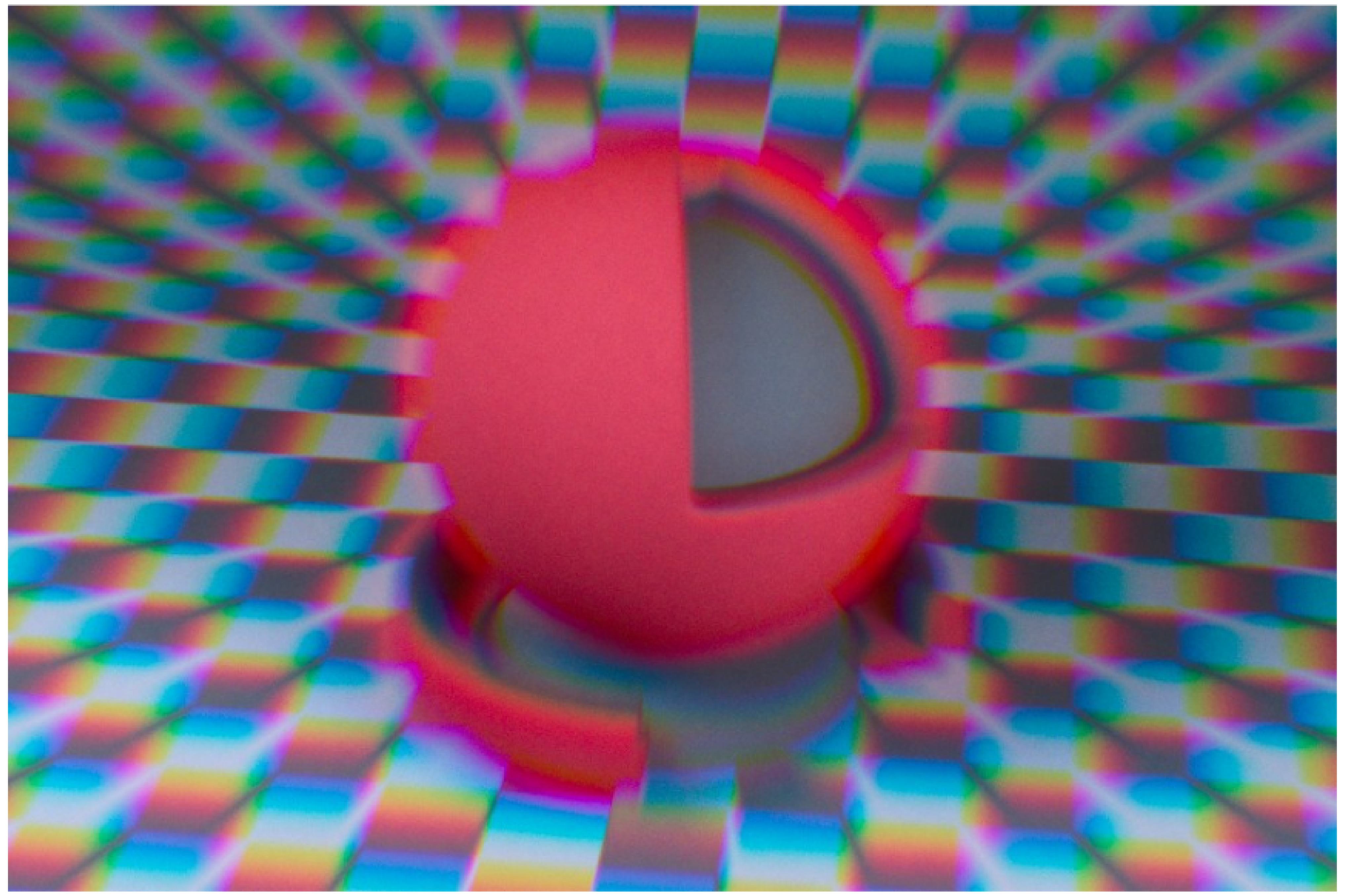


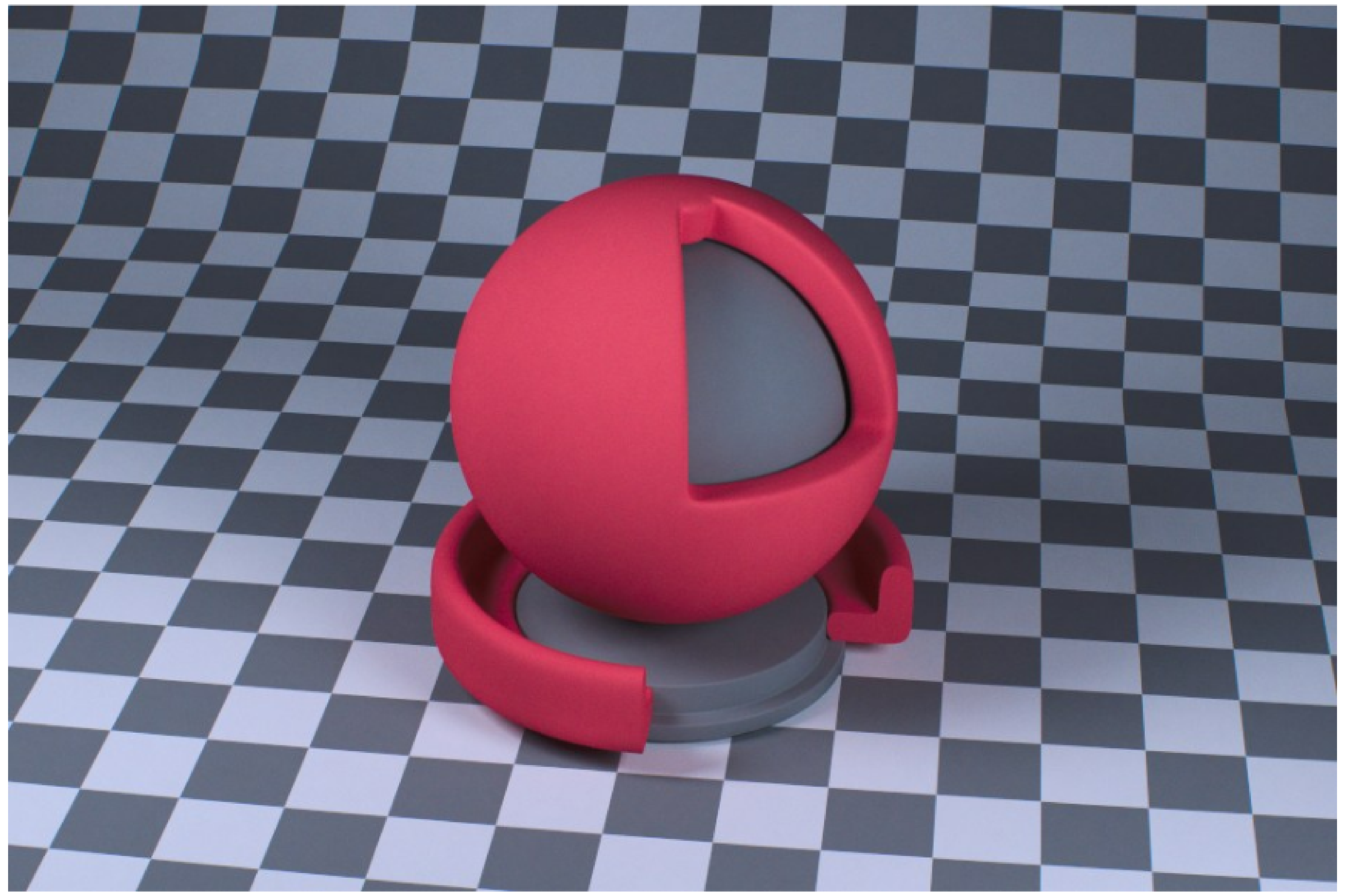
(a) (b)

state of the art

all based on Taylor expansion

- scary formulas for analytic differentiation required!
- not precise in outer rims [HD14]
    - use Taylor configuration, optimise coefficients





state of the art

Taylor polynomials

- don't like cumbersome analytic Taylor expansions
  - needs pen and paper or computer algebra software
  - expand every lens element, insert, re-truncate polynomials
- want polynomial with only few coefficients (fast evaluation!)
- and more precision (analytic Taylor expansion hardly tractable for high degree)
  - idea: select from higher-degree terms

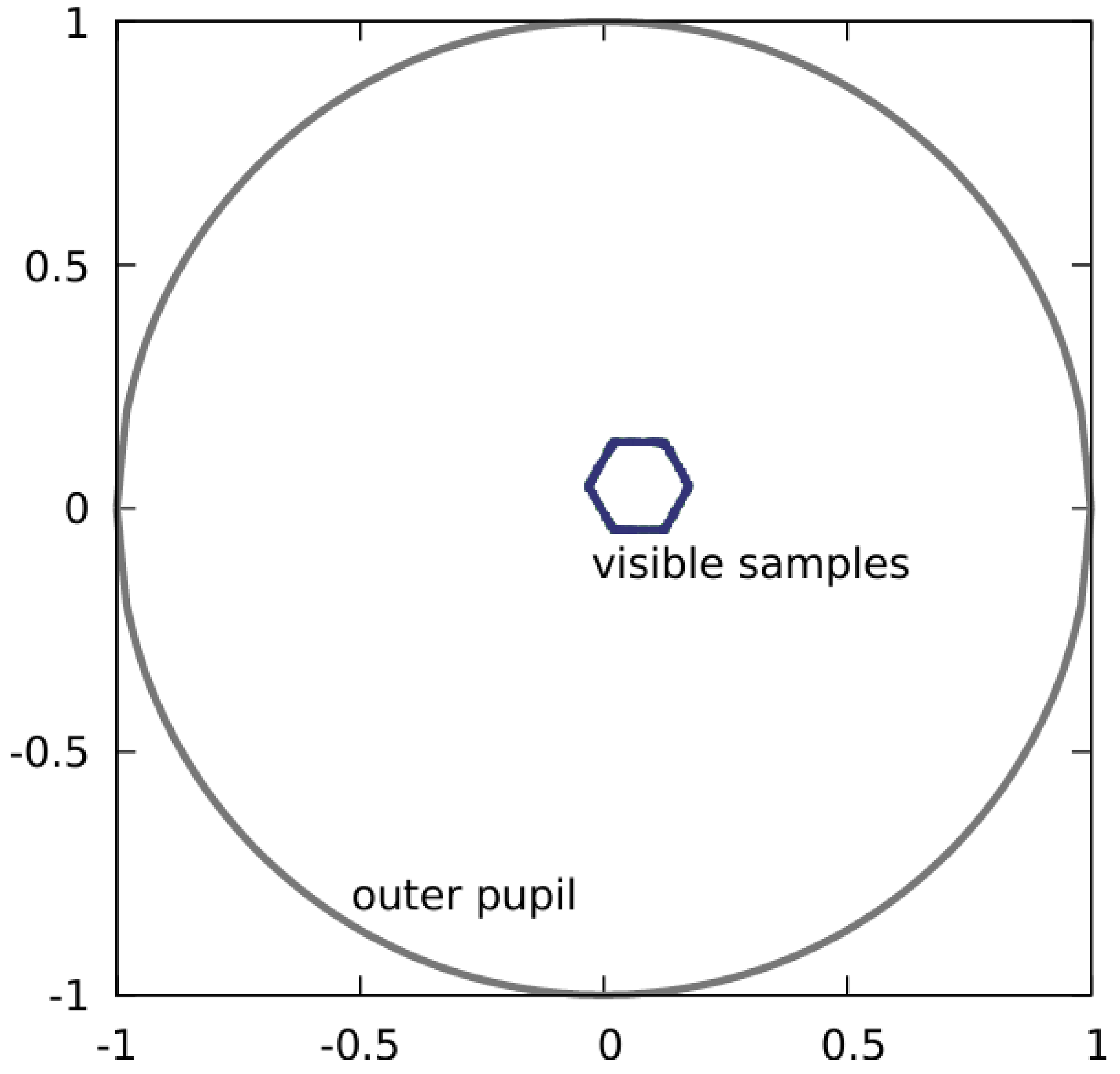
$$\cos x = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n$$

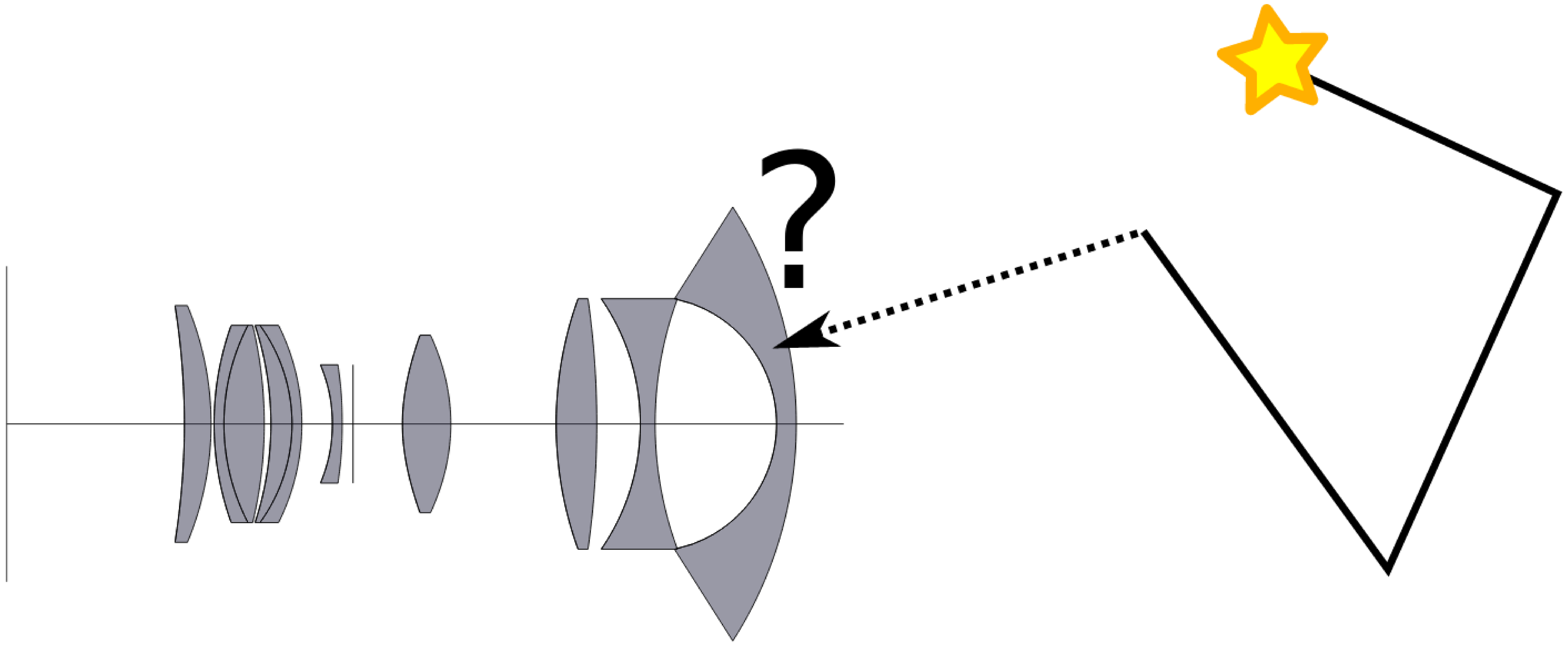
$$= f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{(4)}(0)}{4!}x^4 + \frac{f^{(5)}(0)}{5!}x^5 + \cdots$$

state of the art

fisheye lenses

- precision in periphery *is* important!
- current lens connections (for light tracing)
  - sample outer pupil uniformly
  - have terrible performance



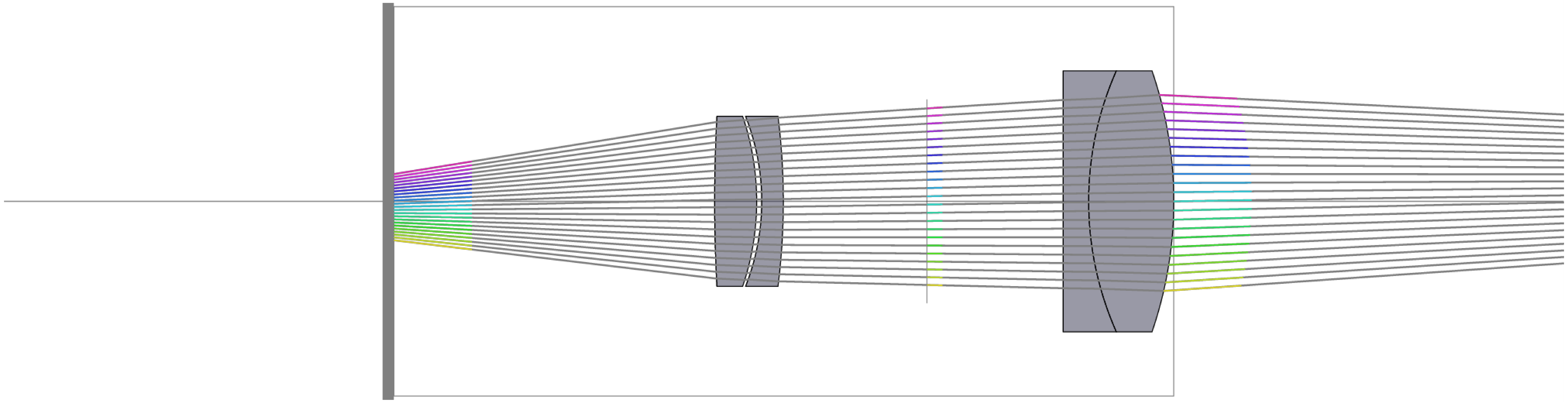


state of the art

sampling the outer pupil

- why didn't we bother earlier?
  - not such a bad strategy for long lenses

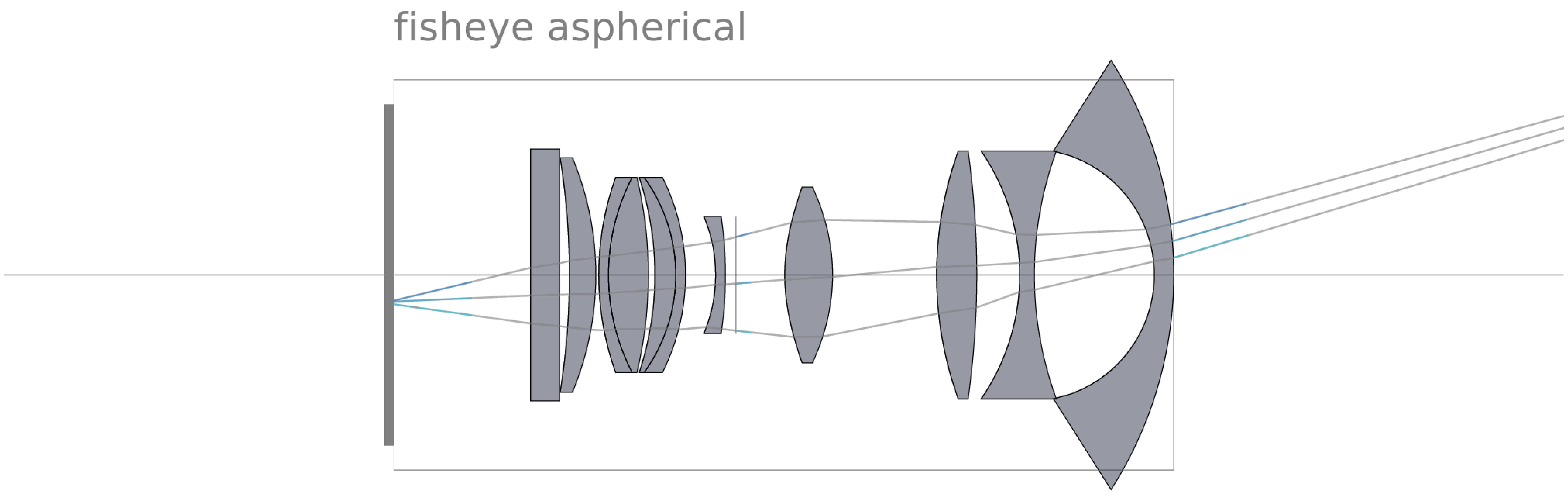
petzval kodak



state of the art

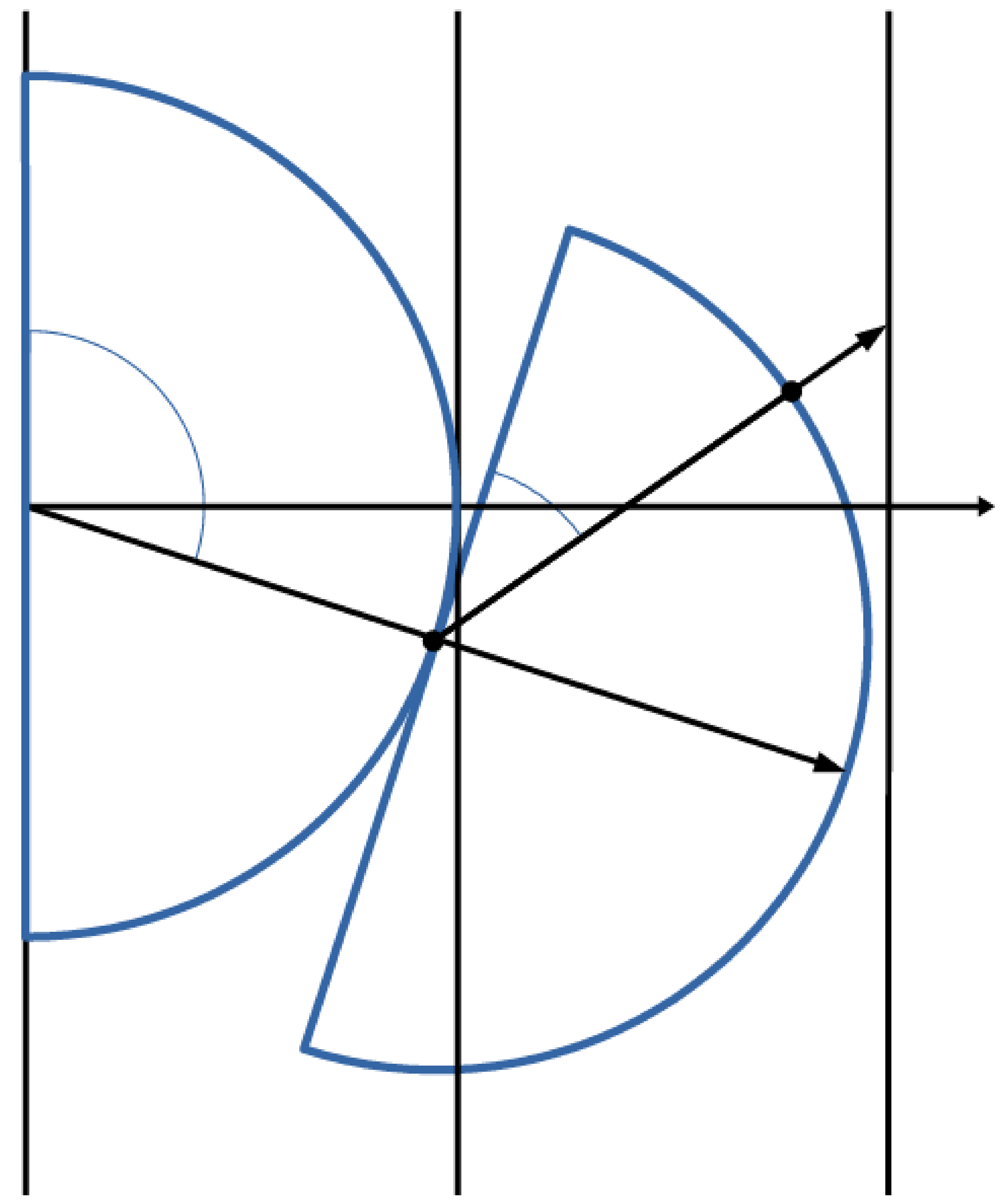
sampling the outer pupil

- why didn't we bother earlier?
  - but terrible for fisheyes



technical contributions

- parametrise the light fields for fisheyes
  - no plane/plane 180-degree limit

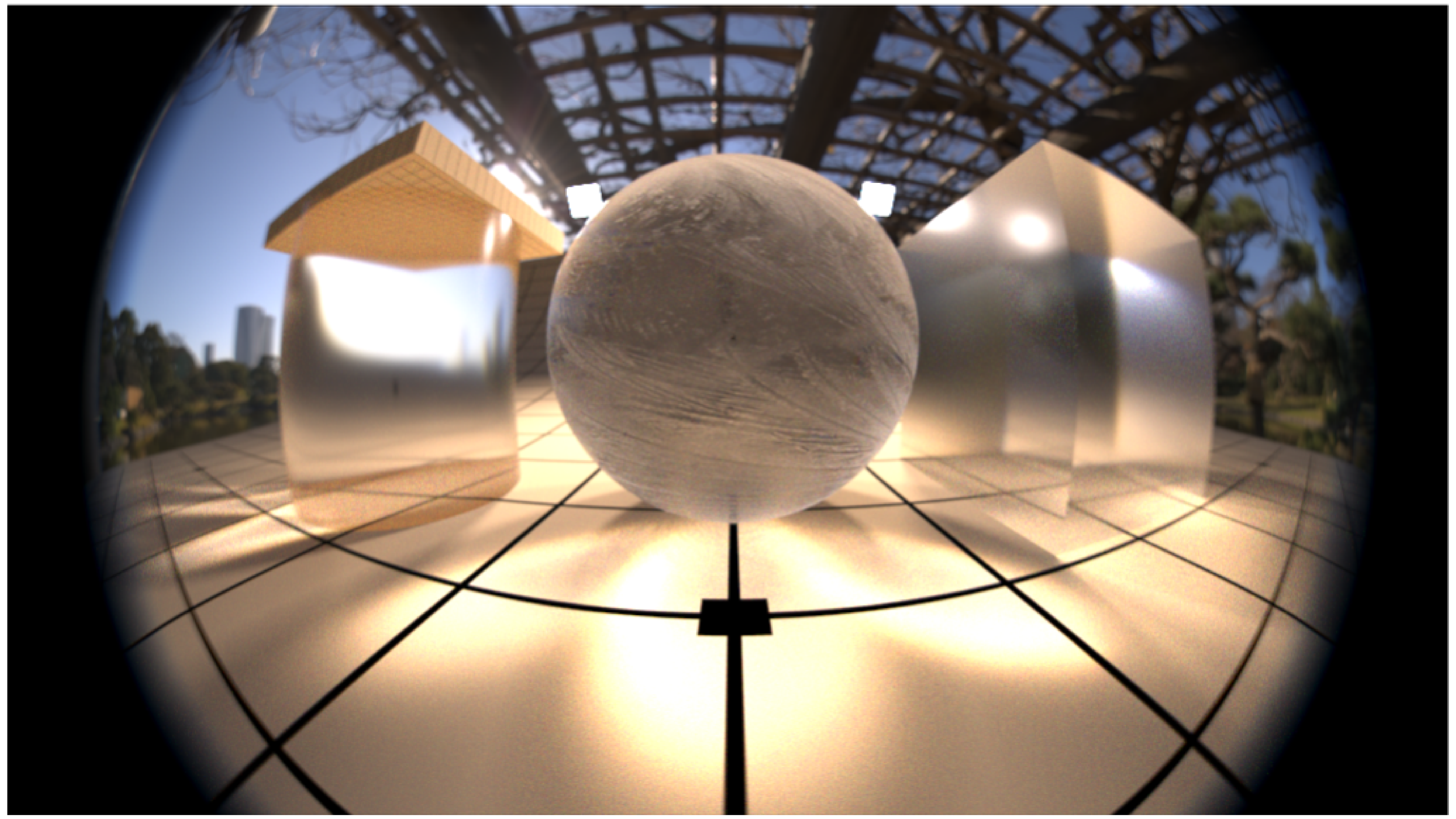


technical contributions

- parametrise the light fields for fisheyes
  - no plane/plane 180-degree limit
- sparse fitting of high-degree polynomials
  - use orthogonal matching pursuit (OMP)
  - enables trade-off between approximation error and evaluation speed

technical contributions

- parametrise the light fields for fisheyes
  - no plane/plane 180-degree limit
- sparse fitting of high-degree polynomials
  - use orthogonal matching pursuit (OMP)
  - enables trade-off between approximation error and evaluation speed
- aperture sampling for light tracing
  - enable the use in bidirectional path tracing etc.

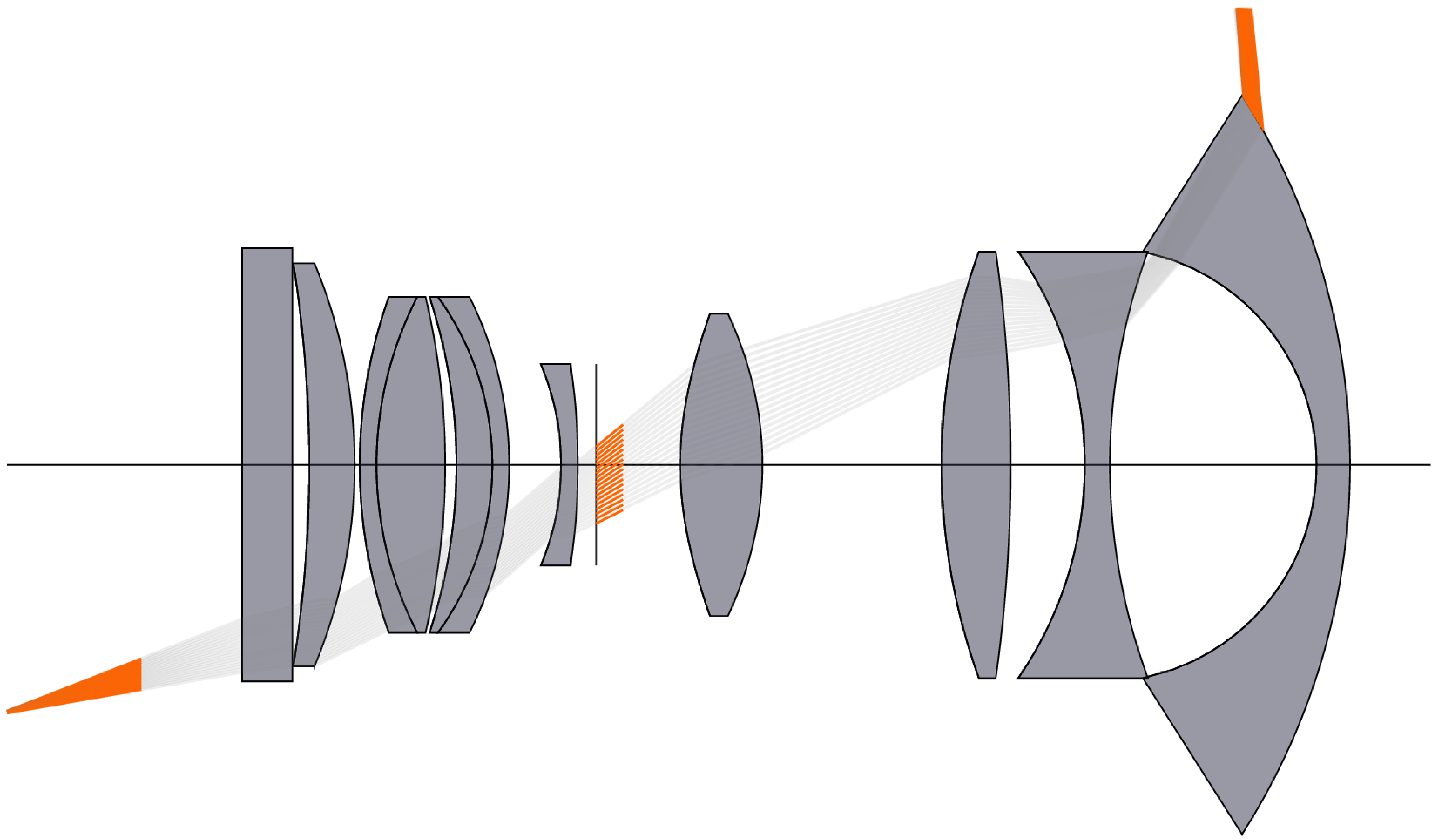


technical contributions

- parametrise the light fields for fisheyes
  - no plane/plane 180-degree limit
- sparse fitting of high-degree polynomials
  - use orthogonal matching pursuit (OMP)
  - enables trade-off between approximation error and evaluation speed
- aperture sampling for light tracing
  - enable the use in bidirectional path tracing etc.
- fast GPU preview rendering implementation

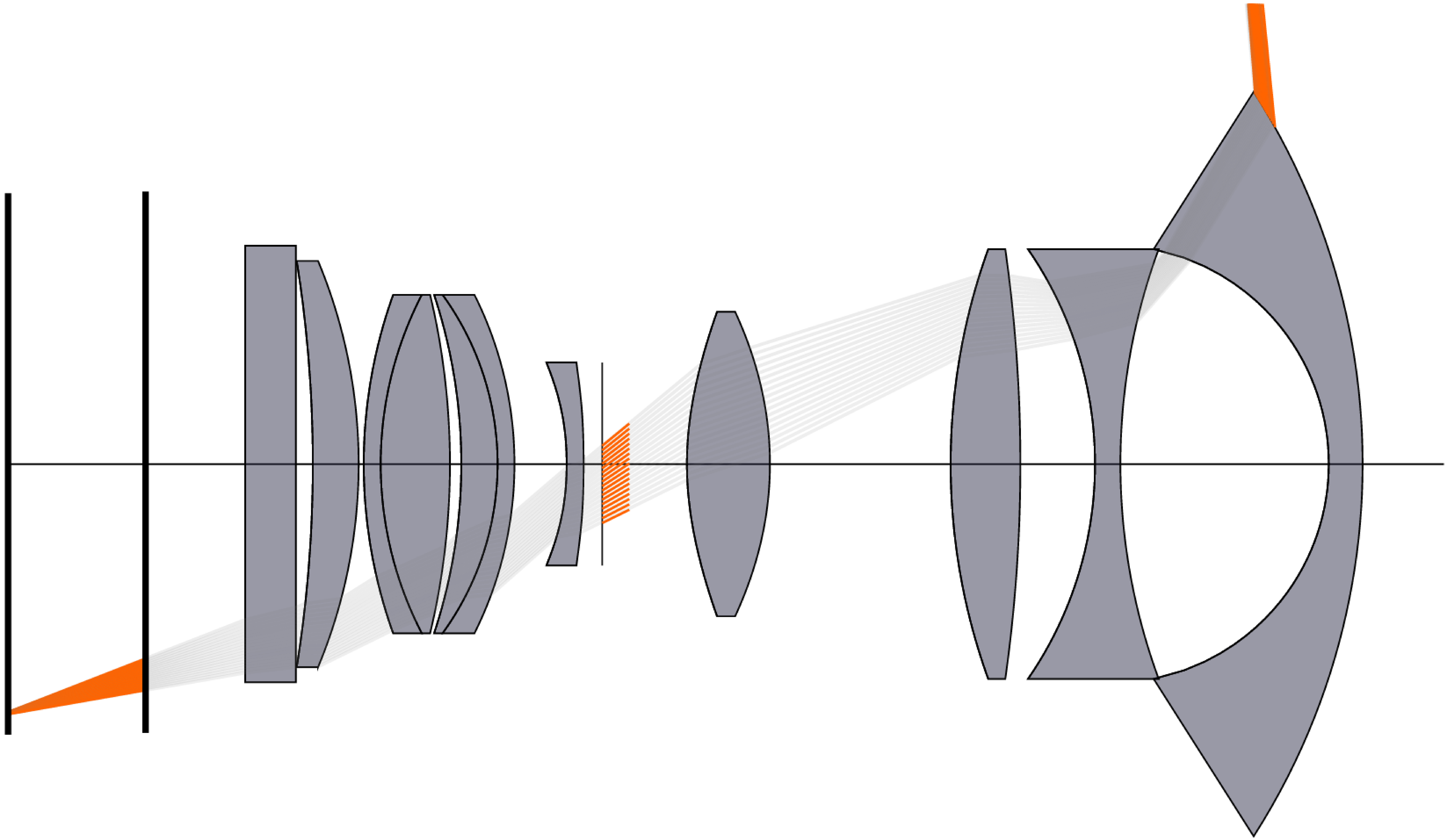
parametrisation suitable for fisheye lenses

parametrisation suitable for fisheye lenses



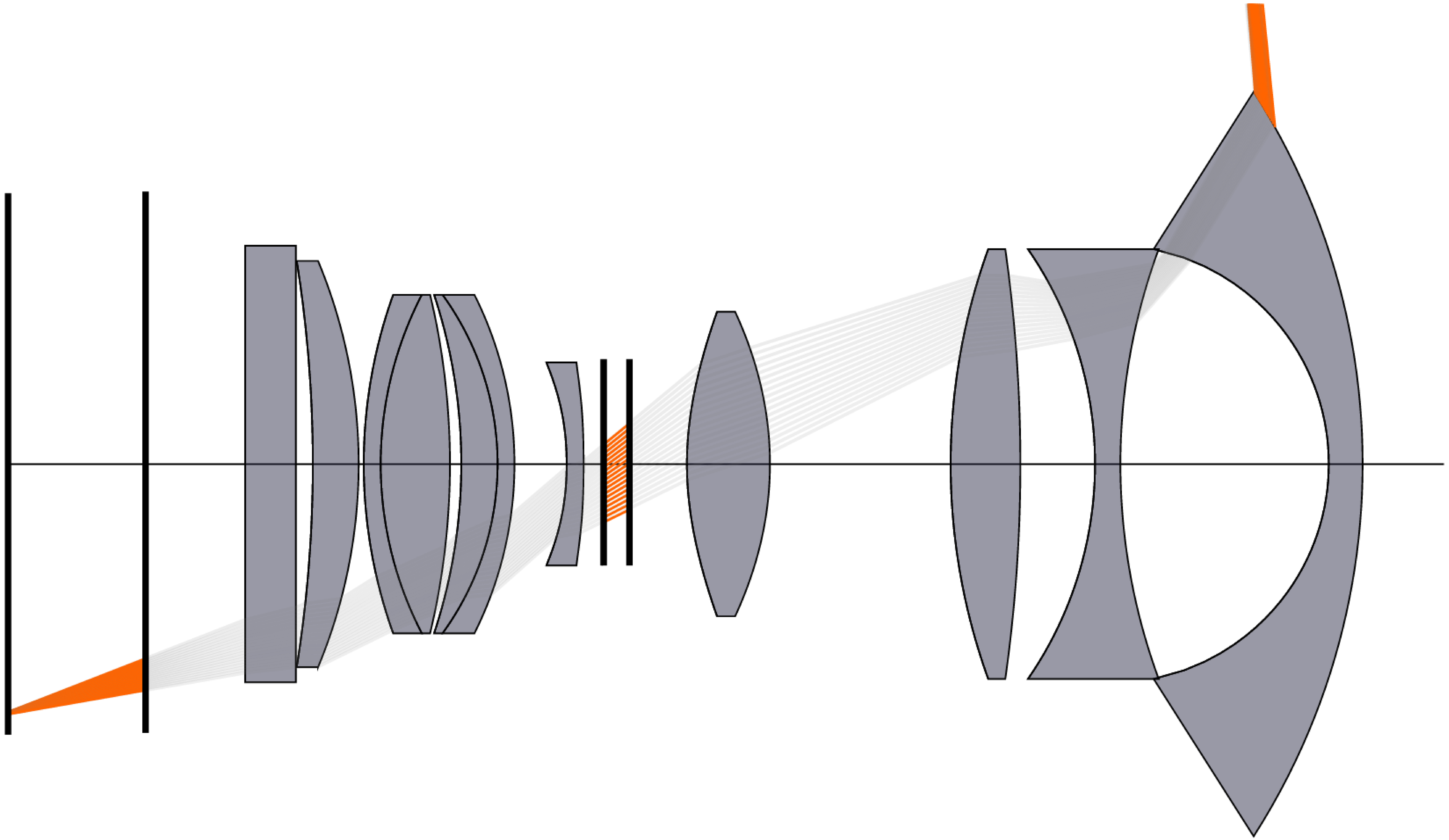
parametrisation suitable for fisheye lenses

- plane/plane on sensor



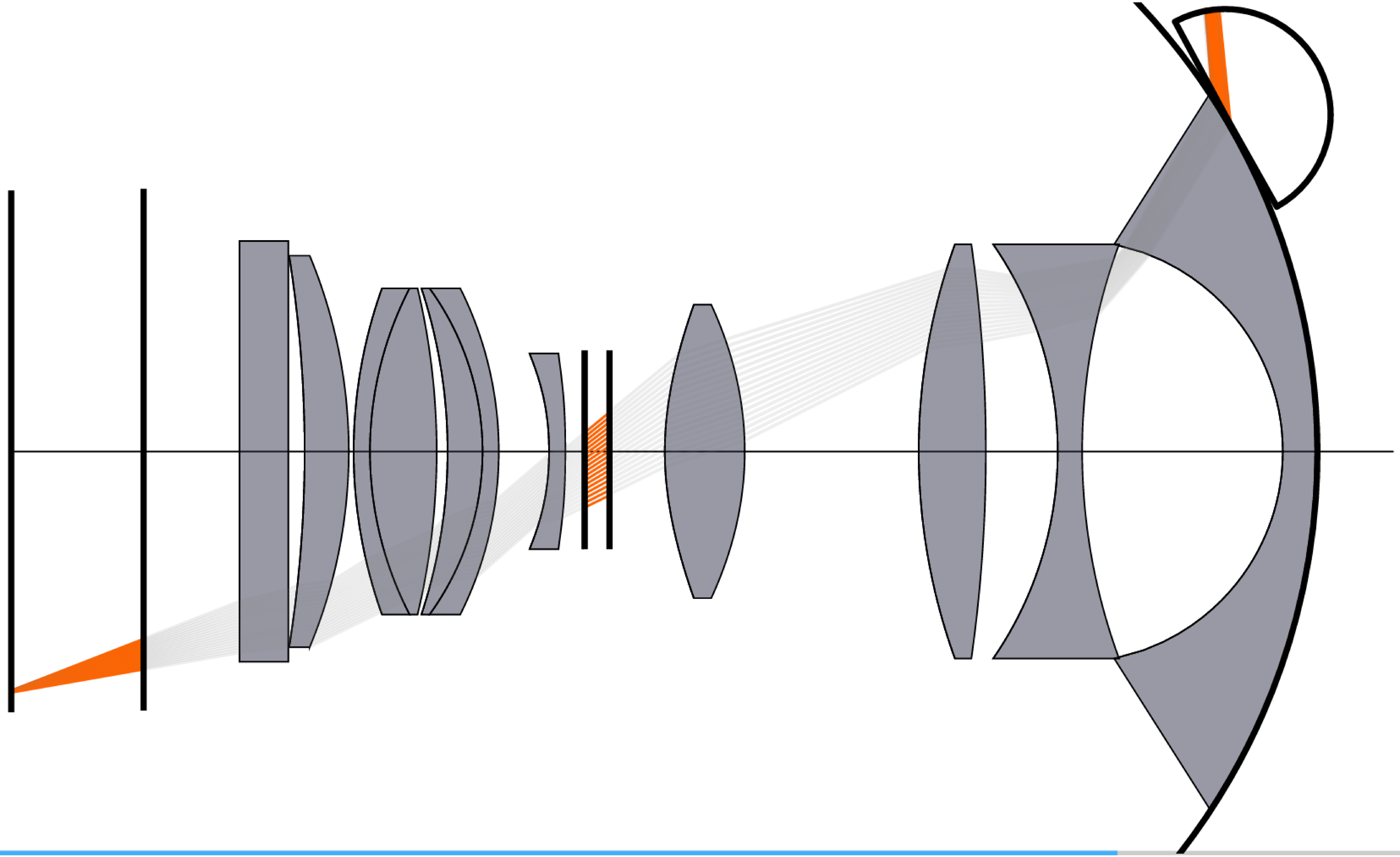
parametrisation suitable for fisheye lenses

- plane/plane on sensor
- plane/plane on aperture



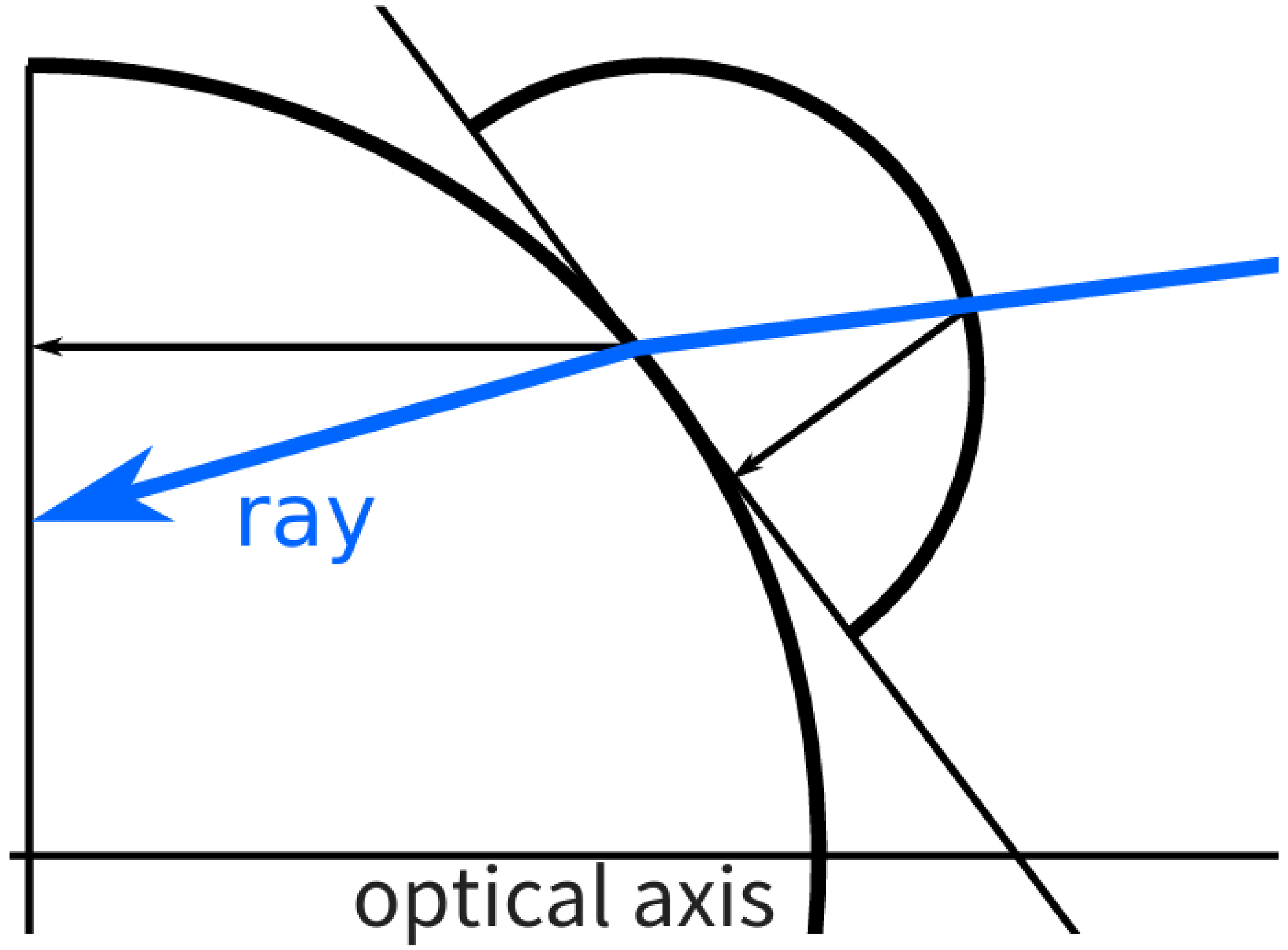
parametrisation suitable for fisheye lenses

- plane/plane on sensor
- plane/plane on aperture
- hemi-sphere/hemi-sphere on outer pupil



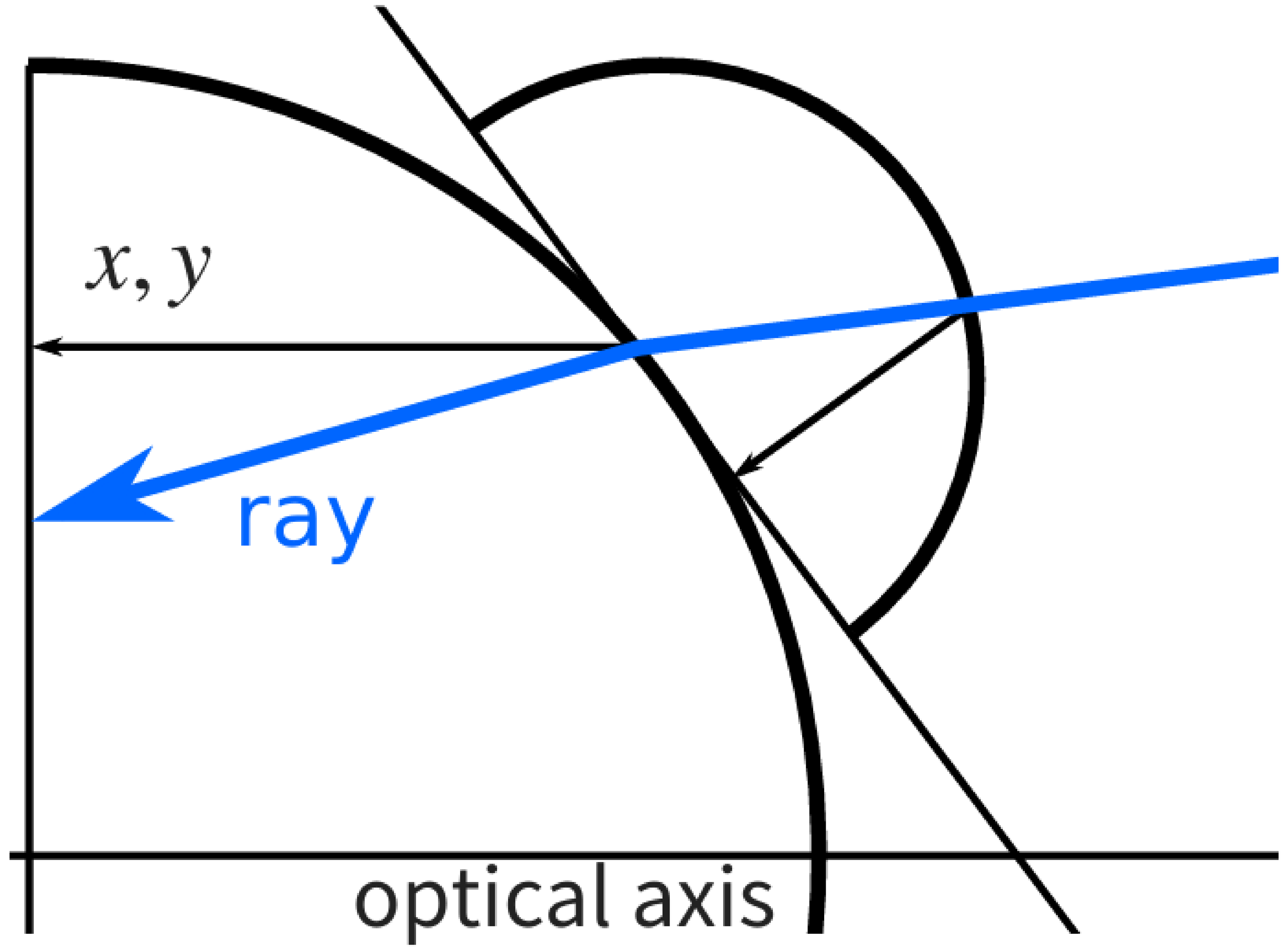
parametrisation suitable for fisheye lenses

hemi-sphere/hemi-sphere on outer pupil



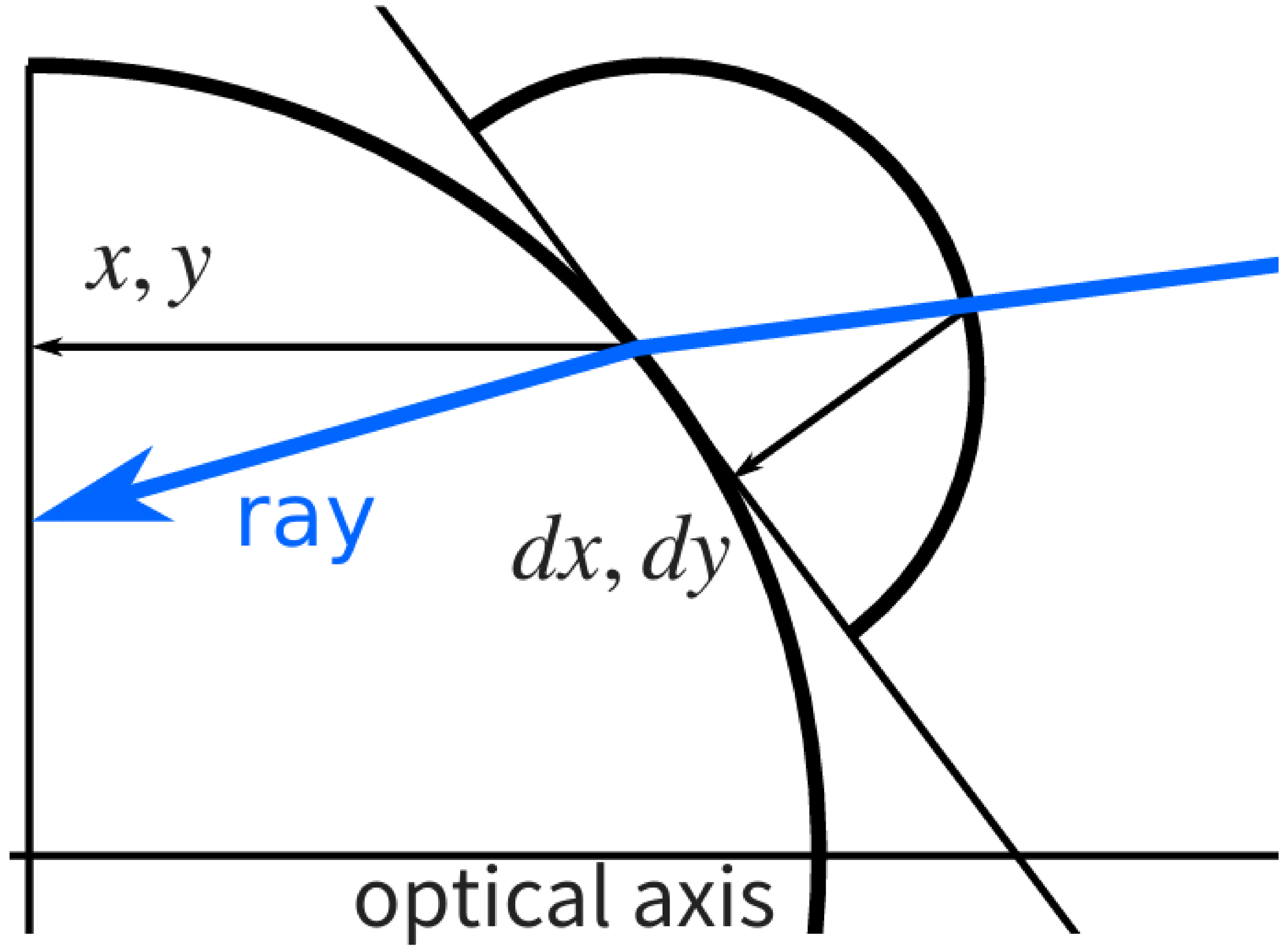
parametrisation suitable for fisheye lenses

hemi-sphere/hemi-sphere on outer pupil



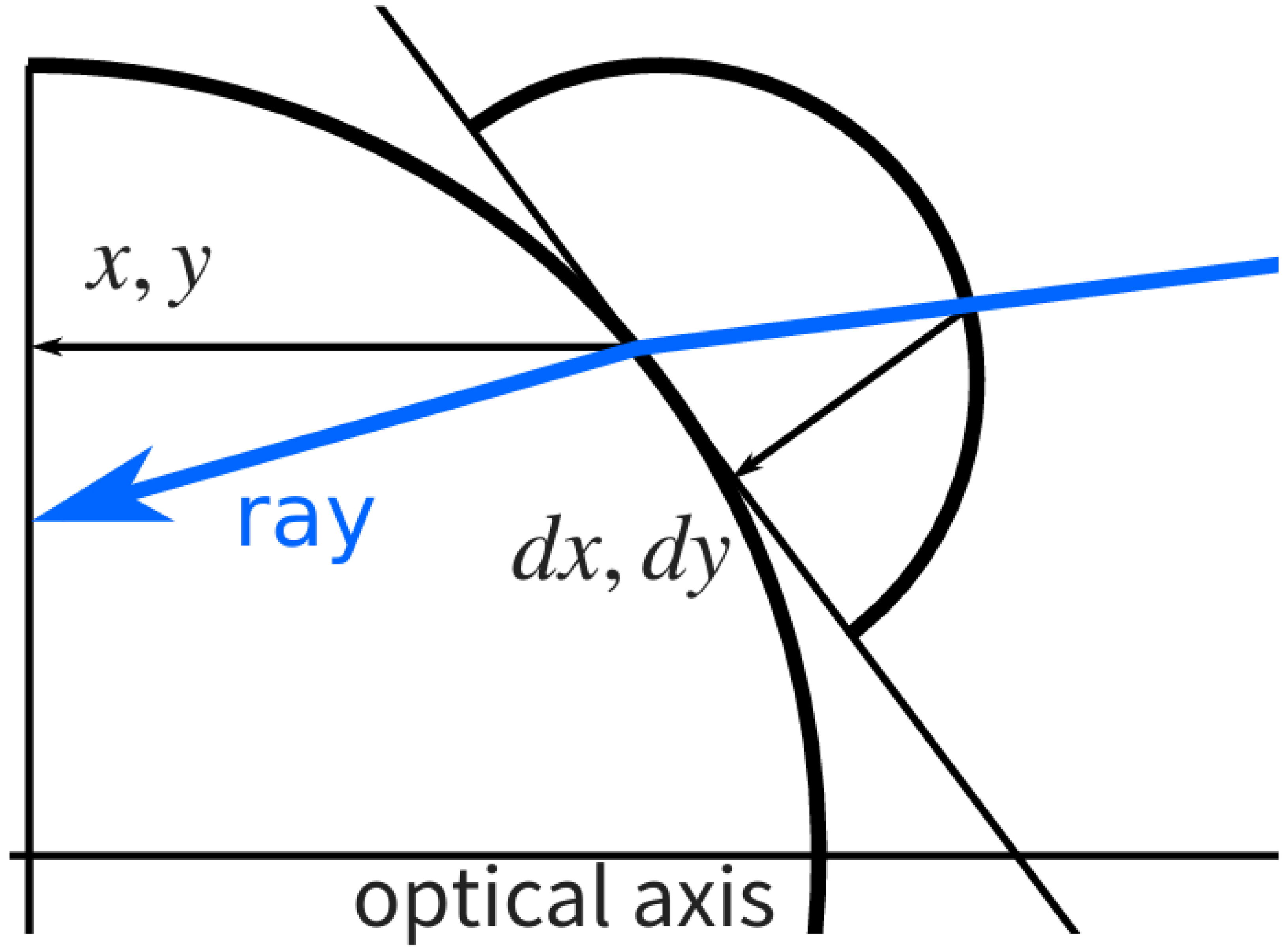
parametrisation suitable for fisheye lenses

hemi-sphere/hemi-sphere on outer pupil



parametrisation suitable for fisheye lenses

hemi-sphere/hemi-sphere on outer pupil



- need to specify tangent frame for dx, dy
- avoid the singularity in interesting regions on the outer pupil

parametrisation suitable for fisheye lenses

finding a polynomial

- polynomial consists of these terms:

$$c \cdot \underbrace{x_s^{d_0} y_s^{d_1} dx_s^{d_2} dy_s^{d_3} \lambda_s^{d_4}}_{=:T_t} \text{ with degree } \sum_{i=0}^{4} d_i \leq d$$

finding a polynomial

- polynomial consists of these terms:

$$c \cdot \underbrace{x_s^{d_0} y_s^{d_1} dx_s^{d_2} dy_s^{d_3} \lambda_s^{d_4}}_{=:T_t} \text{ with degree } \sum_{i=0}^{4} d_i \leq d$$

- find most closely matching polynomial for given set of ray traced reference samples
  - linear problem, Galerkin projection of function $\mathbf{O} = P_o(\mathbf{S})$ to

$$\mathbf{O} \approx \hat{\Phi} \cdot \mathbf{c}$$

with

finding a polynomial

- polynomial consists of these terms:

$$c \cdot \underbrace{x_s^{d_0} y_s^{d_1} dx_s^{d_2} dy_s^{d_3} \lambda_s^{d_4}}_{=:T_t} \text{ with degree } \sum_{i=0}^{4} d_i \leq d$$

- find most closely matching polynomial for given set of ray traced reference samples
  - linear problem, Galerkin projection of function $\mathbf{O} = P_o(\mathbf{S})$ to

$$\mathbf{O} \approx \hat{\Phi} \cdot \mathbf{c}$$

with

$$\hat{\Phi} = \begin{pmatrix} T_1 & T_2 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & \cdots & T_{N-1} & T_N \\ \cdots & & & & \\ T_1 & T_2 & \cdots & T_{N-1} & T_N \end{pmatrix}$$

finding a polynomial

- polynomial consists of these terms:

$$c \cdot \underbrace{x_s^{d_0} y_s^{d_1} dx_s^{d_2} dy_s^{d_3} \lambda_s^{d_4}}_{=:T_t} \text{ with degree } \sum_{i=0}^{4} d_i \leq d$$

- find most closely matching polynomial for given set of ray traced reference samples
  - linear problem, Galerkin projection of function $\mathbf{O} = P_o(\mathbf{S})$ to

$$\mathbf{O} \approx \hat{\Phi} \cdot \mathbf{c}$$

    with

$$\hat{\Phi} = \begin{pmatrix} T_1 & T_2 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & \cdots & T_{N-1} & T_N \\ \cdots & & & & \\ T_1 & T_2 & \cdots & T_{N-1} & T_N \end{pmatrix}$$

  - each row in $\hat{\Phi}$ corresponds to one input sample ($10 \times N$ ray traced references)

finding a polynomial

- polynomial consists of these terms:

$$c \cdot \underbrace{x_s^{d_0} y_s^{d_1} dx_s^{d_2} dy_s^{d_3} \lambda_s^{d_4}}_{=:T_t} \text{ with degree } \sum_{i=0}^{4} d_i \leq d$$

- find most closely matching polynomial for given set of ray traced reference samples
  - linear problem, Galerkin projection of function $\mathbf{O} = P_o(\mathbf{S})$ to

$$\mathbf{O} \approx \hat{\Phi} \cdot \mathbf{c}$$

  with

$$\hat{\Phi} = \begin{pmatrix} T_1 & T_2 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & \cdots & T_{N-1} & T_N \\ \cdots & & & & \\ T_1 & T_2 & \cdots & T_{N-1} & T_N \end{pmatrix}$$

  - each row in $\hat{\Phi}$ corresponds to one input sample ($10 \times N$ ray traced references)
  - N depends on the max degree d as $N(d) = \binom{n+d}{d} = 4368$ (for 5 variables and degree $d = 11$)

finding a polynomial

- polynomial consists of these terms:

$$c \cdot \underbrace{x_s^{d_0} y_s^{d_1} dx_s^{d_2} dy_s^{d_3} \lambda_s^{d_4}}_{=:T_t} \text{ with degree } \sum_{i=0}^{4} d_i \leq d$$

- find most closely matching polynomial for given set of ray traced reference samples
  - linear problem, Galerkin projection of function $\mathbf{O} = P_o(\mathbf{S})$ to

$$\mathbf{O} \approx \hat{\Phi} \cdot \mathbf{c}$$

    with

$$\hat{\Phi} = \begin{pmatrix} T_1 & T_2 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & \cdots & T_{N-1} & T_N \\ \cdots & & & & \\ T_1 & T_2 & \cdots & T_{N-1} & T_N \end{pmatrix}$$

  - each row in $\hat{\Phi}$ corresponds to one input sample ($10 \times N$ ray traced references)
  - N depends on the max degree d as $N(d) = \binom{n+d}{d} = 4368$ (for 5 variables and degree $d = 11$)
- standard procedure (linear least squares), but the matrix is too large for our taste!

finding a *sparse* polynomial

- use *orthogonal matching pursuit* [TG07]
- iteratively select most important columns in $\hat{\Phi}$

$$\hat{\Phi} \cdot \mathbf{c} = \begin{pmatrix} T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ \cdots & & & & & \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ . \\ . \\ c_{N-1} \\ c_N \end{pmatrix} \approx \mathbf{O}$$

finding a *sparse* polynomial

- use *orthogonal matching pursuit* [TG07]
- iteratively select most important columns in $\hat{\Phi}$

$$\hat{\Phi} \cdot \mathbf{c} = \begin{pmatrix} T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ \cdots & & & & & \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ . \\ . \\ c_{N-1} \\ c_N \end{pmatrix} \approx \mathbf{O}$$

finding a *sparse* polynomial

- use *orthogonal matching pursuit* [TG07]
- iteratively select most important columns in $\hat{\Phi}$

$$\hat{\Phi} \cdot \mathbf{c} = \begin{pmatrix} T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ \cdots & & & & & \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ . \\ . \\ c_{N-1} \\ c_N \end{pmatrix} \approx \mathbf{O}$$

finding a *sparse* polynomial

- use *orthogonal matching pursuit* [TG07]
- iteratively select most important columns in $\hat{\Phi}$

$$\hat{\Phi} \cdot \mathbf{c} = \begin{pmatrix} T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ \cdots & & & & & \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ . \\ . \\ c_{N-1} \\ c_N \end{pmatrix} \approx \mathbf{O}$$

finding a *sparse* polynomial

- use *orthogonal matching pursuit* [TG07]
- iteratively select most important columns in $\hat{\Phi}$

$$\hat{\Phi} \cdot \mathbf{c} = \begin{pmatrix} T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \\ \cdots & & & & & \\ T_1 & T_2 & T_3 & \cdots & T_{N-1} & T_N \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ . \\ . \\ c_{N-1} \\ c_N \end{pmatrix} \approx \mathbf{O}$$

- original just looks for largest impact on residual (fast)
- we got better results by re-fitting all coefficents c of all previously selected columns in the inner loop (somewhat slower)
- details see the paper

finding a *sparse* polynomial

- use *orthogonal matching pursuit* [TG07]
- iteratively select most important columns in $\hat{\Phi}$

$$\hat{\Phi} \cdot \mathbf{c} = \begin{pmatrix} T_1 & T_3 & \cdots & T_N \\ T_1 & T_3 & \cdots & T_N \\ \cdots & & & \\ T_1 & T_3 & \cdots & T_N \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_3 \\ . \\ c_N \end{pmatrix} \approx \mathbf{O}$$

- we use up to 40 coefficients per equation (out of 4368 for degree 11)

finding a *sparse* polynomial

- use *orthogonal matching pursuit* [TG07]
- iteratively select most important columns in $\hat{\Phi}$

$$\hat{\Phi} \cdot \mathbf{c} = \begin{pmatrix} T_1 & T_3 & \cdots & T_N \\ T_1 & T_3 & \cdots & T_N \\ \cdots & & & \\ T_1 & T_3 & \cdots & T_N \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_3 \\ . \\ c_N \end{pmatrix} \approx \mathbf{O}$$

- we use up to 40 coefficients per equation (out of 4368 for degree 11)
- works transparently for aspheric and anamorphic lens elements
  - in particular no analytic Taylor expansion required!

finding a *sparse* polynomial

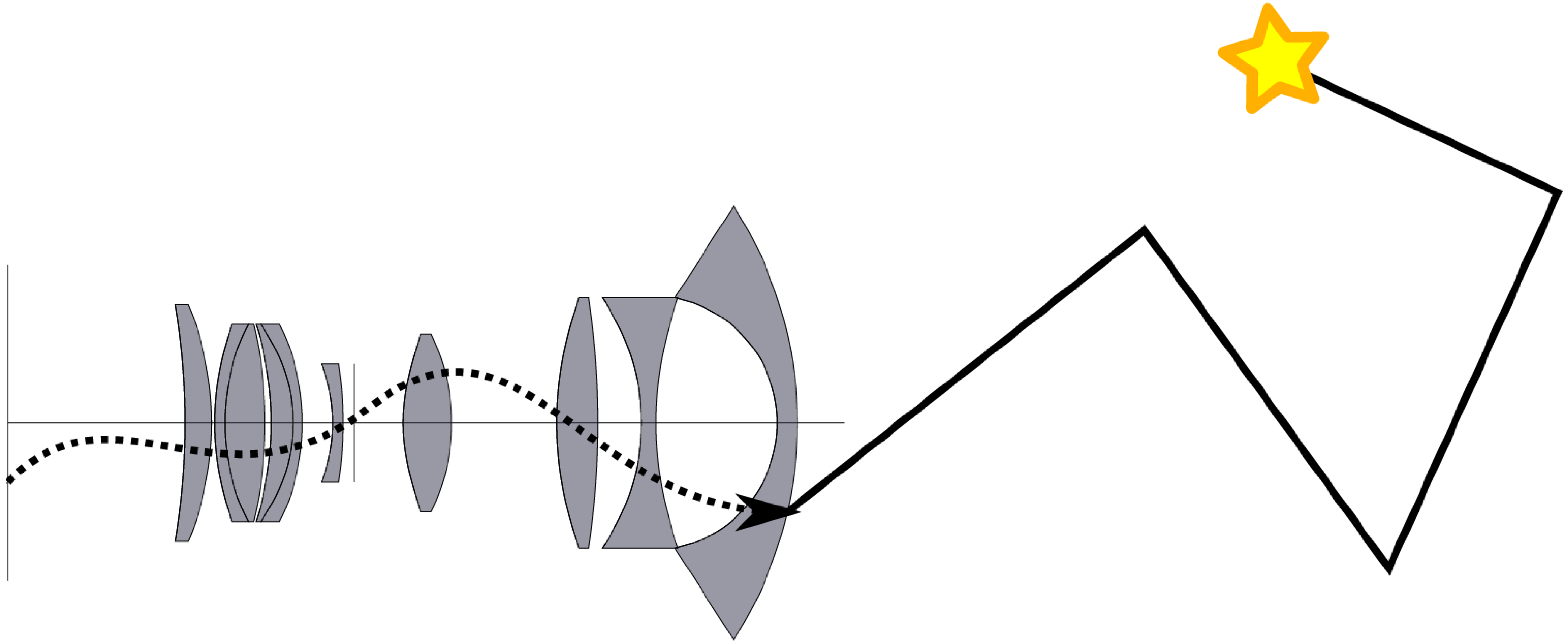
- use *orthogonal matching pursuit* [TG07]
- iteratively select most important columns in $\hat{\Phi}$

$$\hat{\Phi} \cdot \mathbf{c} = \begin{pmatrix} T_1 & T_3 & \cdots & T_N \\ T_1 & T_3 & \cdots & T_N \\ \cdots & & & \\ T_1 & T_3 & \cdots & T_N \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_3 \\ . \\ c_N \end{pmatrix} \approx \mathbf{O}$$

- we use up to 40 coefficients per equation (out of 4368 for degree 11)
- works transparently for aspheric and anamorphic lens elements
  - in particular no analytic Taylor expansion required!
- we also fit Fresnel transmittance τ to support coatings

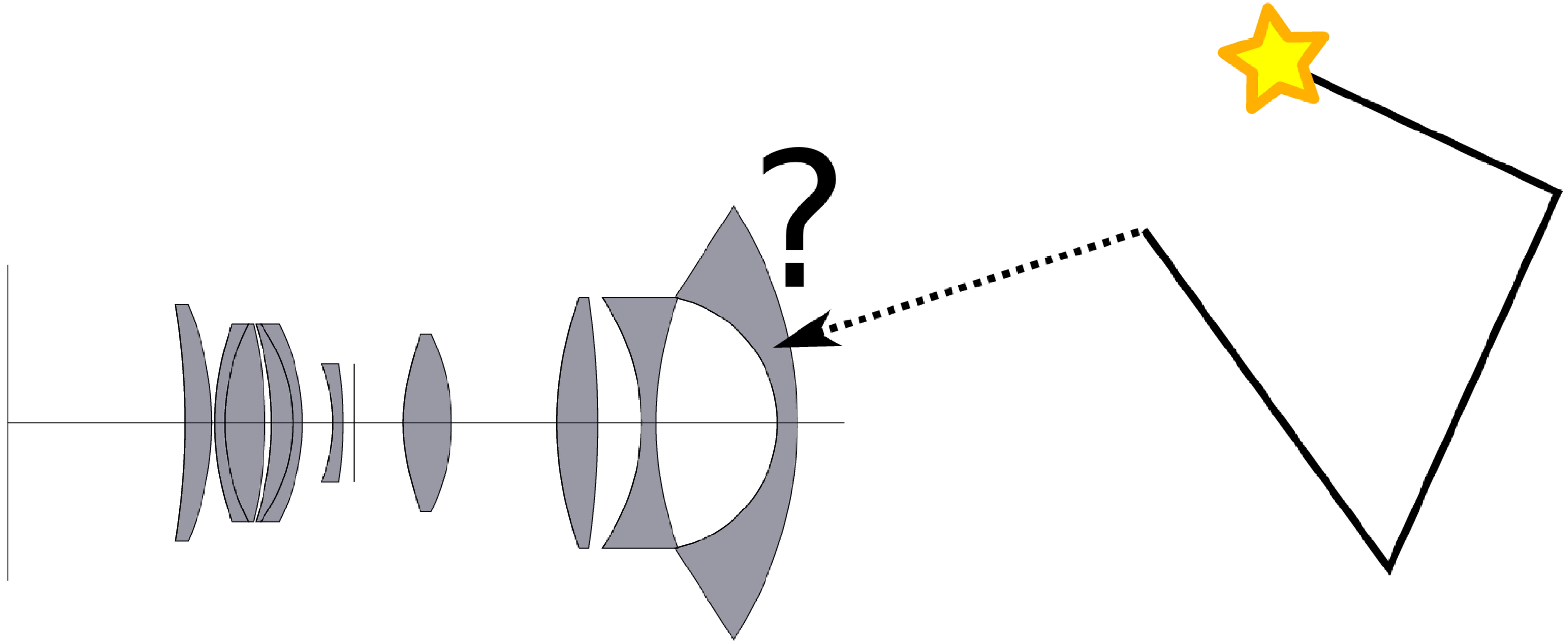
use polynomials for path tracing

- we know how do do this for path tracing from the camera
- efficiently by sampling the aperture [HD14]:
  - sample point on aperture
  - iteratively find position and direction on sensor
- require derivatives of polynomial
  - Newton's method



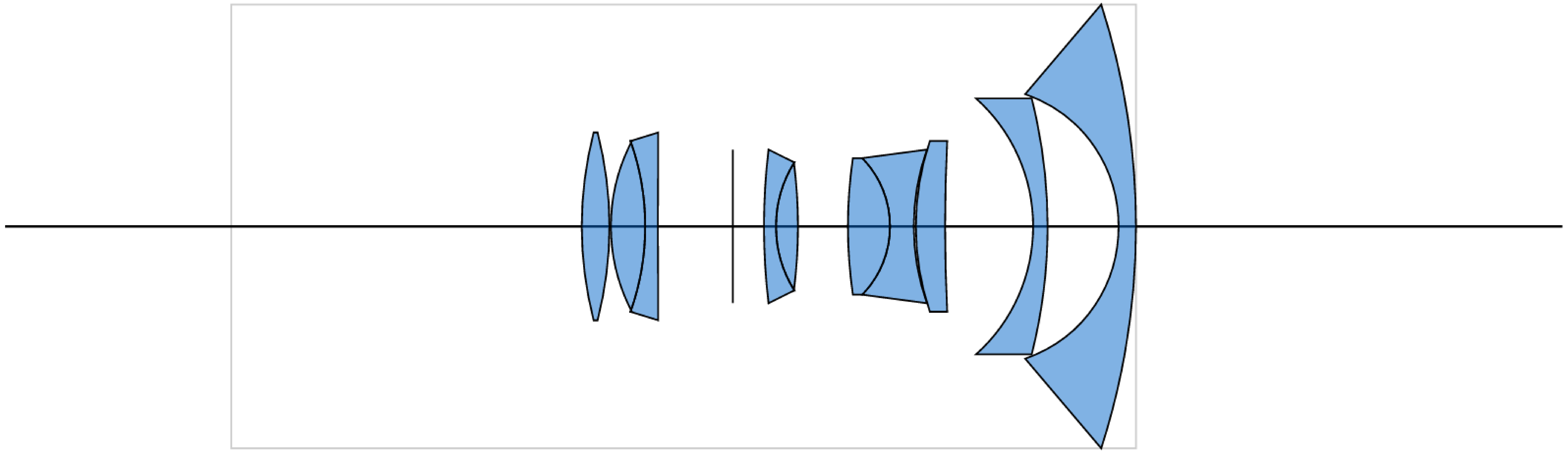
use polynomials for light tracing

- light tracing/deterministic camera connection?
  - sample point on aperture
  - keep point in scene fixed
  - iteratively find position and direction on sensor
- transform probability densities for multiple importance sampling
  - details see the paper



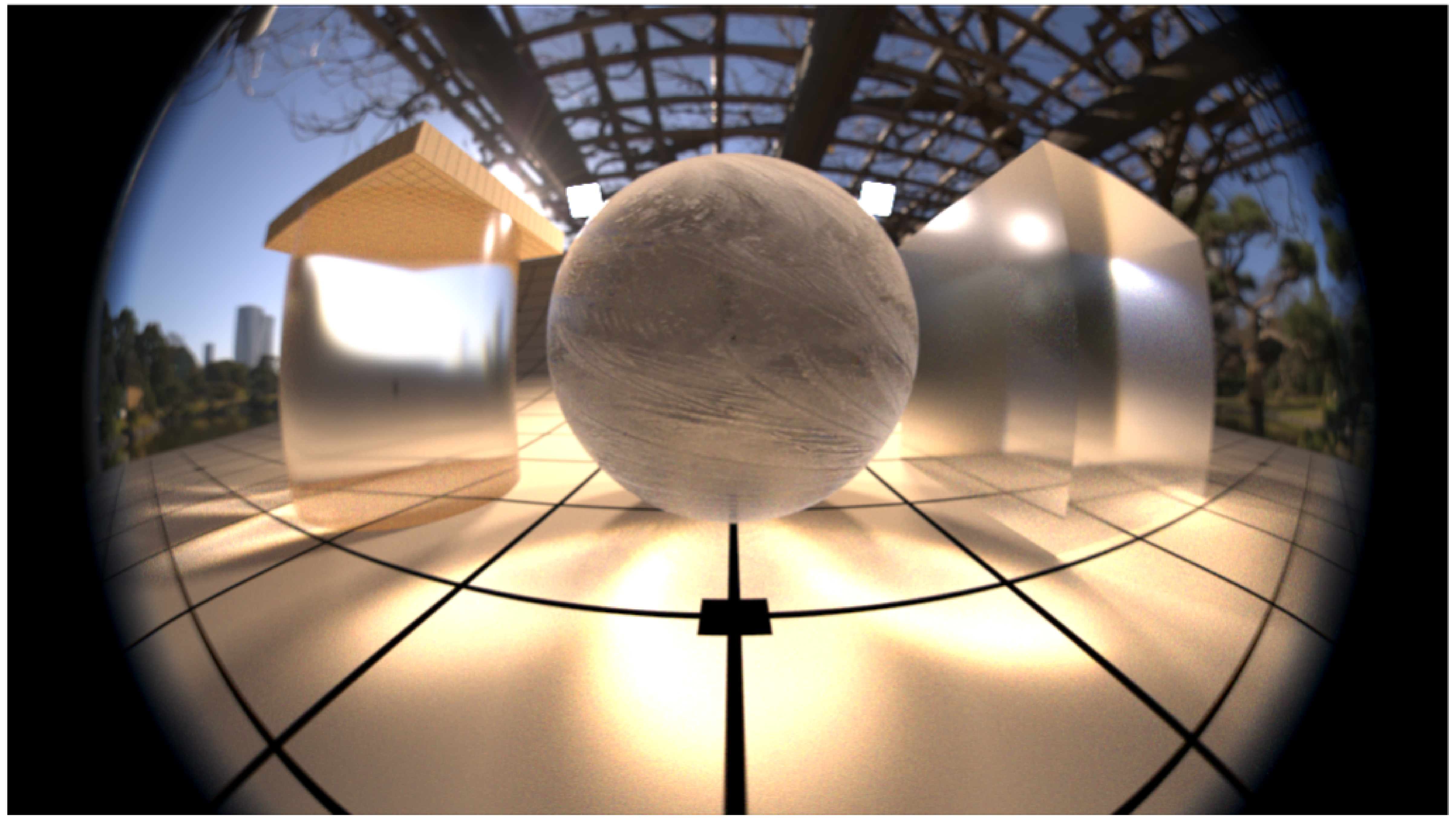
aperture sampling via 2-step Newton iteration

- initial guess: straight on optical axis
- aperture error ⇒ update sensor direction
- error in outgoing direction ⇒ update sensor position

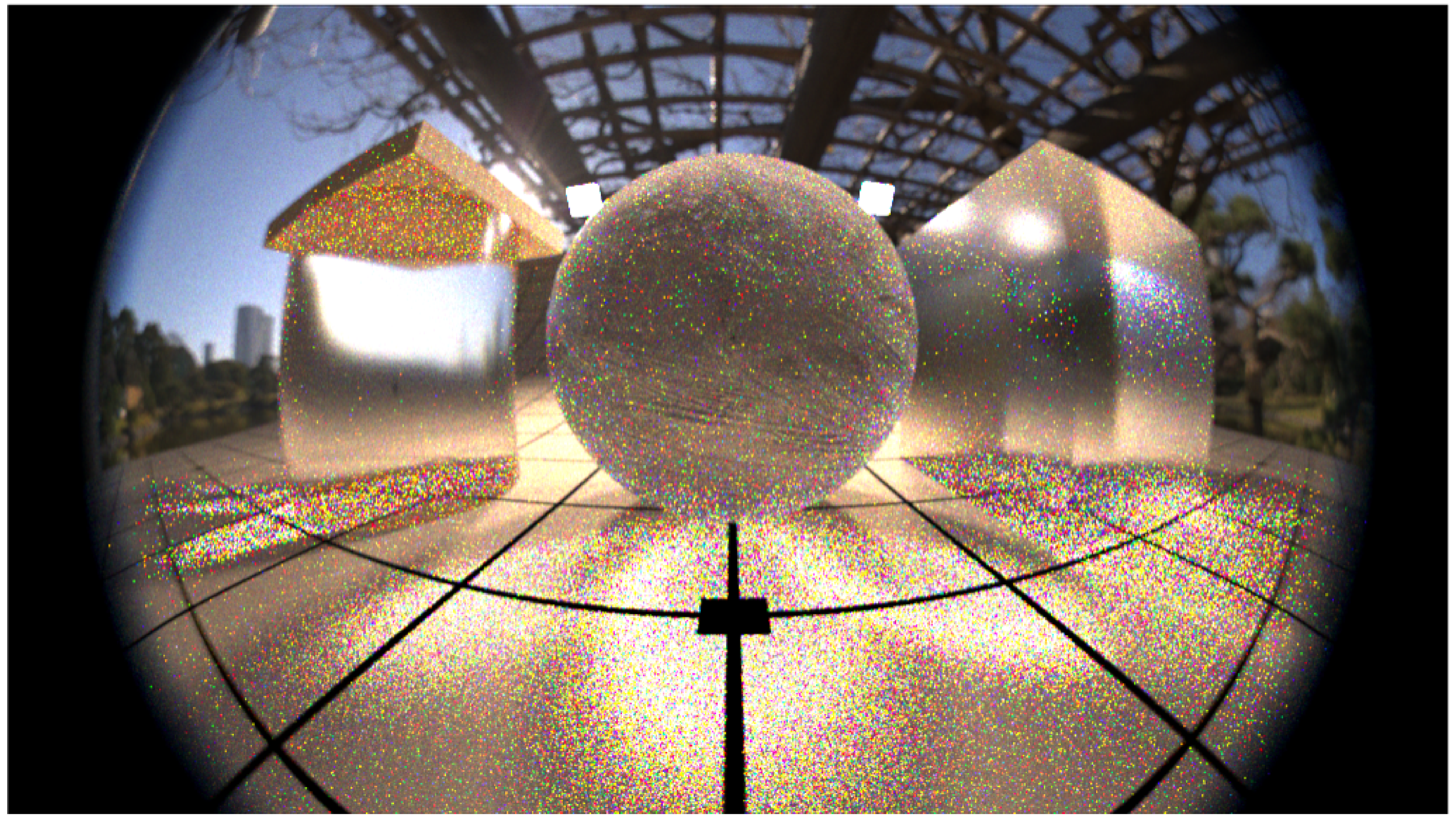


results: aperture sampling for light tracing (512spp)

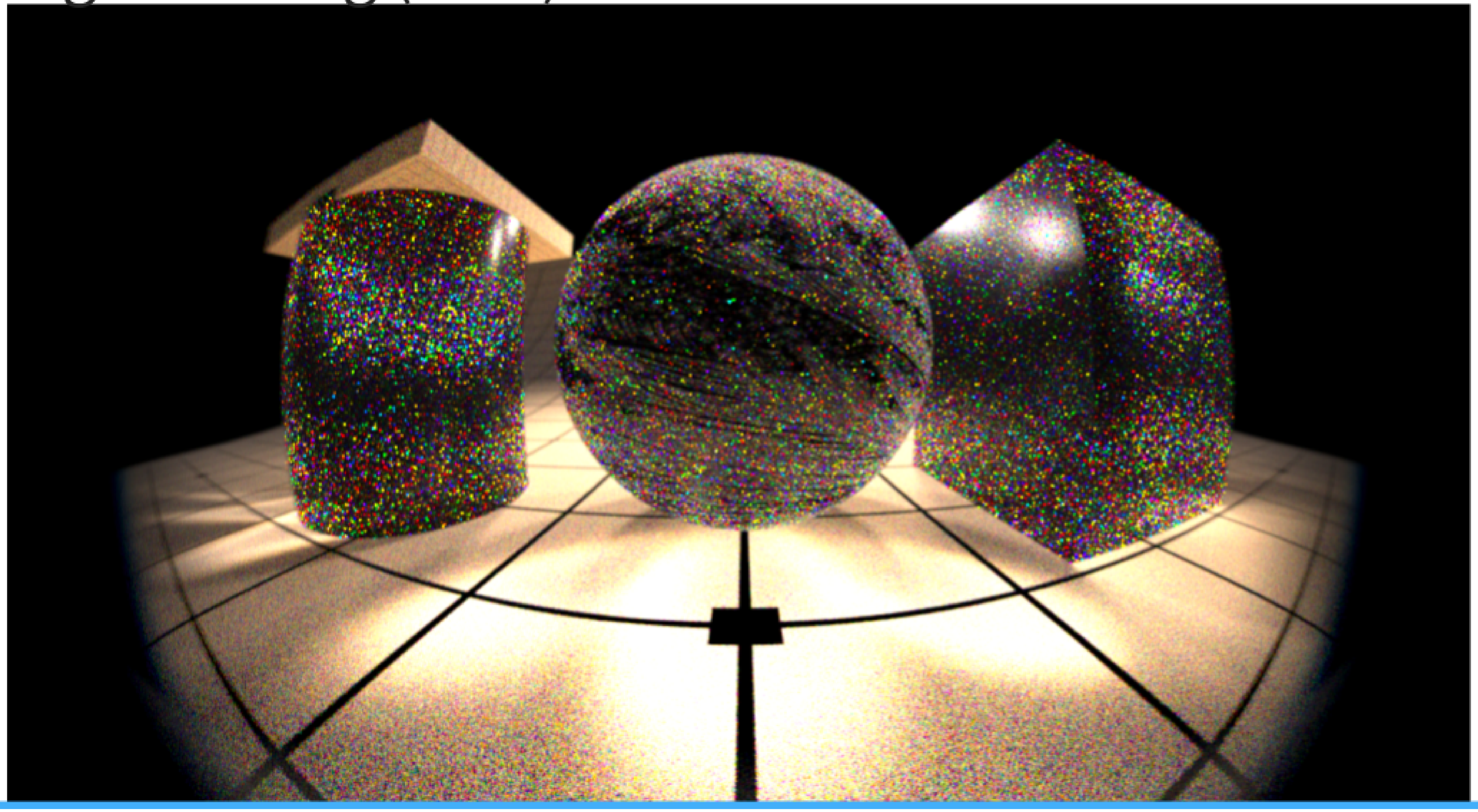
reference



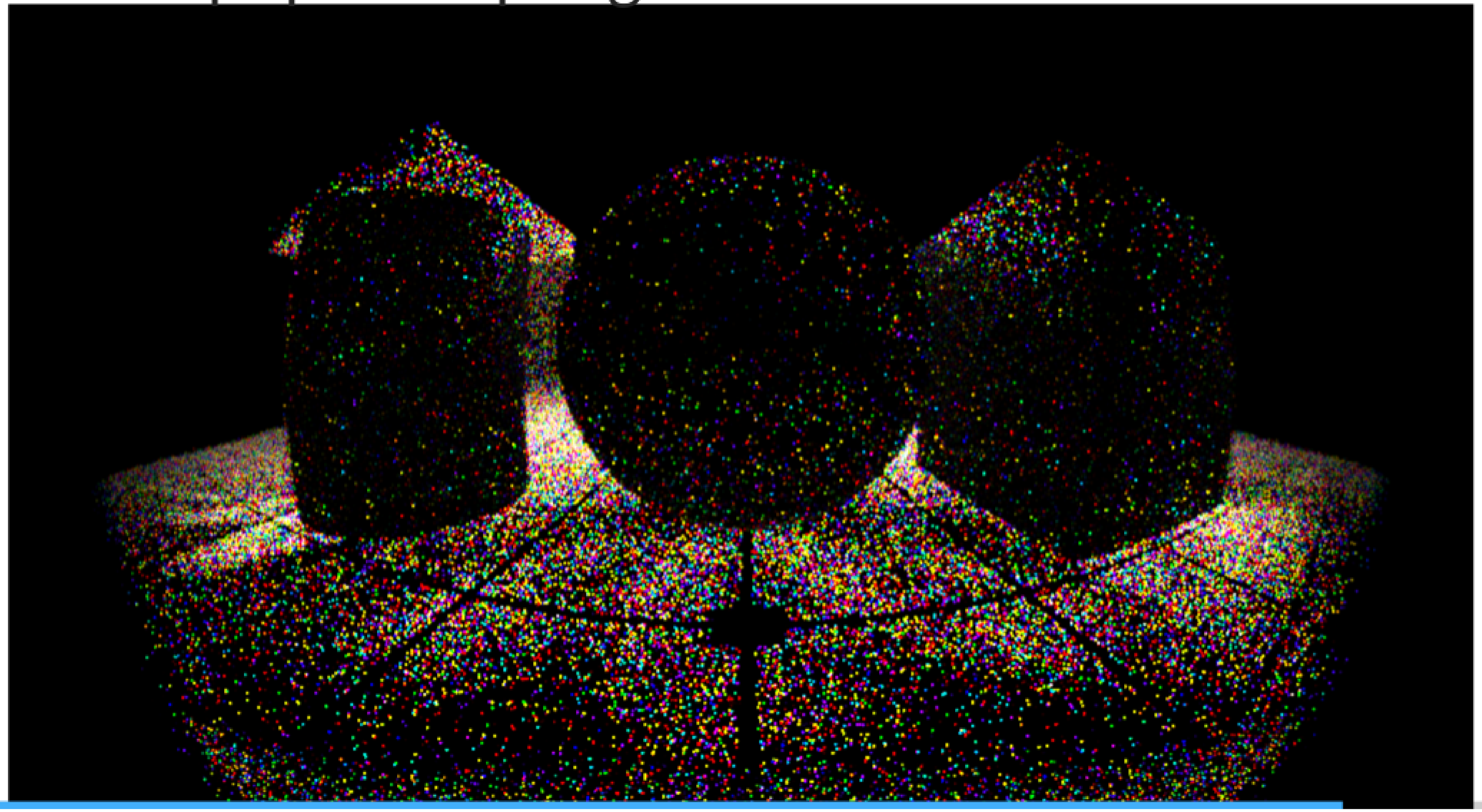
next event estimation



light tracing (ours)

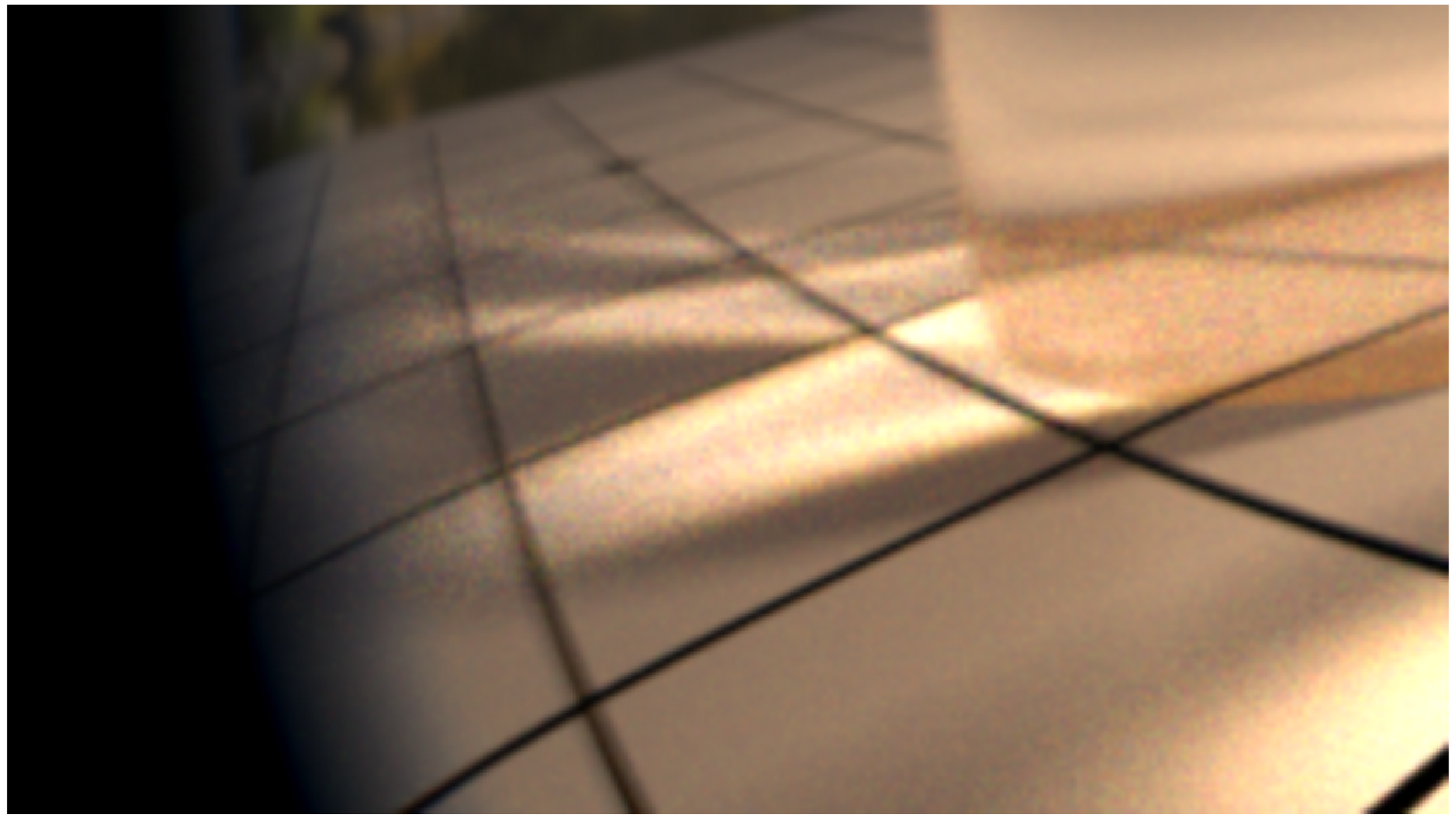


outer pupil sampling

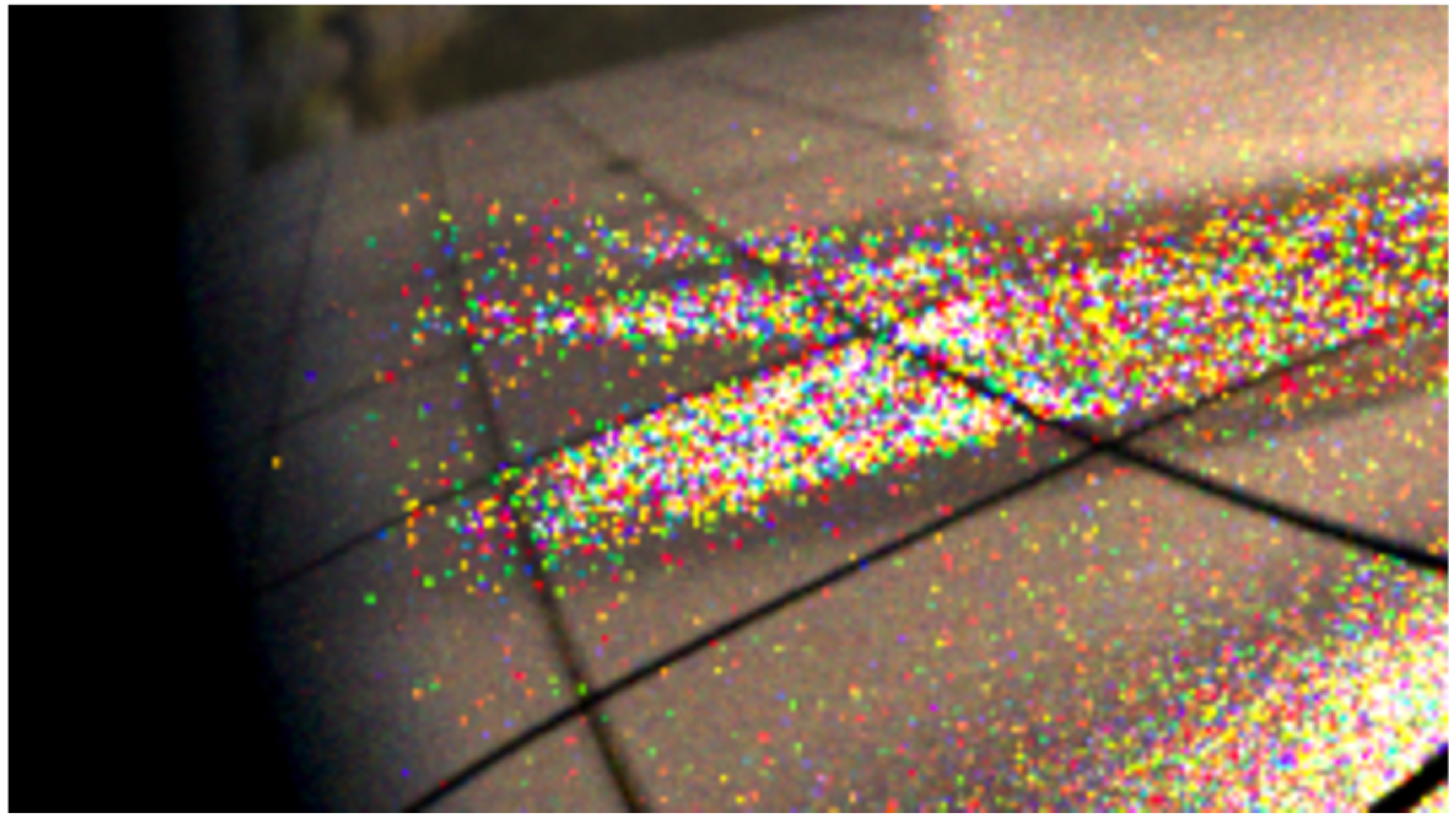


results: aperture sampling for light tracing (closeup)

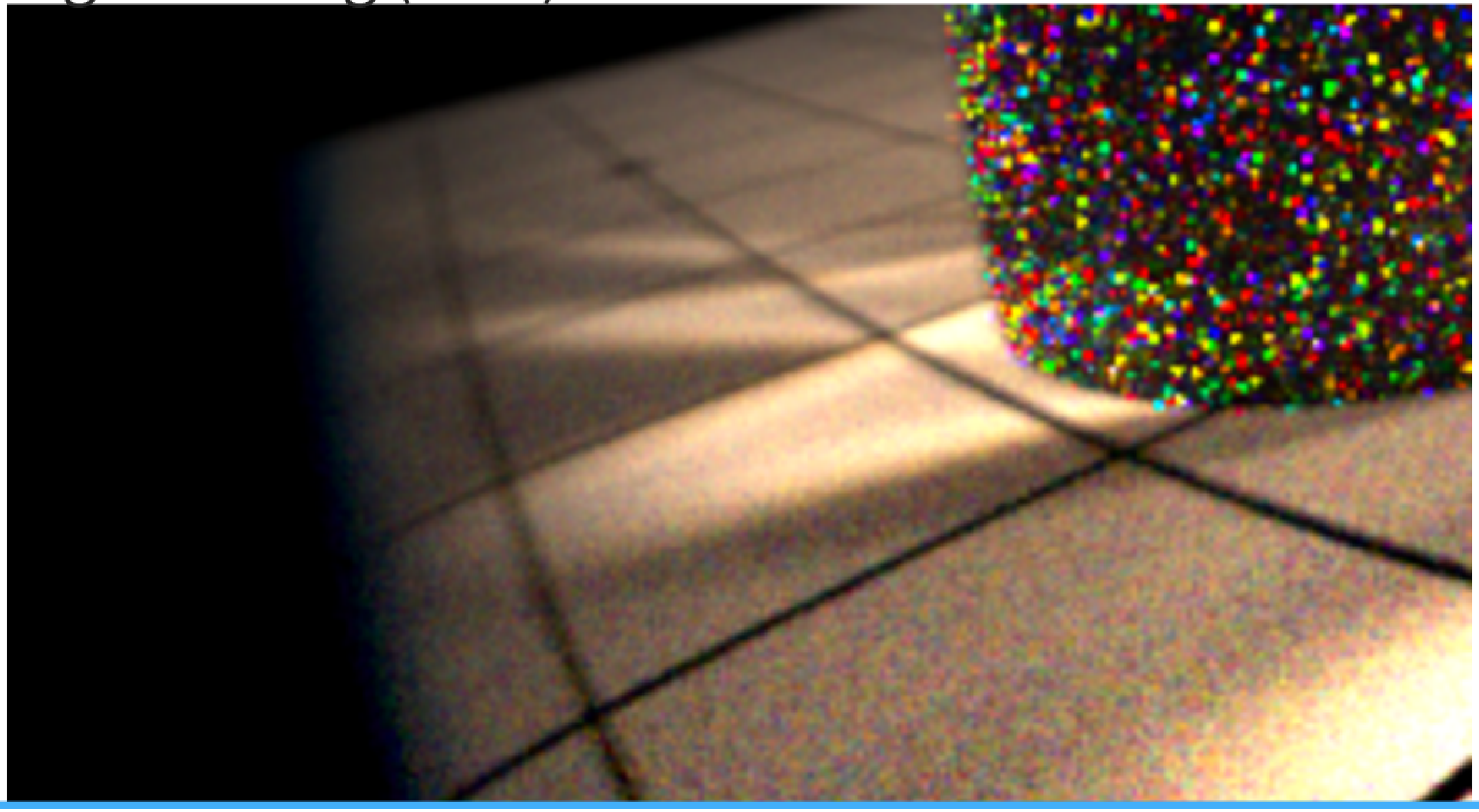
reference



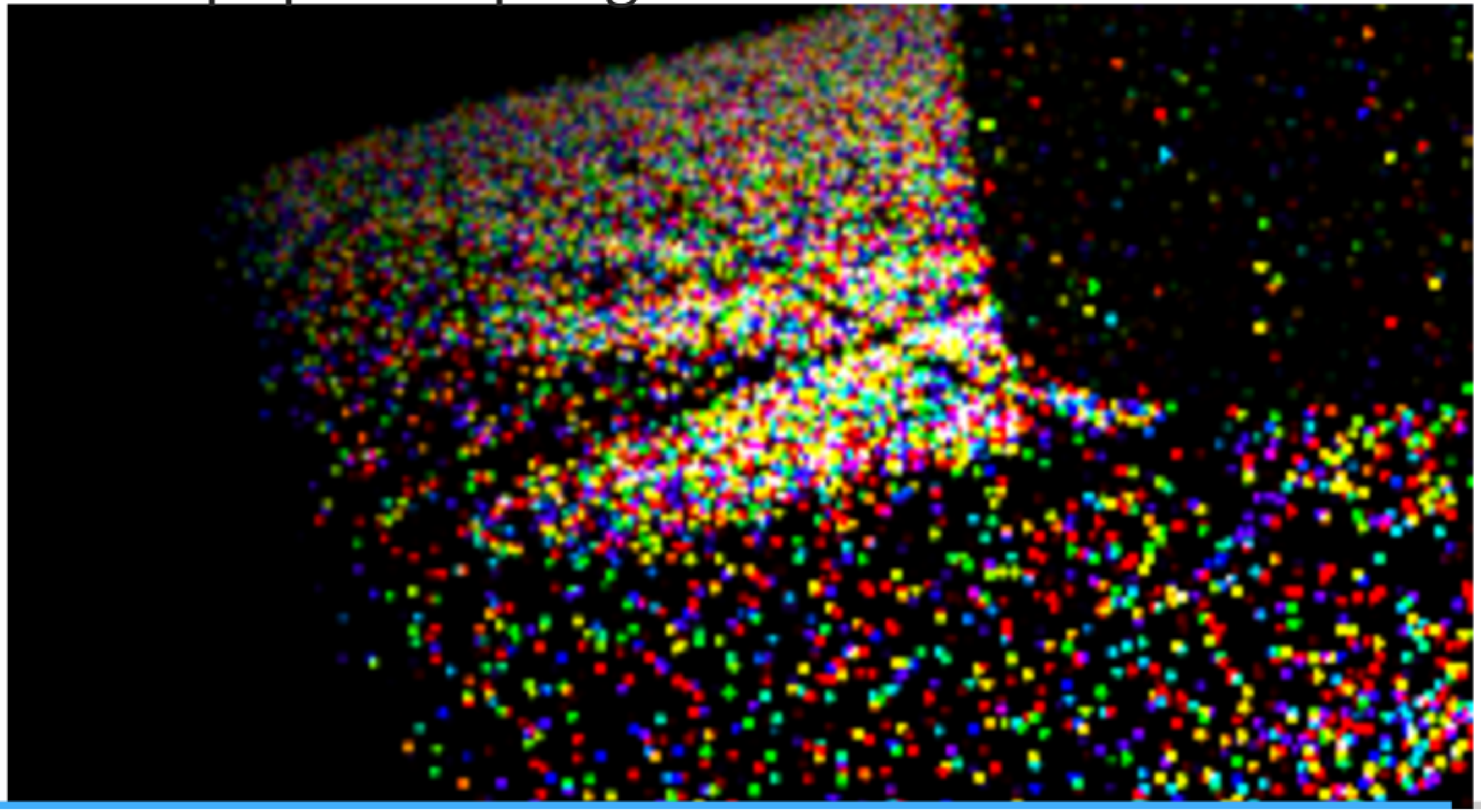
next event estimation



light tracing (ours)

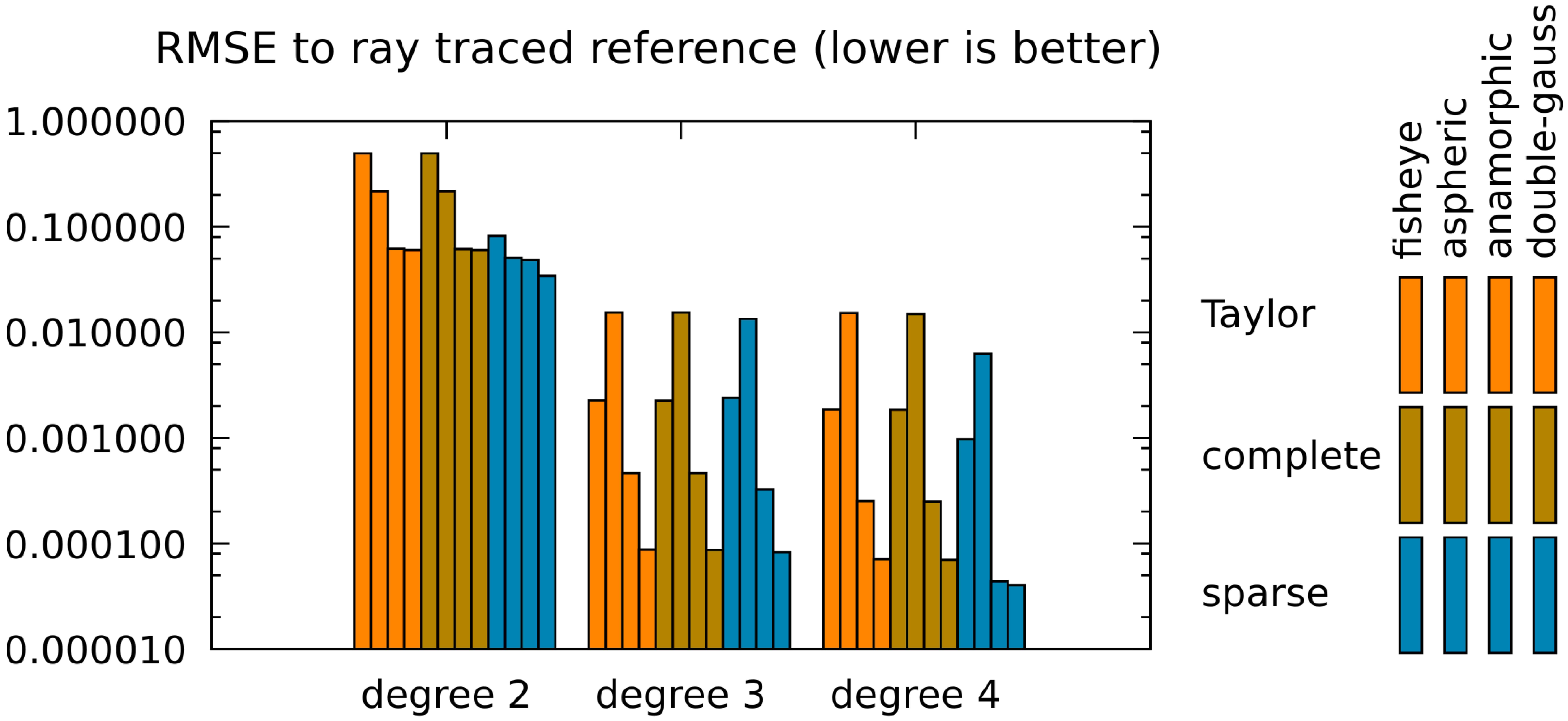


outer pupil sampling



results: accuracy of sparse polynomials

- almost always better than Taylor or full polynomials (use higher degree terms!)
  - Taylor and complete: same degree (2, 3, 4)
  - Taylor and sparse: same number of coefficients
  - analytic Taylor expansion past degree 4 becomes very hard



results: accuracy of sparse polynomials

- almost always better than Taylor or full polynomials (use higher degree terms!)
  - Taylor and complete: same degree (2, 3, 4)
  - Taylor and sparse: same number of coefficients

	coeffs	**fisheye**	**aspheric**	**anamorphic**	**double-gauss**
Taylor 2	4	4.97 · 10 −1	2.17 · 10 −1	6.18 · 10 −2	6.03 · 10 −2
Complete 2	21	4.97 · 10 −1	2.17 · 10 −1	6.17 · 10 −2	6.02 · 10 −2
Sparse	4	**8.21 · 10 −2**	**5.07 · 10 −2**	**4.85 · 10 −2**	**3.43 · 10 −2**
Taylor 3	16	2.26 · 10 −3	1.54 · 10 −2	4.63 · 10 −4	8.76 · 10 −5
Complete 3	56	**2.25 · 10 −3**	1.54 · 10 −2	4.62 · 10 −4	8.69 · 10 −5
Sparse	16	2.40 · 10 −3	**1.34 · 10 −2**	**3.25 · 10 −4**	**8.22 · 10 −5**
Taylor 4	28	1.86 · 10 −3	1.52 · 10 −2	2.52 · 10 −4	7.07 · 10 −5
Complete 4	126	1.85 · 10 −3	1.49 · 10 −2	2.50 · 10 −4	6.98 · 10 −5
Sparse	28	**9.72 · 10 −4**	**6.26 · 10 −3**	**4.40 · 10 −5**	**4.02 · 10 −5**

real time implementation

- works on deep image buffer data (here from [ZKP13])
- evaluate generated polynomial code in GLSL shader
- proof-of-concept implementation
  - 137 ms, 1080x720 px, 144 spp, AMD Radeon R9 390
  - limited by texture fetches more than by lens evaluation
- performance can probably be improved a lot by doing something smarter
  - e.g. Deferred Image-based Ray Tracing/HPG talk on Tuesday..
  - or with rasterisation (Comparison of Projection Methods for Rendering Virtual Reality)

conclusion

- more precise polynomials
  - higher degree terms, still sparse (fast)
- simpler construction
  - no Taylor expansion (which becomes untractable for higher degrees)
- now also practical for bidirectional/Metropolis
  - aperture sampling for light tracing
- proof of concept GPU implementation
- source code available

thank you for listening!

source code at https://jo.dreggn.org/home/2016_optics.tar.bz2