The Path Tracing Revolution in the Movie Industry

Experiences with path space sampling algorithms in Manuka

Johannes Hanika

Weta Digital

Outline

- rendering problems we deal with:
  - detailed digi double characters/headshots
  - big environments with many lights
  - massive (battle) scenes
- we need to render movies:
  - requirements (vs. stills)
- a look at path tracing algorithms
- conclusions

[clips]

- apes/apes2
- thb3

Problem setting

we’re rendering VFX for movies

- requirements:
  - high visual complexity
  - temporal stability
  - motion blur
  - fast turnarounds, typically finals with 100s samples per pixel, not 1000s+

- why physically-based rendering?
  - deliver true detail (e.g. interaction hair/subsurface)
  - less caches
  - this talk: technical experiences with PBR in production

Physical foundation: the rendering equation

measurement contribution and path space

- recursive rendering equation: emission + transport

$$L = L_e + \boldsymbol{T}L$$

- Monte Carlo integration in path space

$$I_j = \int_{\boldsymbol{\mathcal{P}}} f(\boldsymbol{X})\, d\boldsymbol{X} \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(\boldsymbol{X}_i)}{p(\boldsymbol{X}_i)}$$

- path $\boldsymbol{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k) \in \boldsymbol{\mathcal{P}}$ list of vertices $\mathbf{x}$
- $p(\boldsymbol{X}_i)$ in product area measure (convert if not)
- create these paths via (bidirectional) path tracing, photon mapping, etc. ..

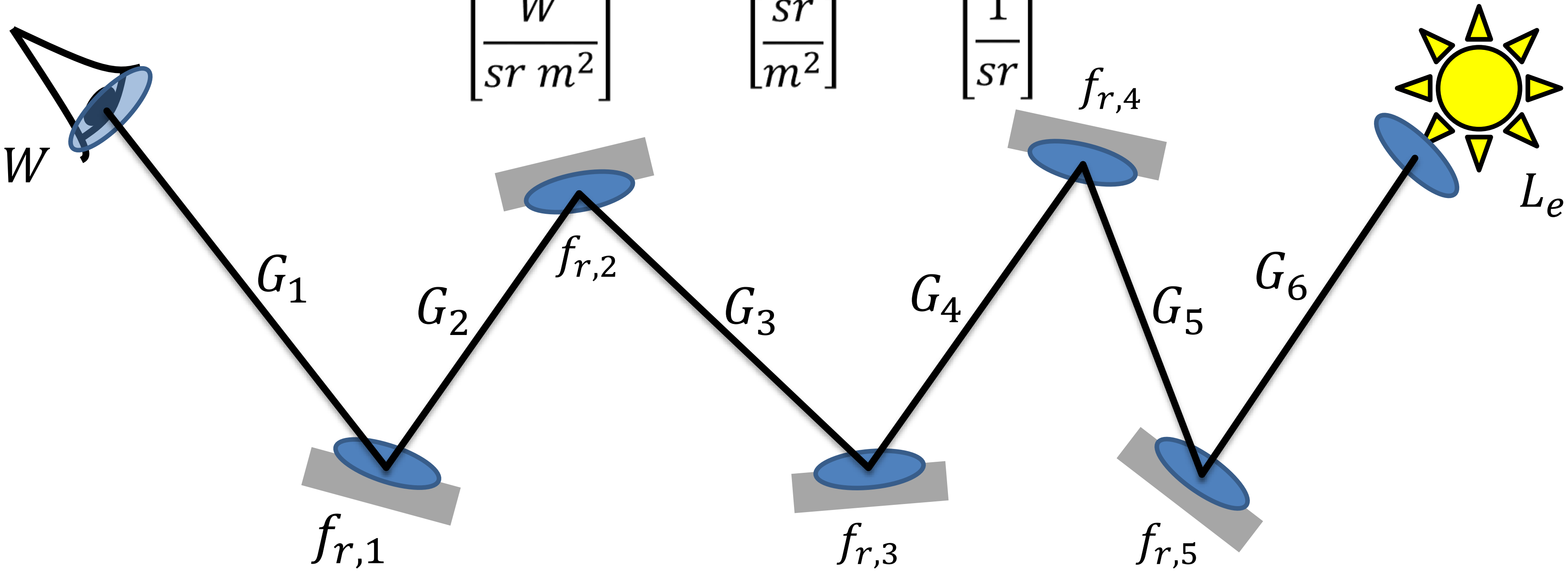
Measurement contribution function

- measurement contribution f for path $\boldsymbol{X}$ with length k (=7 here)

$$f(\boldsymbol{X}) = \boxed{L_e}\boxed{G_{k-1}}\left(\prod_{i=1}^{k-2}\boxed{f_{r,i}}\boxed{G_i}\right)W$$

unitless

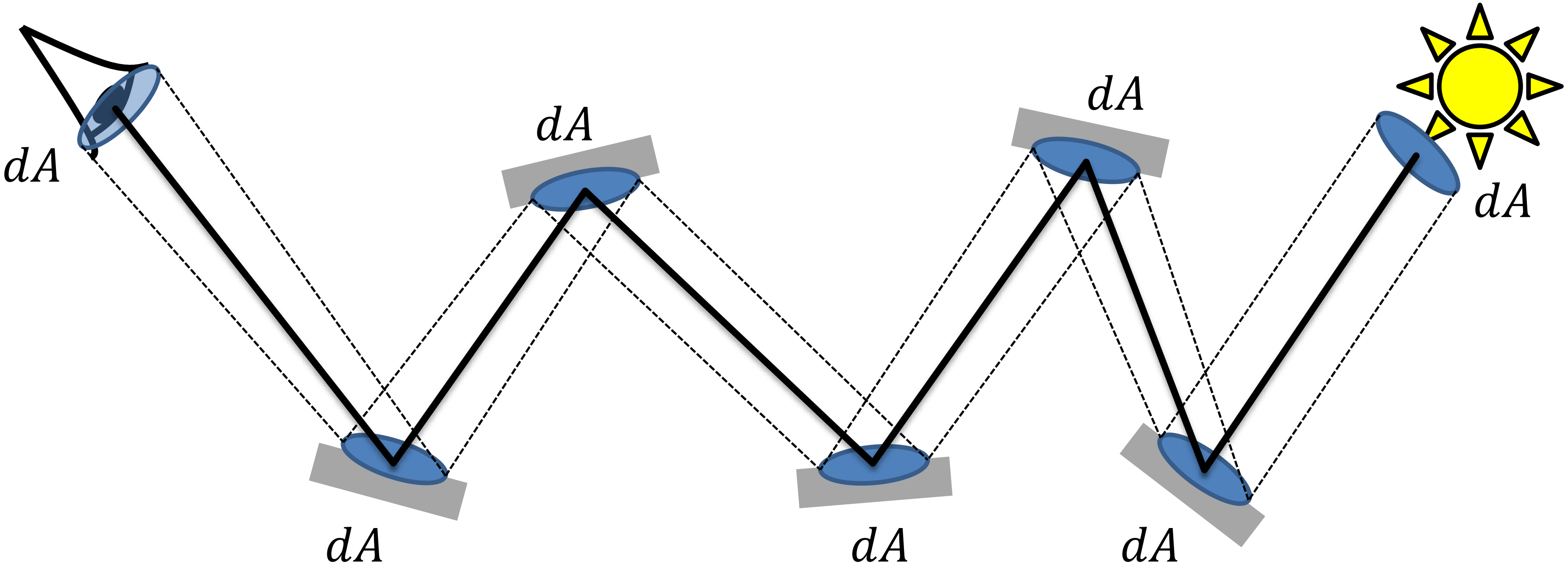
$$\left[\frac{W}{sr\ m^2}\right] \qquad \left[\frac{sr}{m^2}\right] \qquad \left[\frac{1}{sr}\right]$$



Measurement contribution function

- intuition: flux Φ through all differential areas dA_i of a path $\boldsymbol{X}$

$$f(\boldsymbol{X}) = \prod_{i=1}^{k} \frac{d\Phi}{dA_i} = \frac{d\Phi}{d\boldsymbol{X}_k} \qquad [W/(m^2)^k]$$



Available tools to create paths

path tracing algorithms

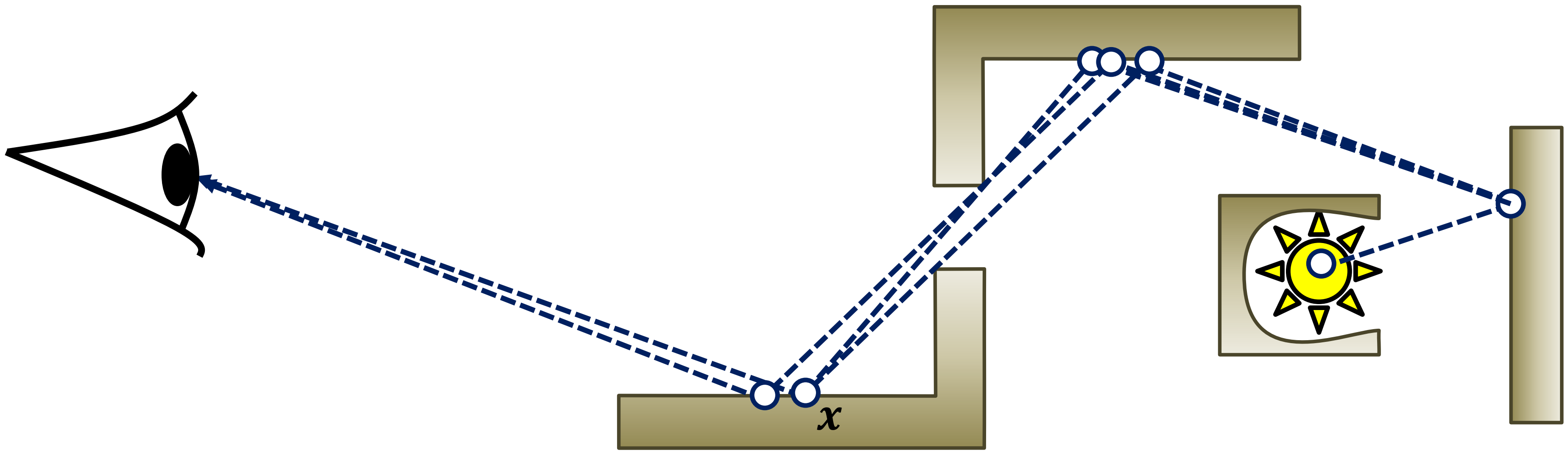
- roughly ordered from simple to more advanced:
  - path tracing, next event estimation
  - light tracing
  - bidirectional path tracing with multiple importance sampling
  - photon mapping
  - Metropolis light transport/Markov chain Monte Carlo

Choice of algorithm

- we have complicated problems
- so we start with the most complicated algorithm in this talk!
  - (it should be the best fit, right?)
  - quick summary how it works
  - short practical evaluation

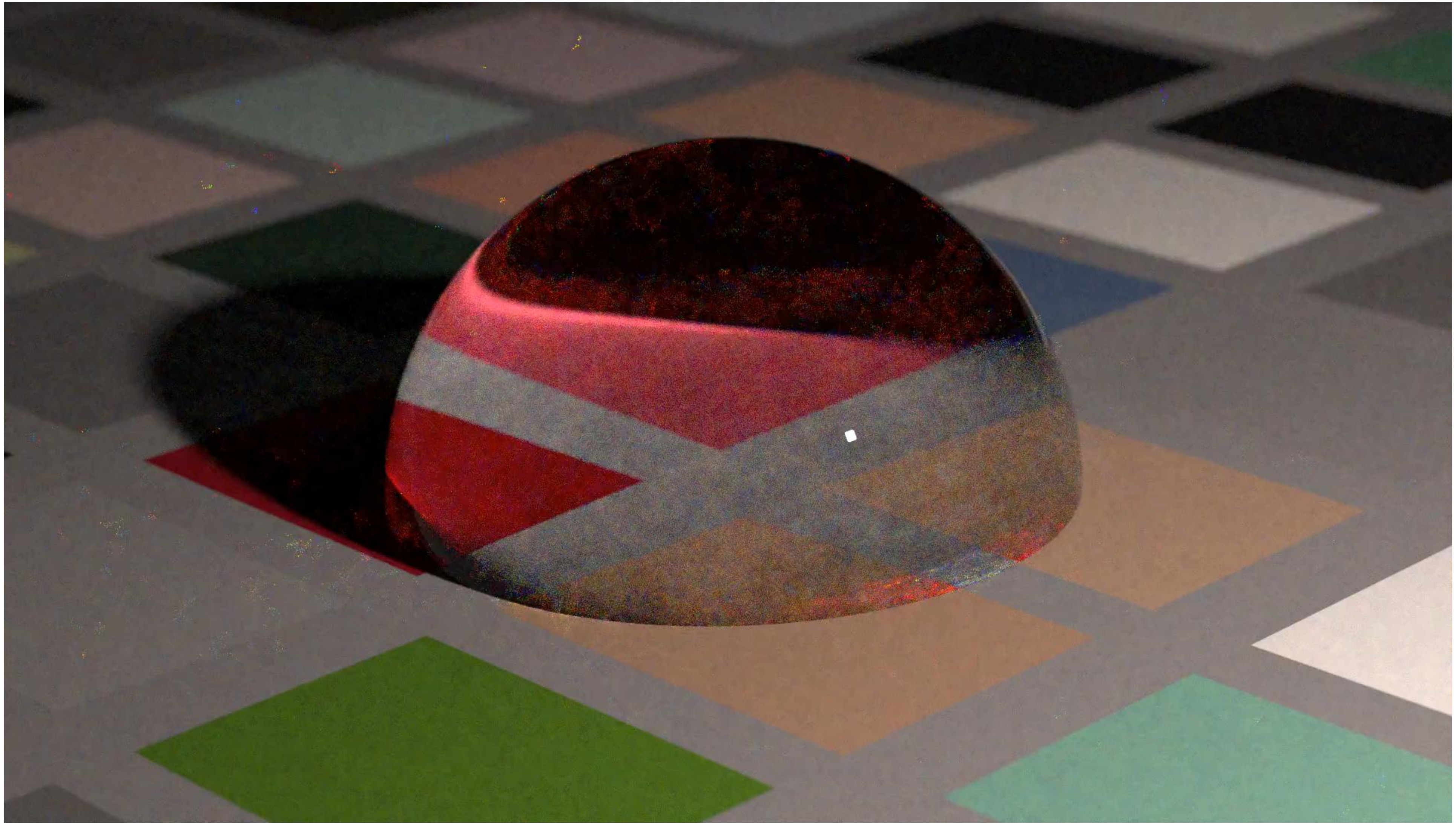
Metropolis Light Transport

- [Veach 1997, Kelemen 2002]
  - Markov chain Monte Carlo (MCMC) method
  - find an interesting path (high measurement contribution)
  - mutate to explore similar paths



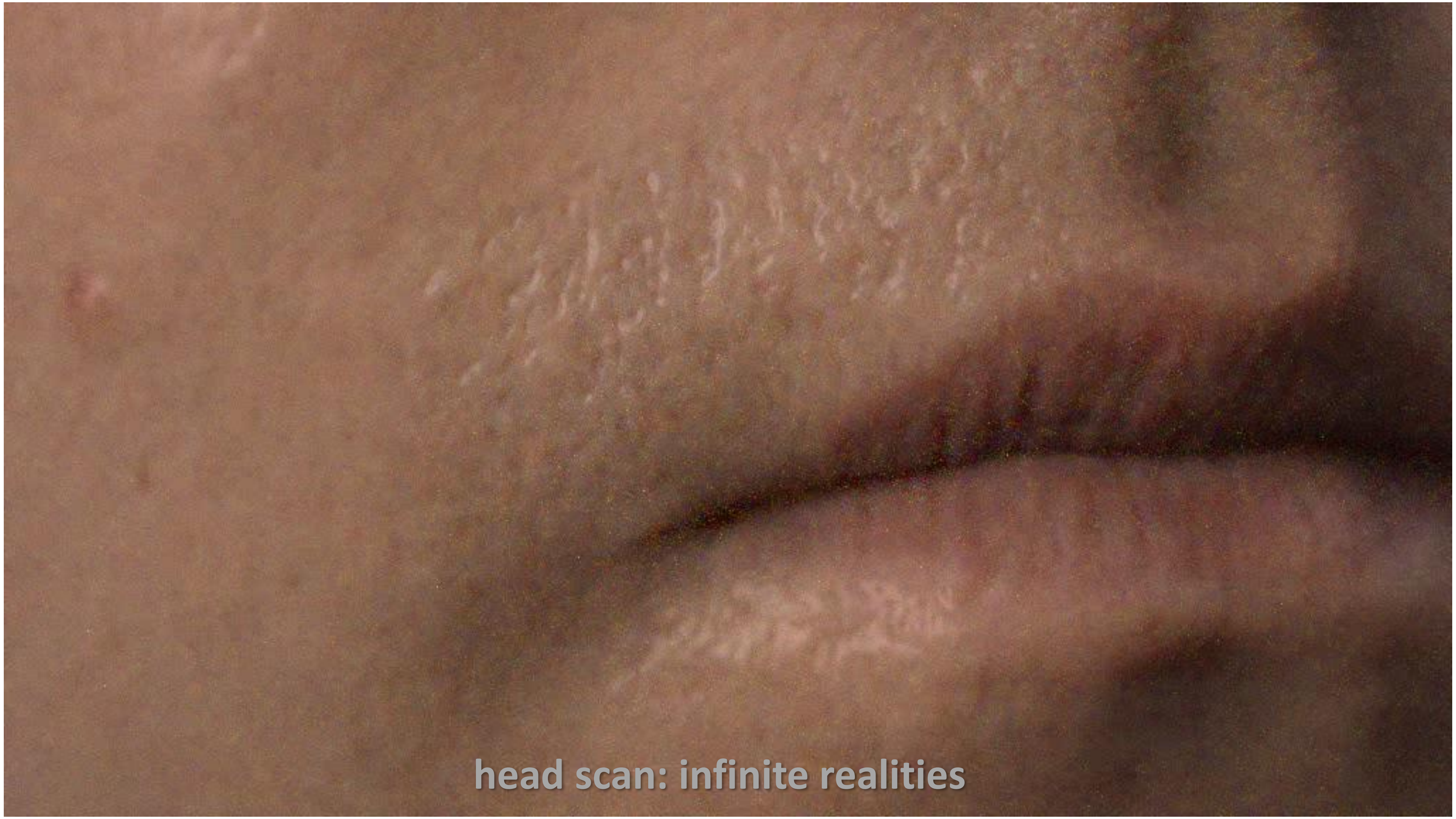
Metropolis light transport (Kelemen style)

- 3min/frame render, no motion blur
- temporal instability/autocorrelation due to random walk behaviour!



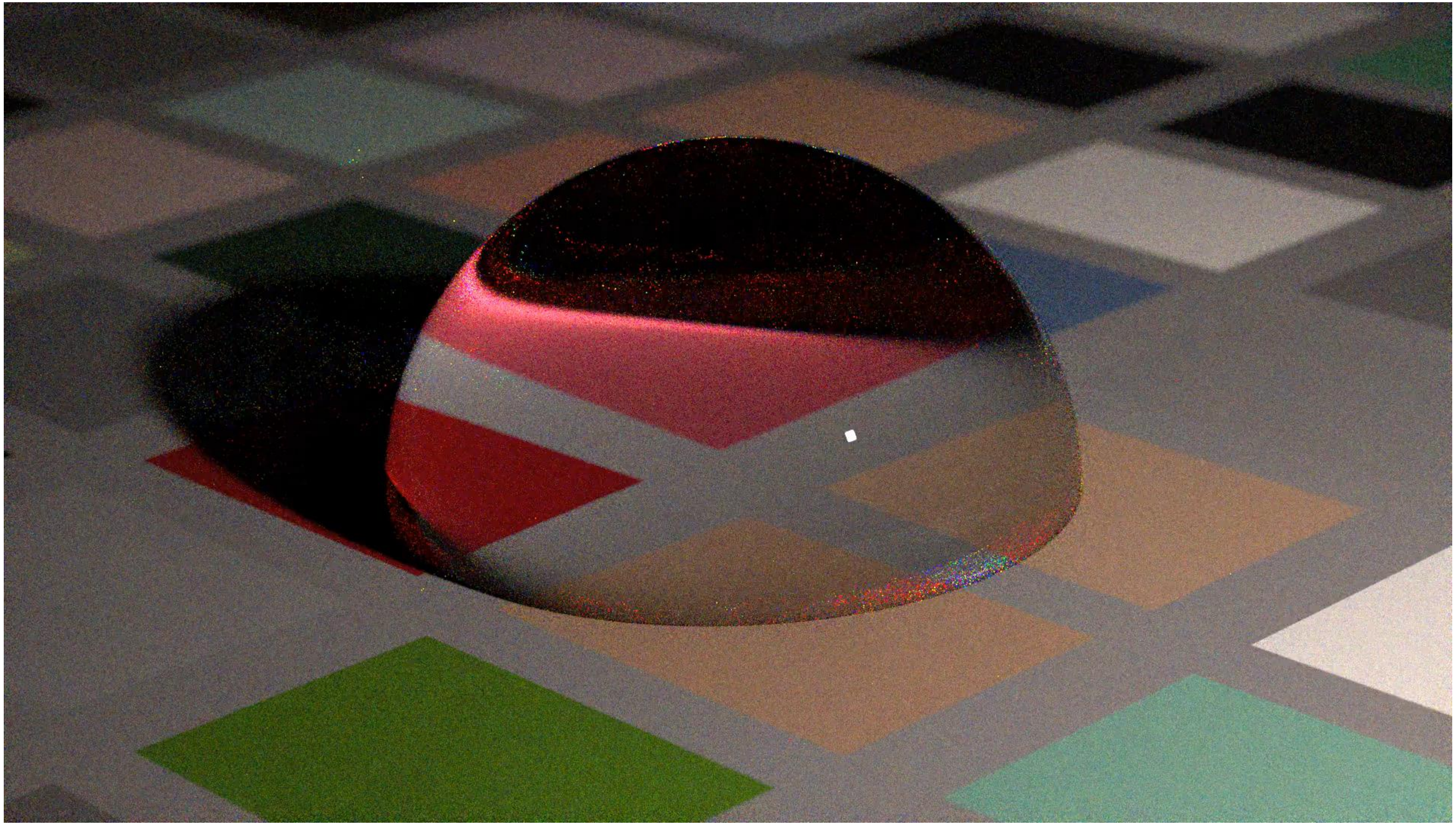
Metropolis light transport

- 1k spp Kelemen Metropolis, still flickering



MLT: half vector space light transport

- stratification by explicit ray differentials
- problems discovering effects (path configuration and sub-spaces)!



MLT: half vector space light transport

- 64spp, relatively lightweight 5M polygons but strongly varying geometric derivatives

reference
20h

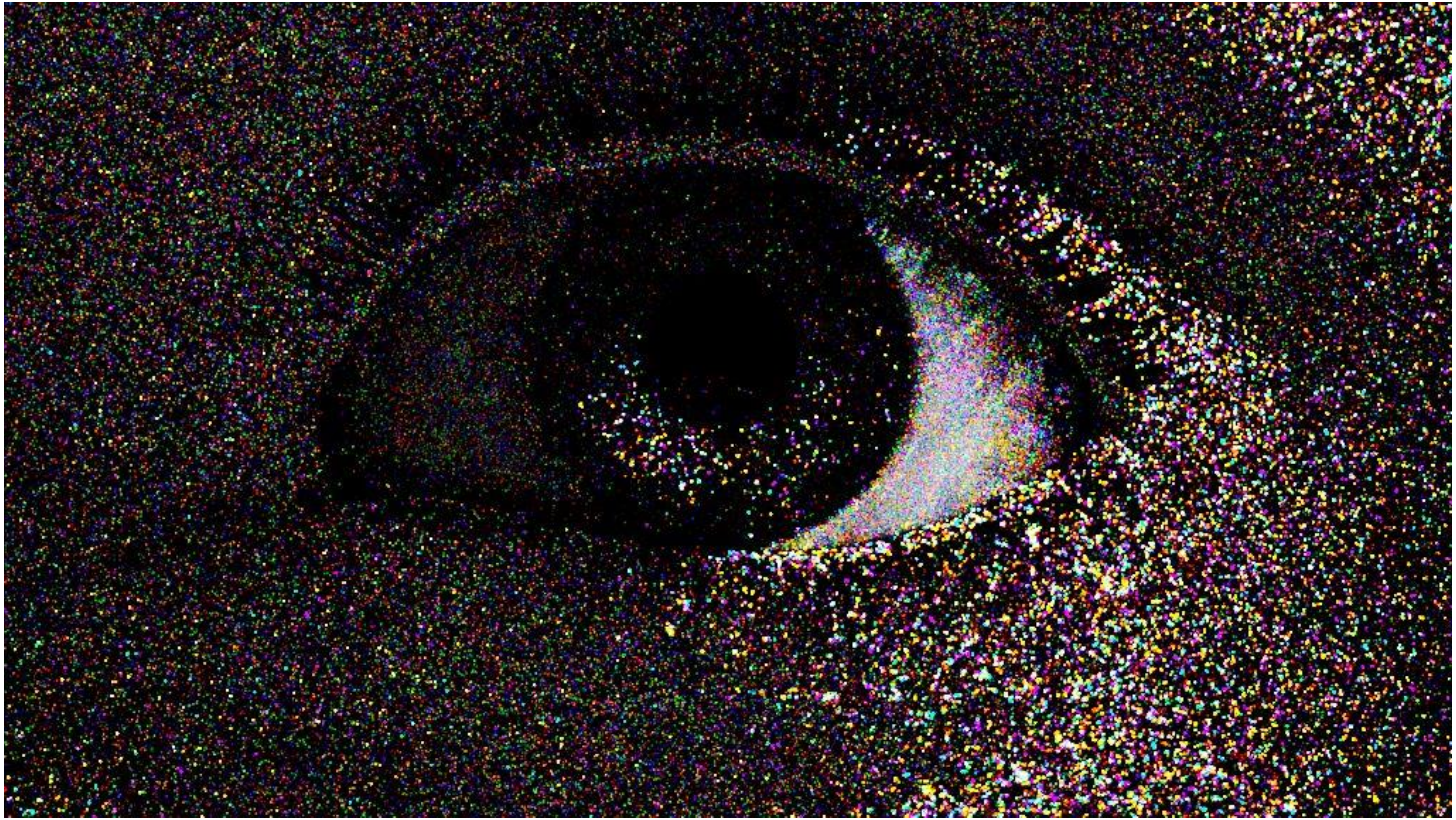
HSLT
113s

HSLT 157s
no displacement

improved
HSLT 54s

MLT: half vector space light transport

- 64spp, relatively lightweight 5M polygons but strongly varying geometric derivatives



reference
20h

HSLT
113s

HSLT 157s
no displacement

improved
HSLT 54s

MLT: half vector space light transport

- 64spp, relatively lightweight 5M polygons but strongly varying geometric derivatives

reference
20h

HSLT
113s

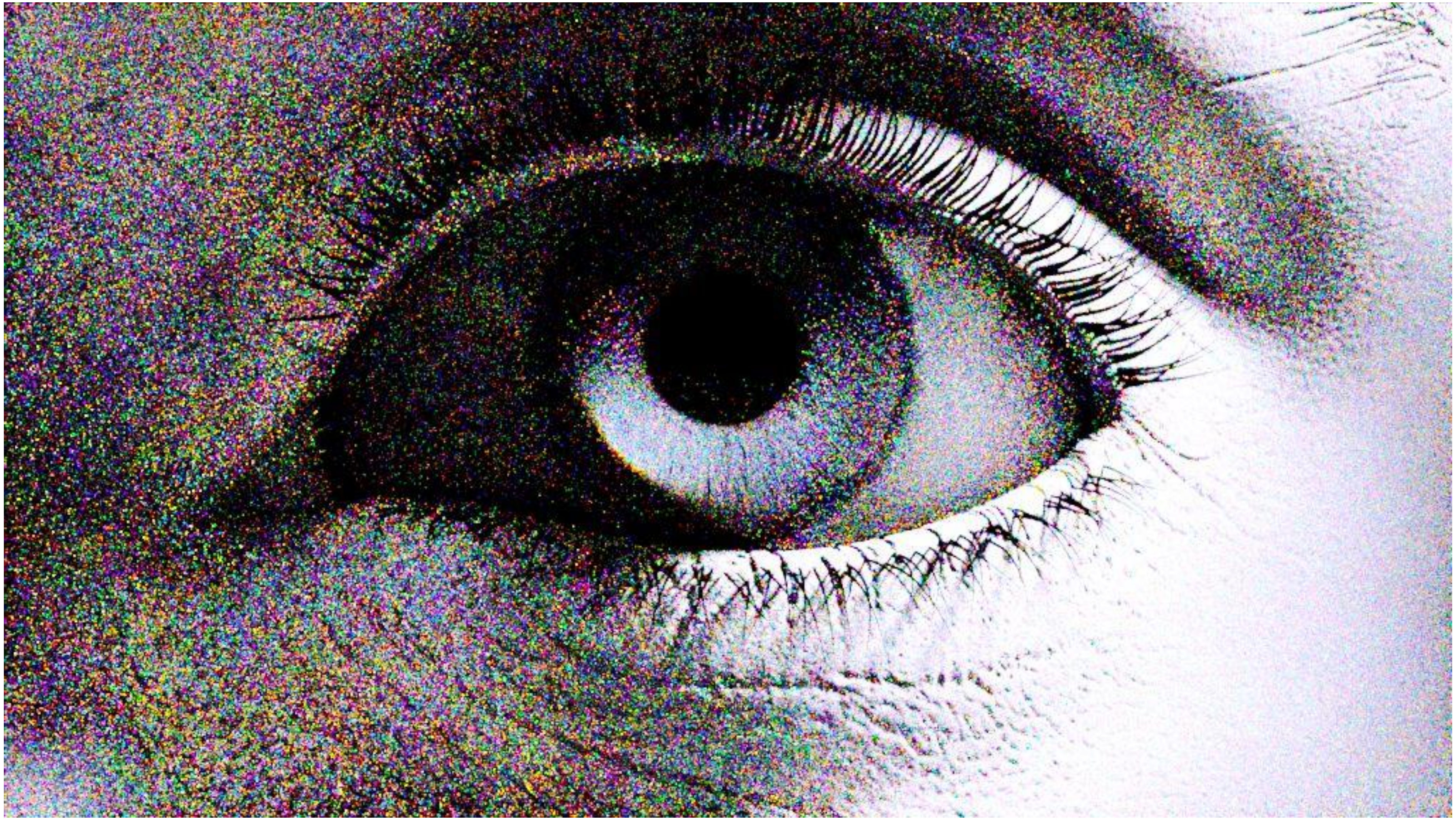
HSLT 157s
no displacement

improved
HSLT 54s

17

MLT: half vector space light transport

- 64spp, relatively lightweight 5M polygons but strongly varying geometric derivatives



reference
20h

HSLT
113s

HSLT 157s
no displacement

improved
HSLT 54s

MLT: half vector space light transport

- still has practical drawbacks
  - only starting to be extended for complex geometry
  - no extension to participating media/skin yet
  - MCMC helps **exploring** modes, not **discovering** them

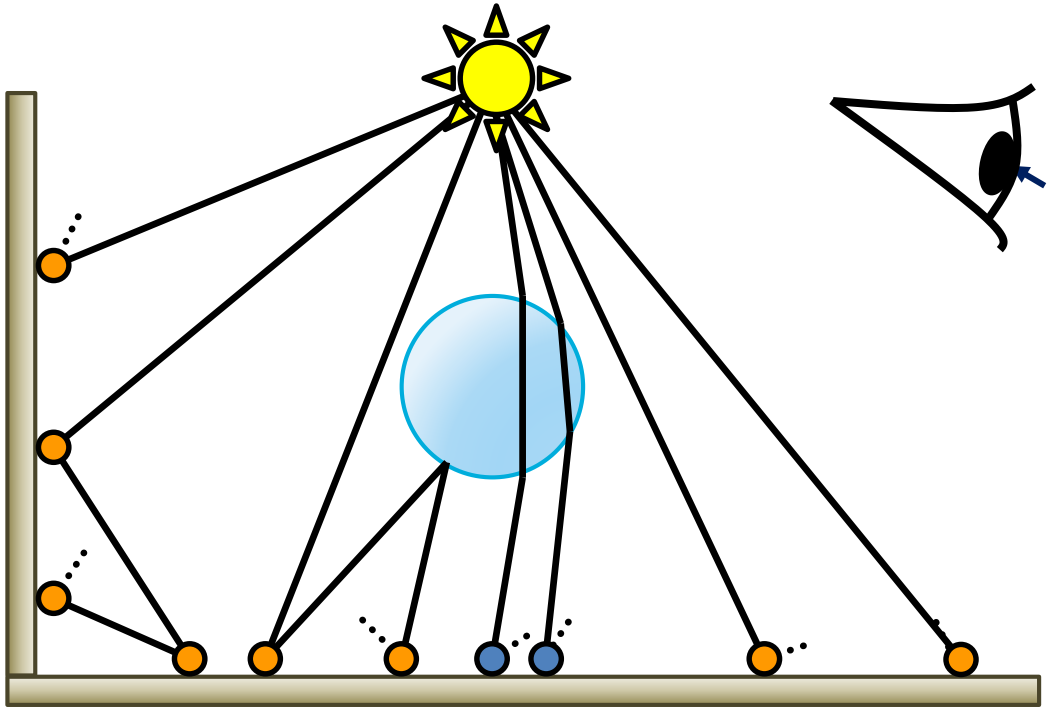
- variant for better discovery: **Energy Redistribution Path Tracing (ERPT)**
  - path tracer (discover) with short bursts of Markov chains (explore)
  - surprisingly viable in our field, path tracing is a good starting point!

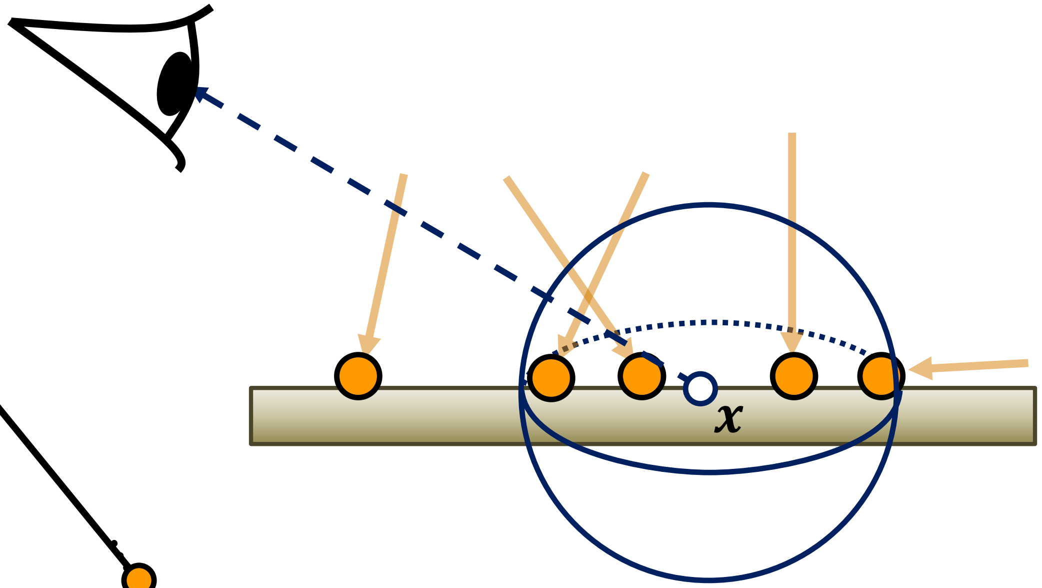
  - parameters **hard to control** (path expression to run on, #chains, #mutations)
    - affect run time/image quality in non trivial ways
    - **not practical** at this point

Photon Mapping

two step algorithm

- launch photons from the light sources
- trace and store on non-specular surfaces
- estimate irradiance by #photons/area
- relatively robust, small bias in practice (will not reduce radius below pixel footprint)





Photon mapping/VCM

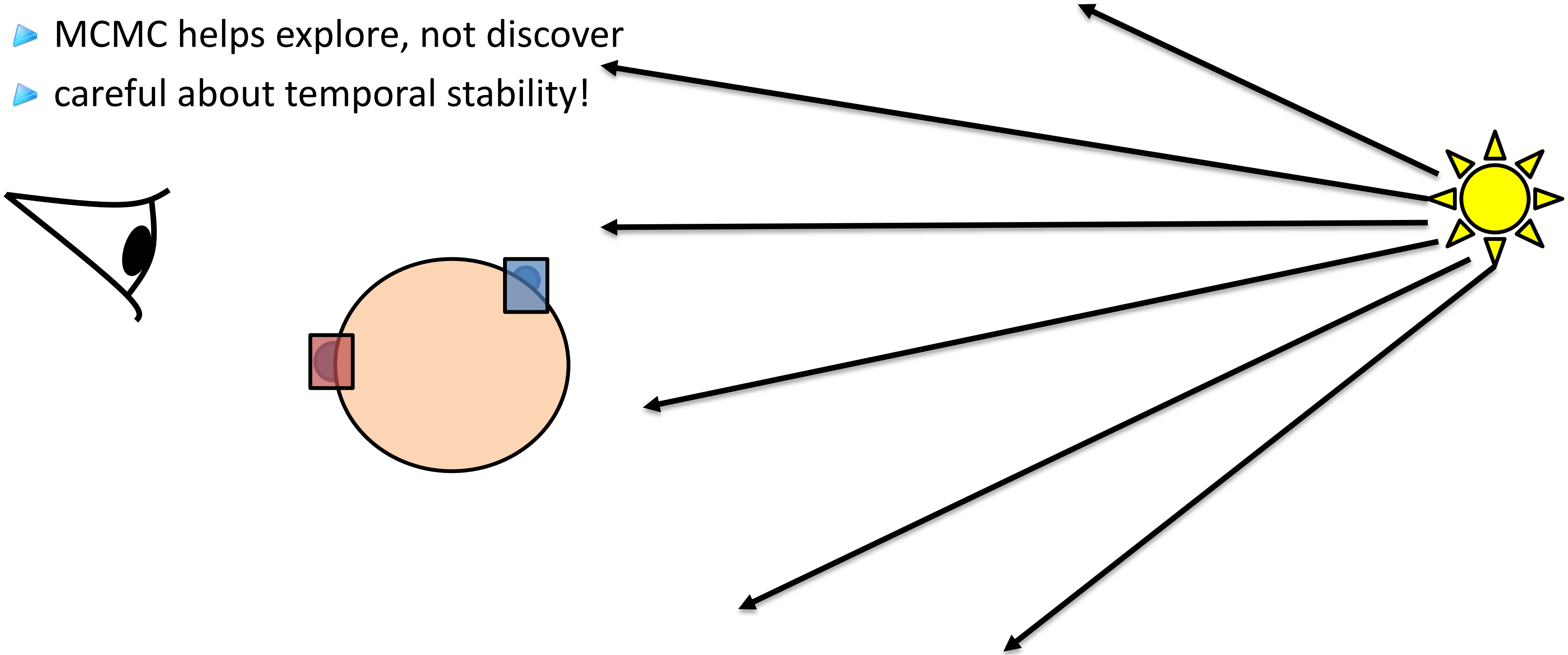
- Vertex Connection and Merging (shoots photons to bounding box of head)
  - still inefficient at finding small caustics
  - high storage and kd-tree building overhead
  - photon paths in hair: expensive tracing, no contribution

PT/NEE 8spp

VCM 8spp

Photon/Light tracing density

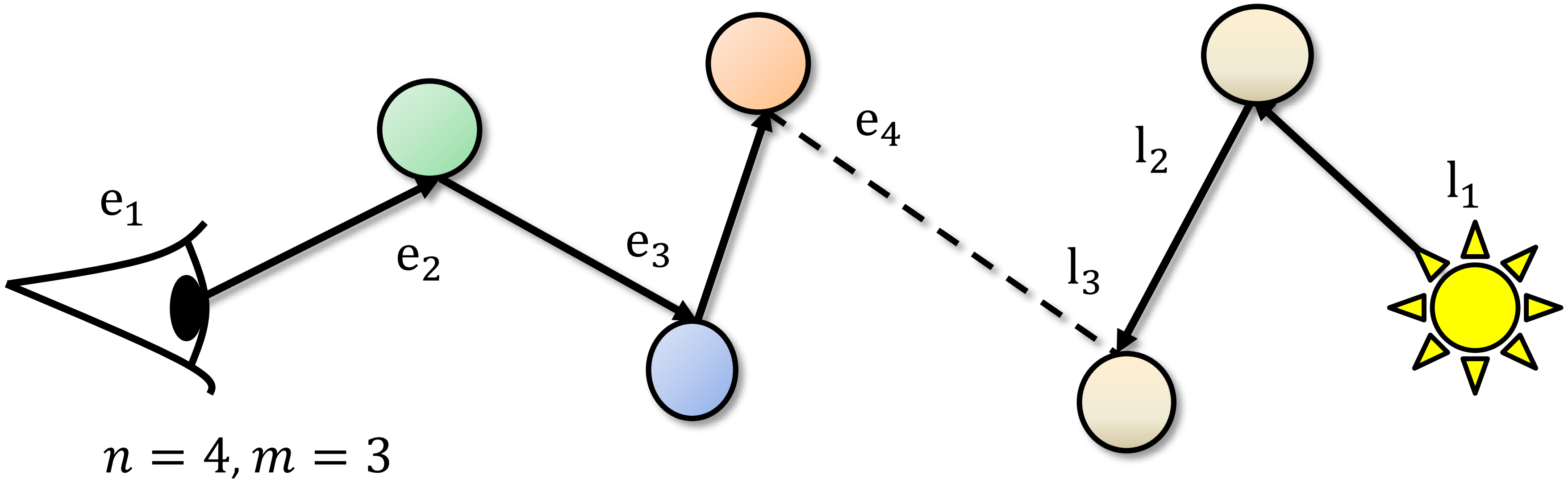
- does not follow importance from the camera
- hard to aim for objects that are expected to cast caustics
- photon attractors? occlusion?
- Markov chain photon maps [Wald 1999, Hachisuka et al. 2014]?
  - MCMC helps explore, not discover
  - careful about temporal stability!



Bidirectional path tracing

- trace from the eye and from the lights
  [Lafortune & Willems 1993, Veach & Guibas 1994])

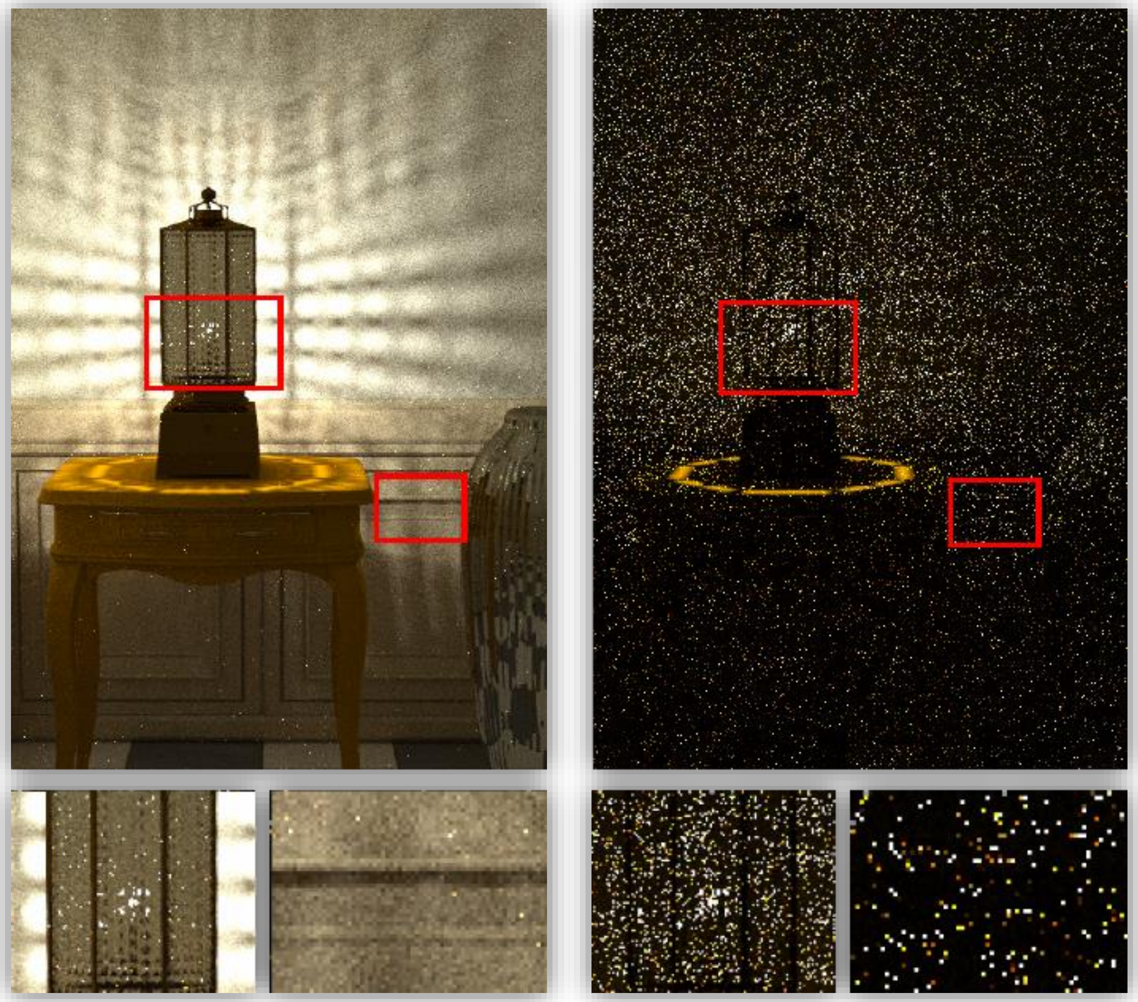
- eye path vertices $\mathrm{e_i}$ (**…E**) and light path vertices $\mathrm{l_i}$ (**L…**)



total number of vertices $k = n + m$

Bidirectional path tracing

- essential in some cases!



BDPT

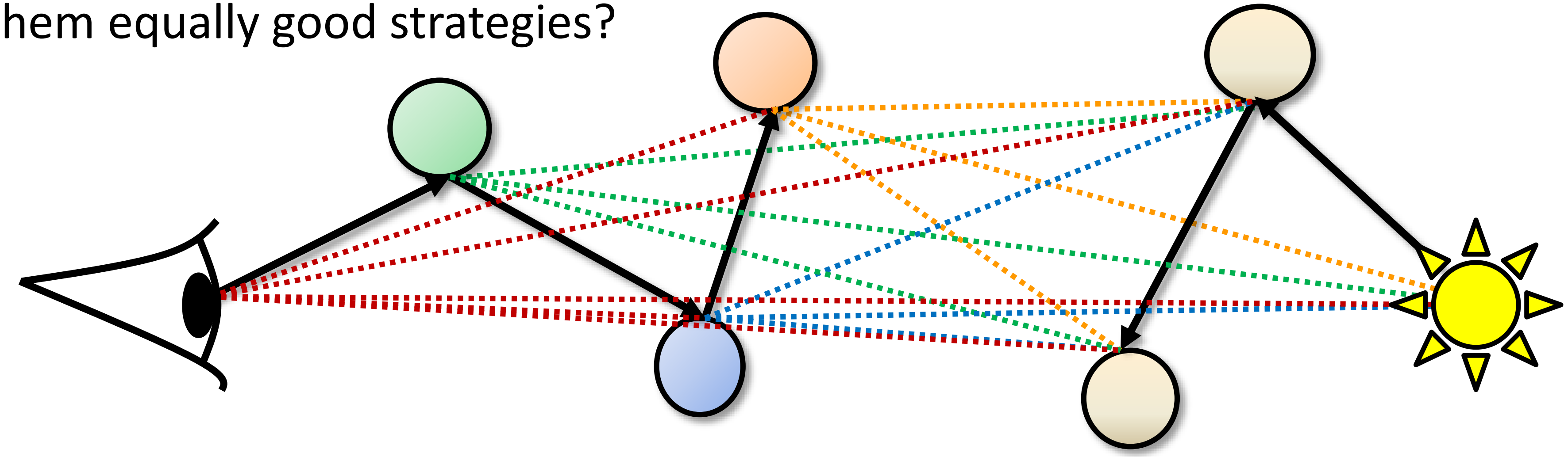
path tracing

images:
Toshiya Hachisuka

Bidirectional path tracing

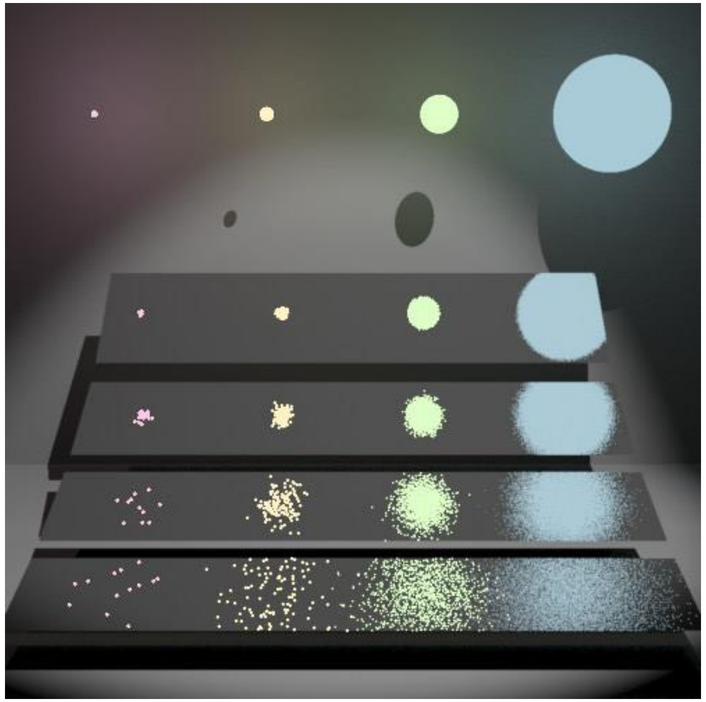
- trace from the eye and from the lights
  [Lafortune & Willems 1993, Veach & Guibas 1994])

- eye path vertices $\mathrm{e_i}$ (**…E**) and light path vertices $\mathrm{l_i}$ (**L…**)
  - and all connections!
  - all of them equally good strategies?

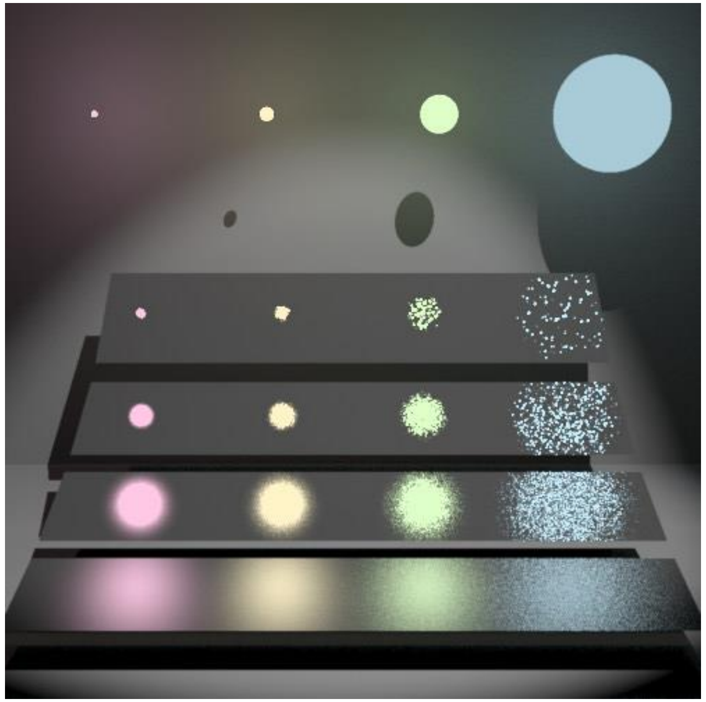


Multiple importance sampling

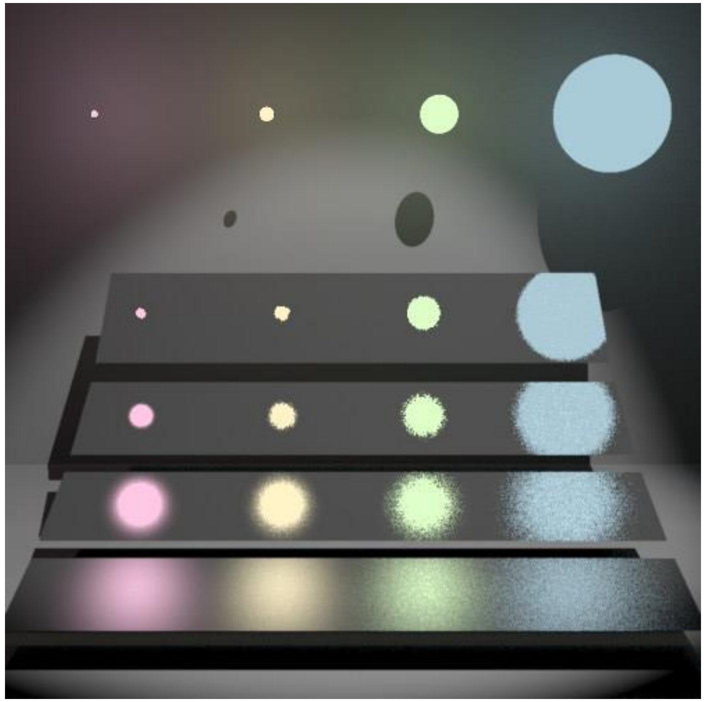
- sample multiple (imperfect) distributions (BSDF and light sources here)
  - optimal combination reduces noise
  - does not replace optimal (product) sampling strategy!



BSDF sampling



light source sampling



multiple importance sampling

images: Eric Veach

Bidirectional path tracing

path pyramid: contributions from individual techniques

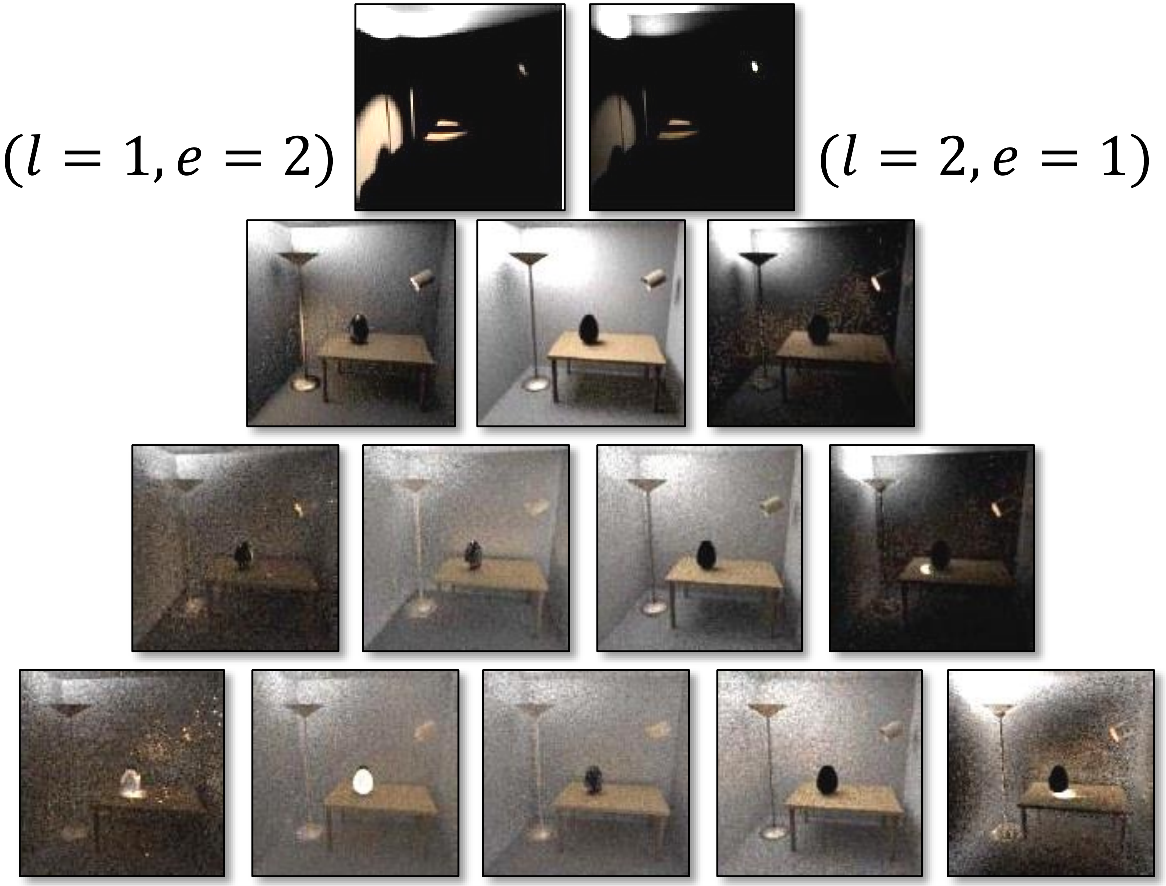
- inner connections have a big contribution (in this scene)

$k = 3$ $(l = 1, e = 2)$ $(l = 2, e = 1)$

$k = 4$

$k = 5$

$k = 6$ $(l = 5, e = 1)$



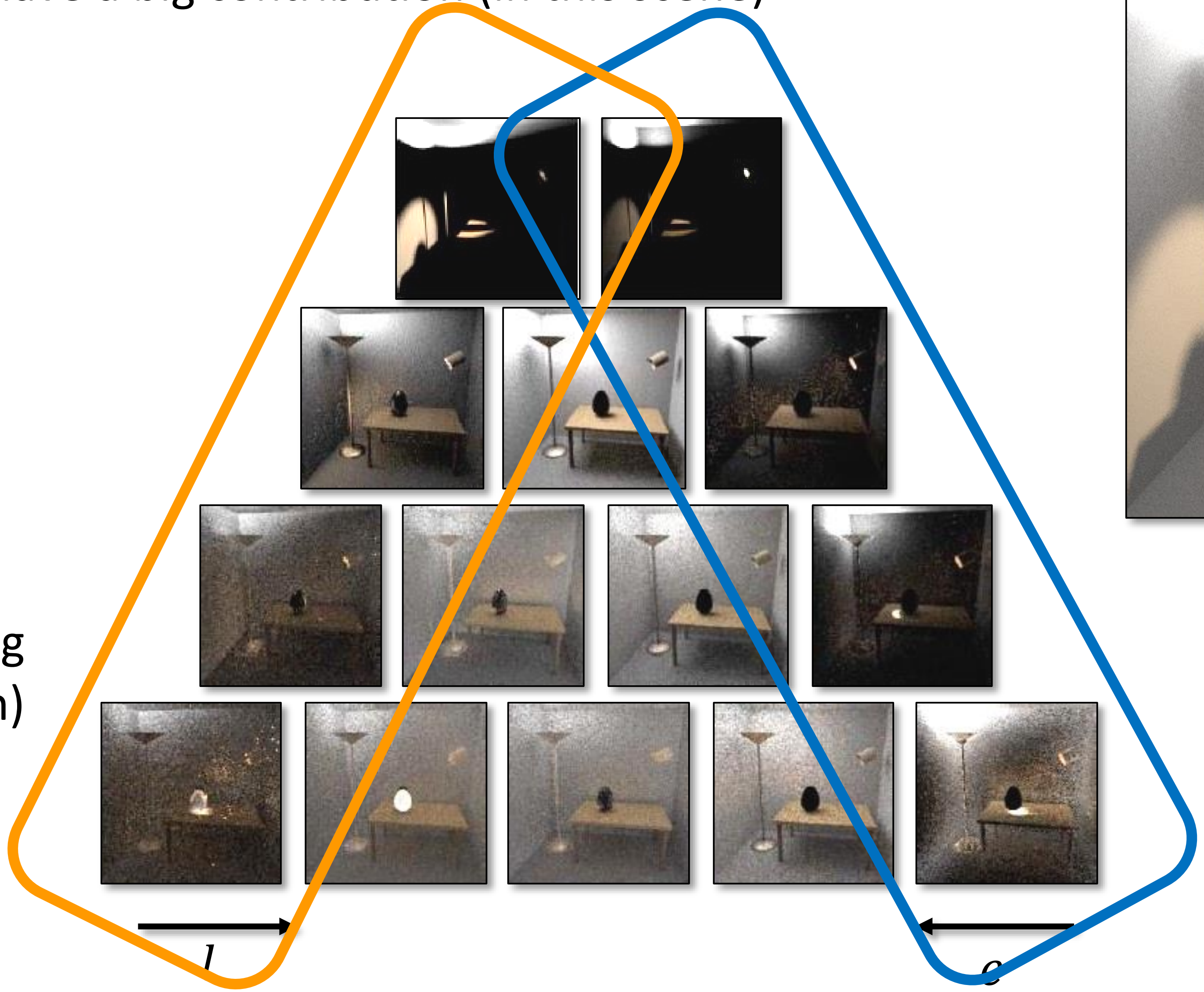
$l \rightarrow$ $\leftarrow e$

images: Eric Veach

Bidirectional path tracing

path pyramid: contributions from individual techniques

- inner connections have a big contribution (in this scene)



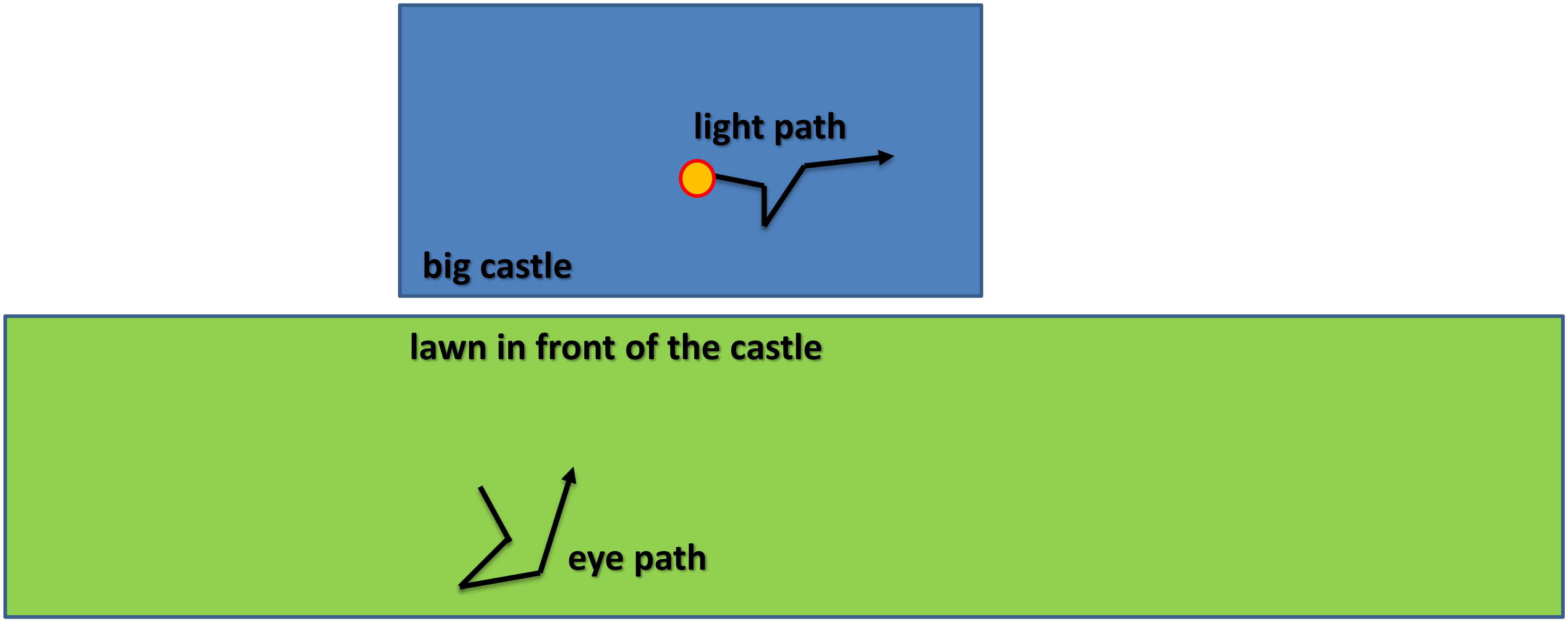
images: Eric Veach

BDPT/MIS character headshot

- simple setup: IBL + single key light. useful connections?

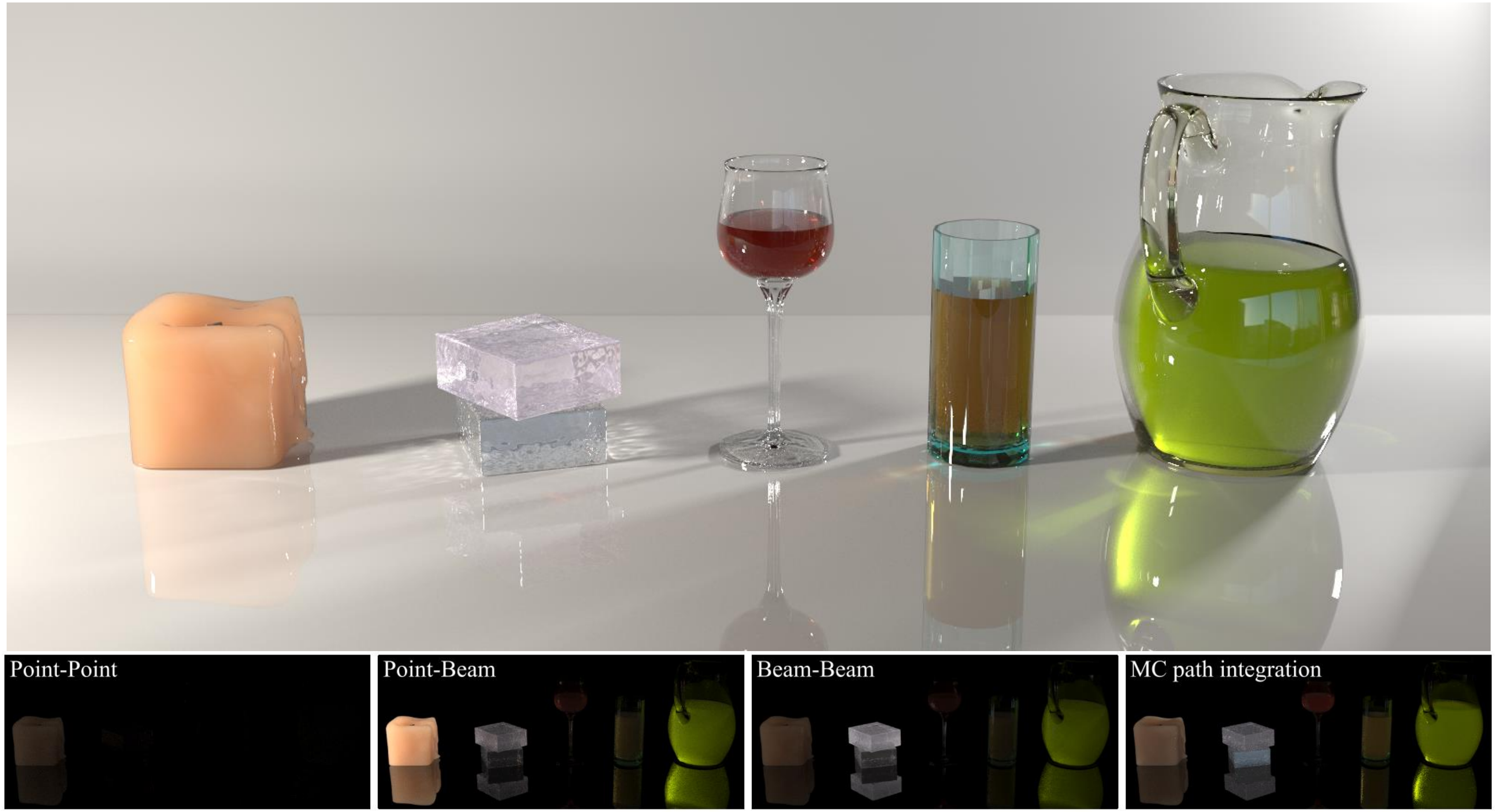
head scan: infinite realities

- many lights and complicated occlusion?
- connections encumbered by *visual complexity* (not scene size)



MIS with even more estimators

- gets worse: *Unifying Points, Beams, and Paths* [Krivanek et al. 2014]
- many estimators => long run times
- MIS `selects' only the best

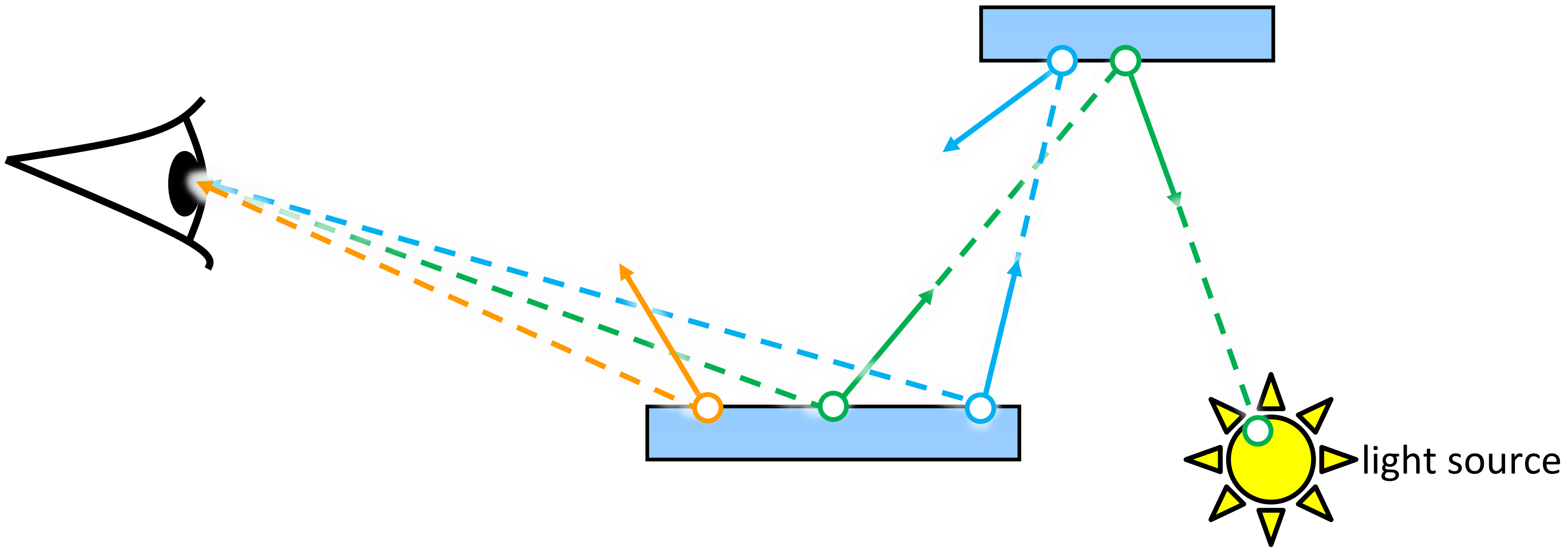


images:
Jaroslav Krivanek et al.

Keep it simple: path tracing

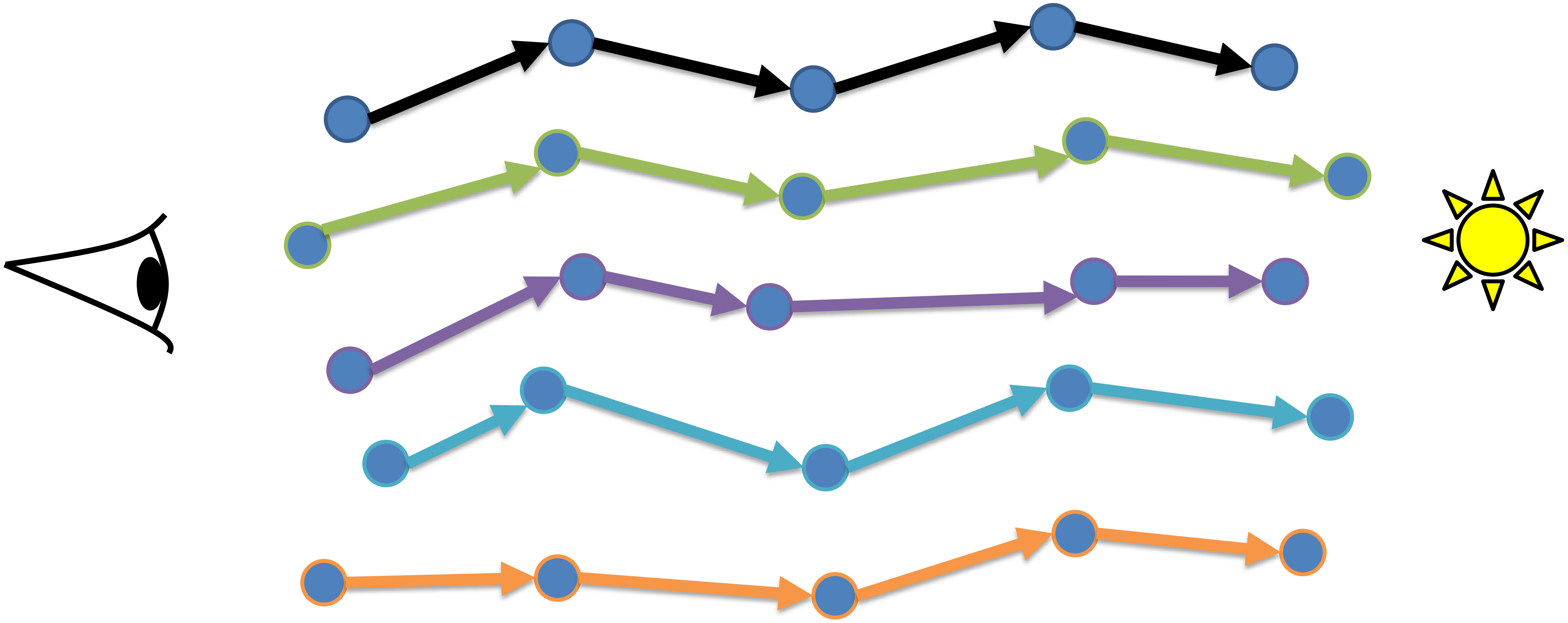
follow the recursive rendering equation

- importance follows pixels most closely
- light source sampling is difficult/sub-optimal
- do deterministic connections (next event estimation)



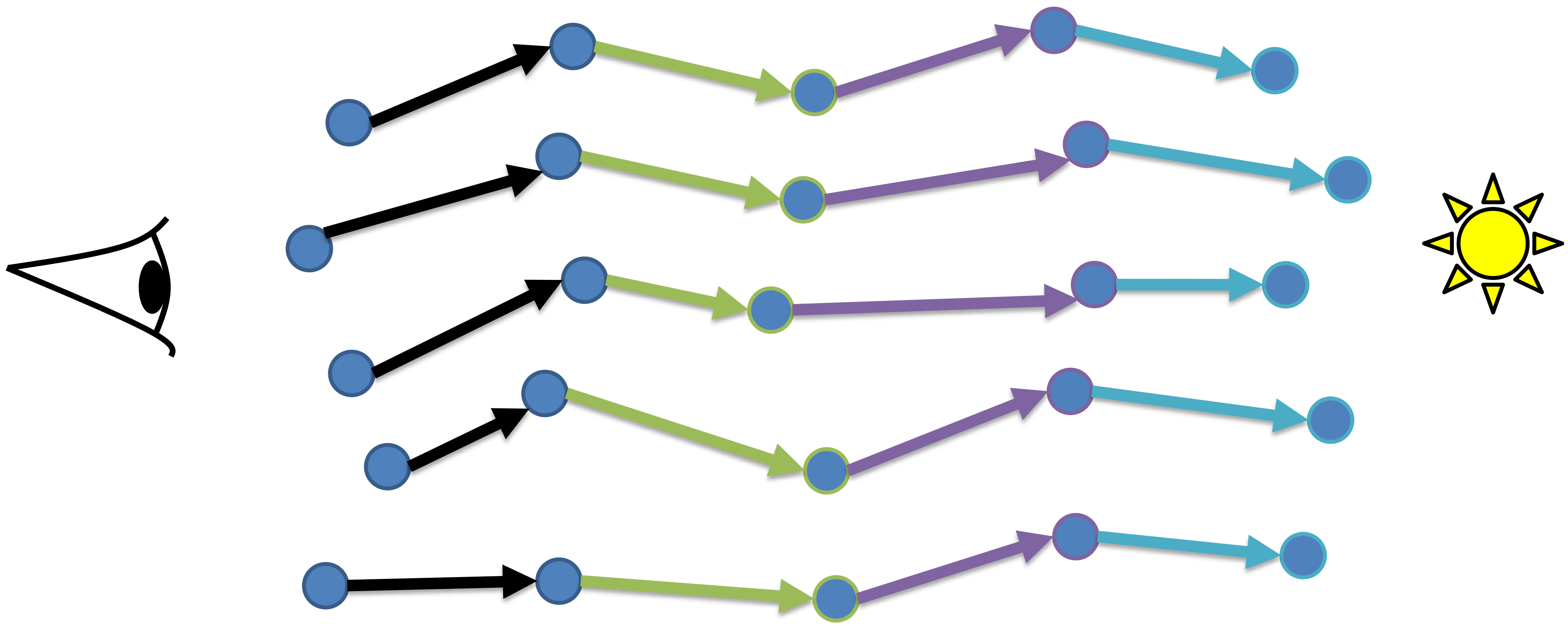
Implementation: order of execution

- trace full path first (flat order)
- **one** complete **path** in memory (colour coded)



Implementation: order of execution

- trace similar rays (wavefront order)
- **large buffers** of fragments of paths in memory at a time (colour coded)



Implementation: order of execution

- ray wavefront vs. flat order
  - locality/SIMD changes the runtime constant
  - full path allows more advanced sampling techniques

- Monte Carlo error is order $\varepsilon \sim \frac{\sigma}{\sqrt{N}}$
  - optimising raw performance tunes N and better algorithms tune σ
  - both may matter at low sample counts
  - need $4 \times N$ or $\sigma/2$ to achieve $\varepsilon/2$

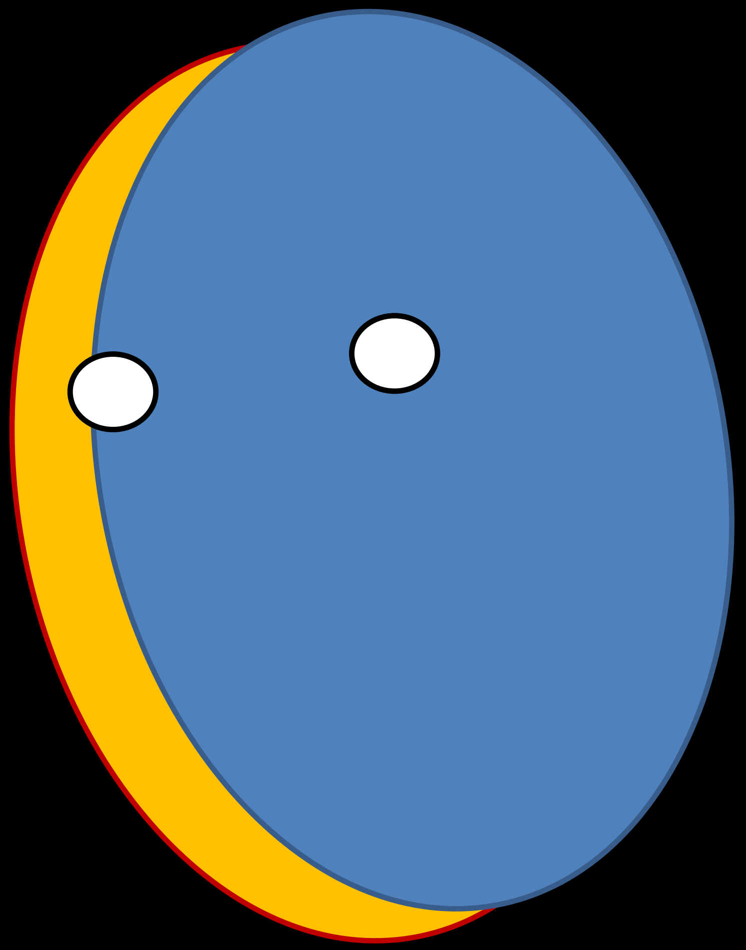
Implementation: order of execution

- error is order $\varepsilon \sim \frac{\sigma}{\sqrt{N}}$
  - raw performance $\Rightarrow N$
  - better algorithms $\Rightarrow \sigma$

- we tried both ordering approaches in Manuka
  - better algorithms are often times a huge gain!
    - Metropolis depends on full path
    - BDPT and complex path expressions to some extent
  - increased locality seems to only pay off when accessing **a lot** of memory / for out-of-core
    - we support RSL shading for pattern generation which helps reduce memory requirements

Path tracing: production asset

- typical lighting setup of a character with a rim light
- needs careful sampling to converge within the time budget

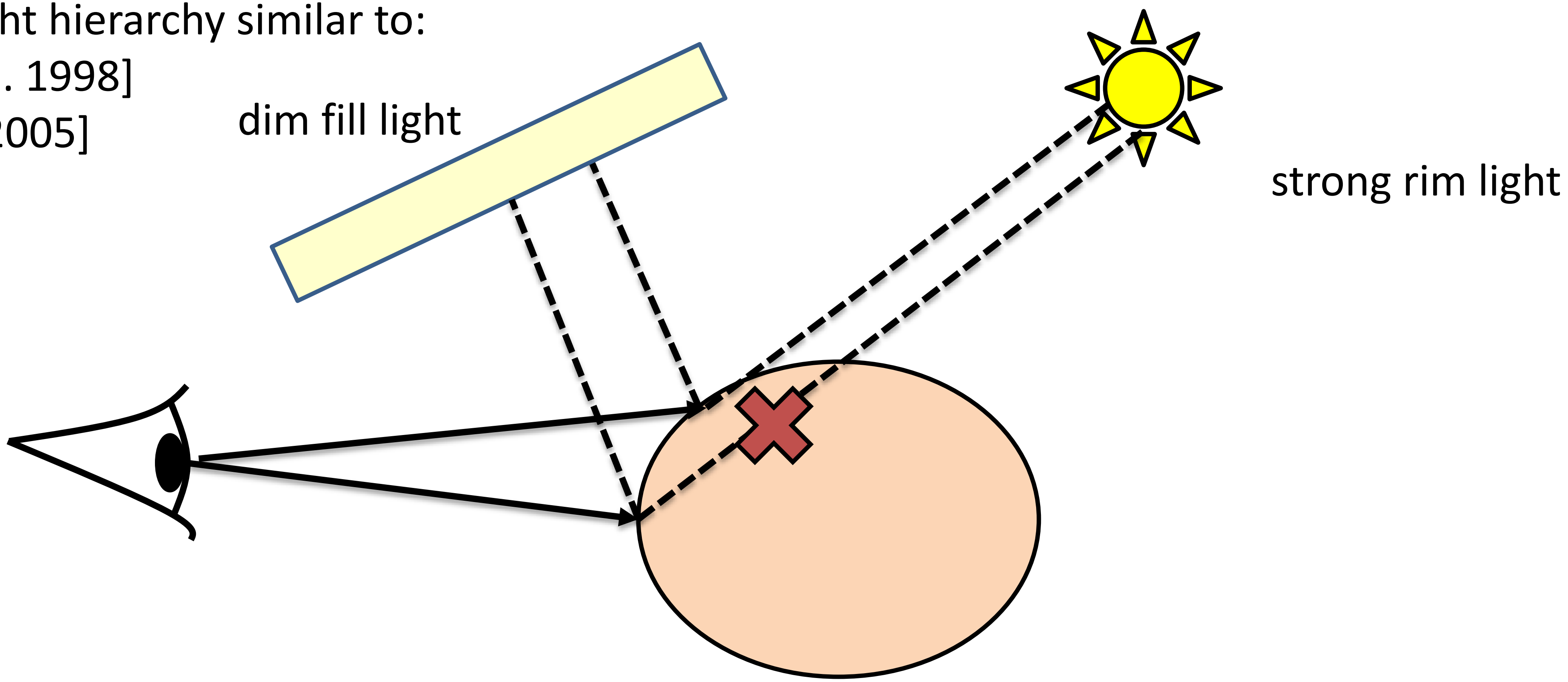


Careful next event estimation

- path tracing is great to sample importance from camera
- lights aren’t sampled equally well out of the box
- careful next event estimation
  - lightcuts hierarchies [Paquette et al. 1998, Walter et al. 2005]
  - manifold next event estimation [Hanika et al. 2015]
  - importance sampling of area lights in participating media [Kulla and Fajardo 2011]
  - portals on environment maps [Bitterli et al. 2015]
  - ..

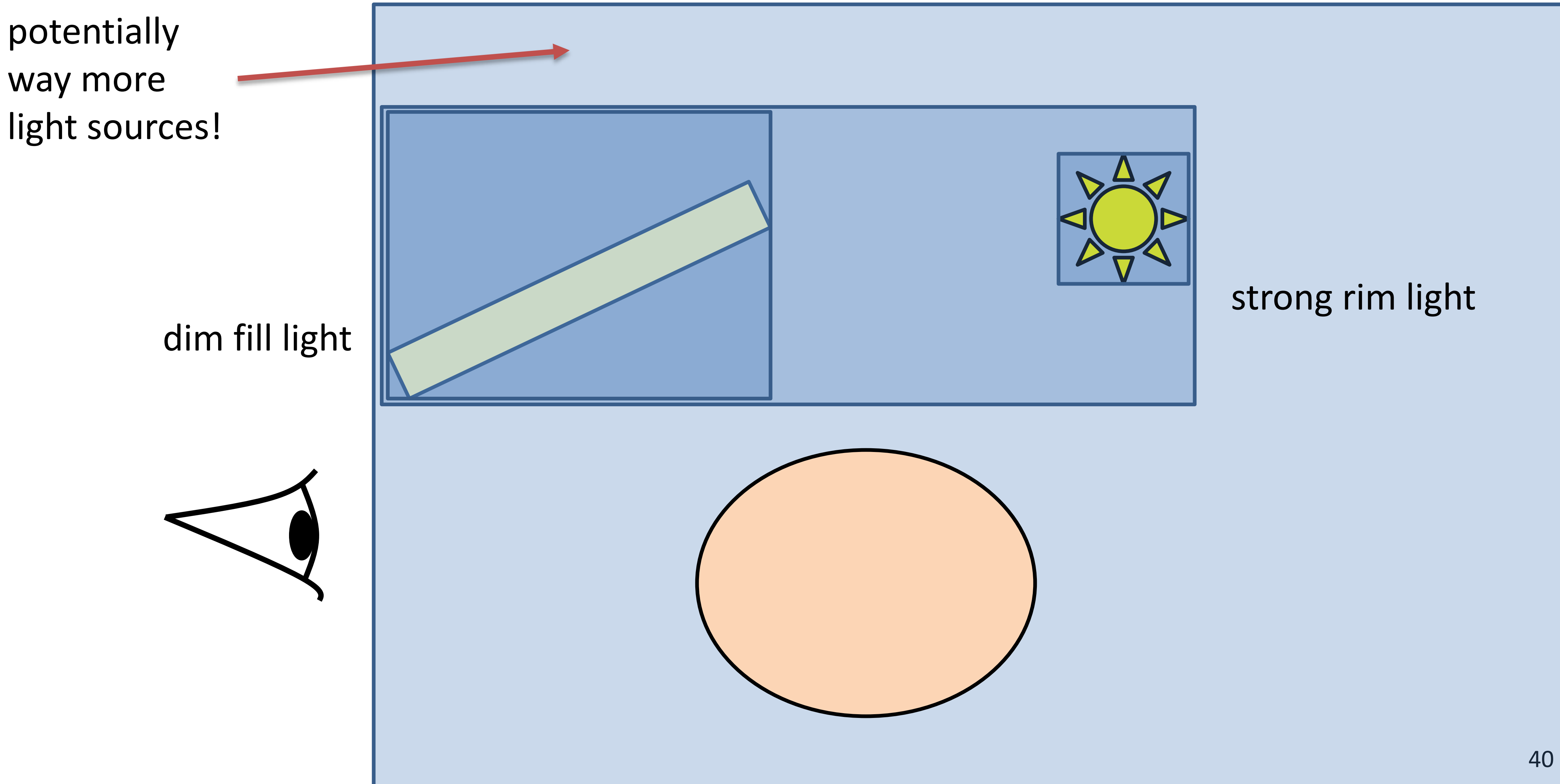
Next event estimation: light source selection

- need to avoid common cases:
  - sample strong rim light even though it is occluded
  - sample dim fill light too often if the rim light is visible
  - sample strong light sources which are too far away
- use a smart light hierarchy similar to:
  [Paquette et al. 1998]
  [Walter et al. 2005]
  [Göppel 2008]



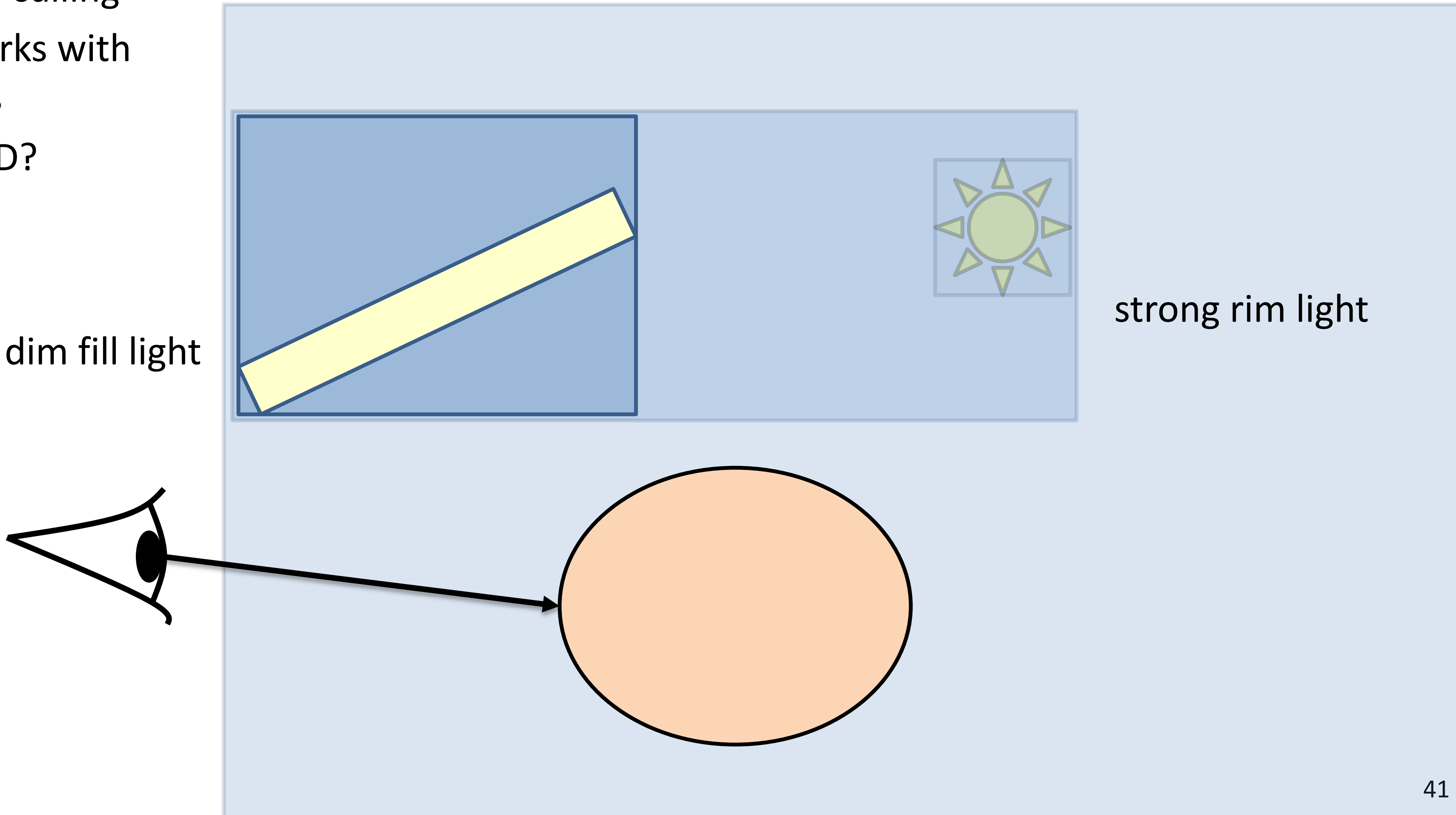
Next event estimation: light source selection

- use a smart light hierarchy similar to [Paquette et al. 1998, Walter et al. 2005, Göppel 2008]



Next event estimation: light source selection

- use a smart light hierarchy similar to [Paquette et al. 1998, Walter et al. 2005, Göppel 2008]
- do hierarchical culling
- still kind of works with ray wavefronts
  - locality/SIMD?



Manifold next event estimation

- back to the VCM problem case:
- specialised problem, can be solved with a specialised algorithm!

PT/MNEE
8spp

PT/NEE
8spp

VCM
8spp

Manifold next event estimation

- back to the VCM problem case:
- specialised problem, can be solved with a specialised algorithm!

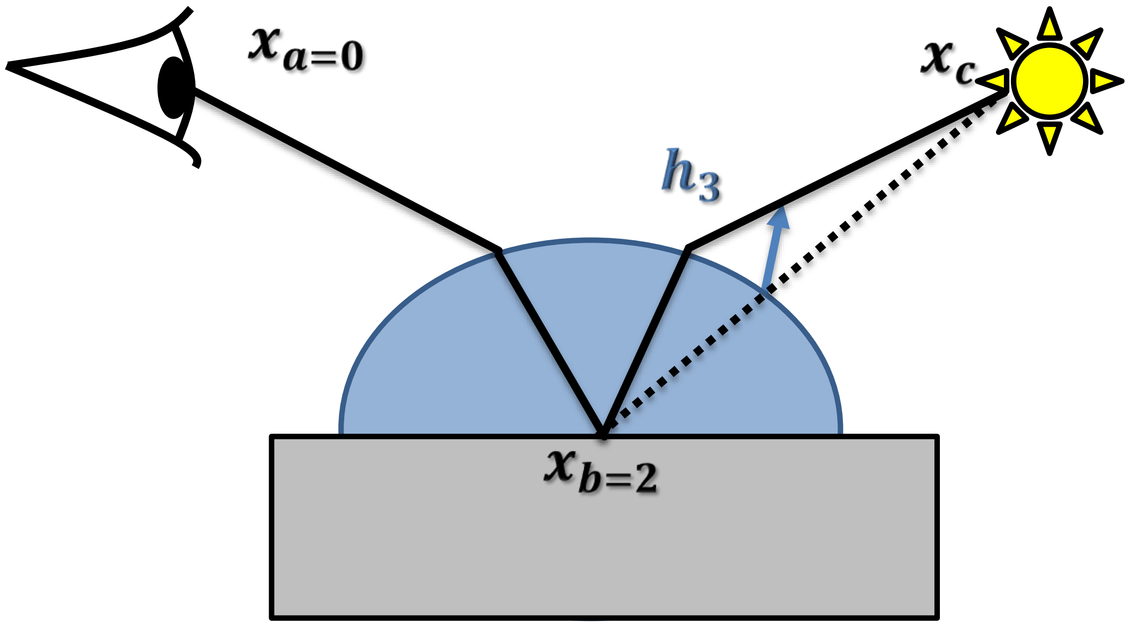
PT/MNEE
1024spp

PT/NEE
1024spp

VCM
1024spp

Manifold next event estimation

- sample point $\boldsymbol{x}_c$ on light source and create seed path $\boldsymbol{Y}$
- sample half vector $\boldsymbol{h}_3$, remember PDF in half vector space
- run (deterministic) Newtonian walk to find admissible path $\boldsymbol{X}$
- compute measurement contribution
- requires full path in memory!



Careful sampling in general

- hero wavelength sampling [Wilkie et al. 2014]
- on-line learning of mixture models [Vorba et al. 2014]
- gradient domain path tracing [Kettunen et al. 2015]
- ..

- more exciting new algorithms will be coming up in the community!
  - better keep our renderers ready for it!

Future Directions

- nothing is a solved problem 😐
  - reliably **discover** all interesting effects
    - quasi-Monte Carlo?
    - regularisation?
  - optimally **explore** regions of path space
    - MCMC
    - gradient domain shifts?
  - **memory access** times / **level of detail**
  - ..and all that in a tiny sampling **budget**!

Conclusions

- simple path tracing solves simple problems
  - ..but we have complicated problems!
  - the remaining 10%-20% of the pixels absolutely need bidirectional/photon maps/Metropolis

- good strategy to design a rendering architecture?
  - maintain freedom to try new algorithms (see `solved problems')
    - keep full path wherever you can (MNEE, ERPT, MLT, complicated path space expressions, ..)
  - be careful about memory accesses (but don't build the architecture around this idea)

thank you for listening!