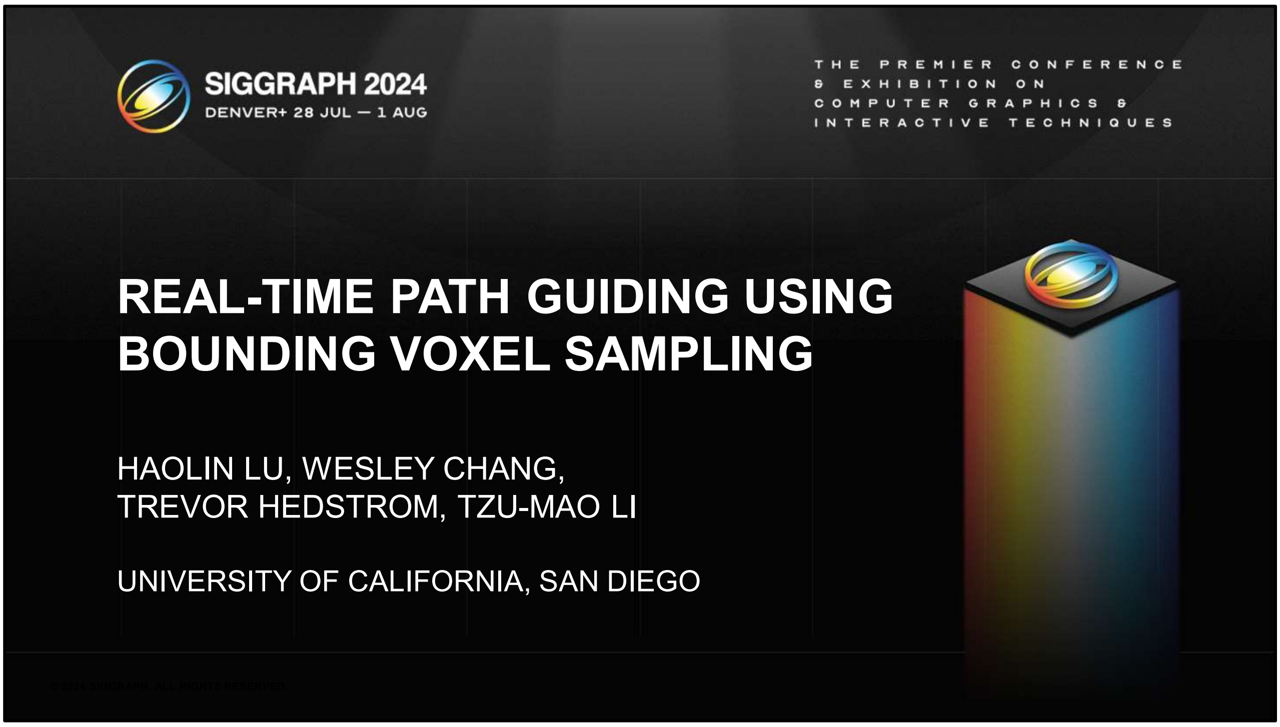


Thanks for the introduction. Hello, everyone. I am Haolin Lu. And the work is with my coauthors Wesley, Trevor and Tzu-mao

REAL-TIME RENDERING & RAY TRACING

SIGGRAPH 2024
DENVER+ 28 JUL – 1 AUG

*

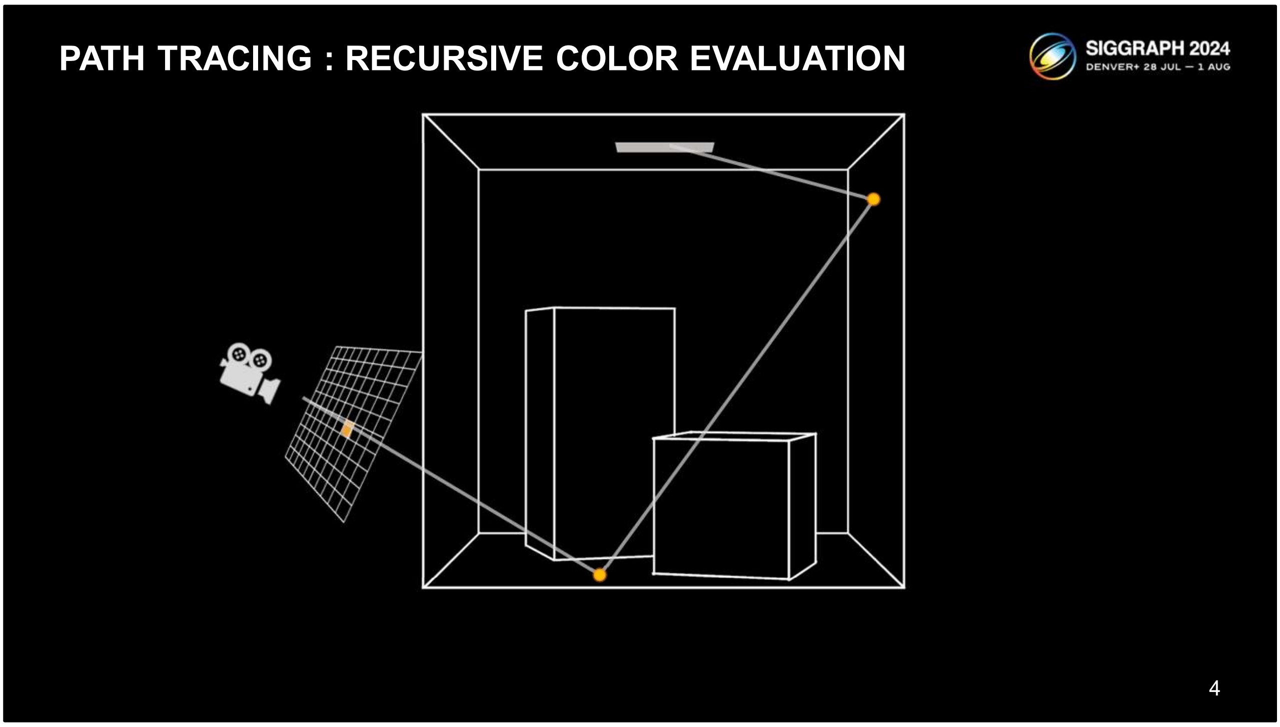
* Leading Lights: NVIDIA Researchers Showcase Groundbreaking Advancements for Real-Time Graphics

2

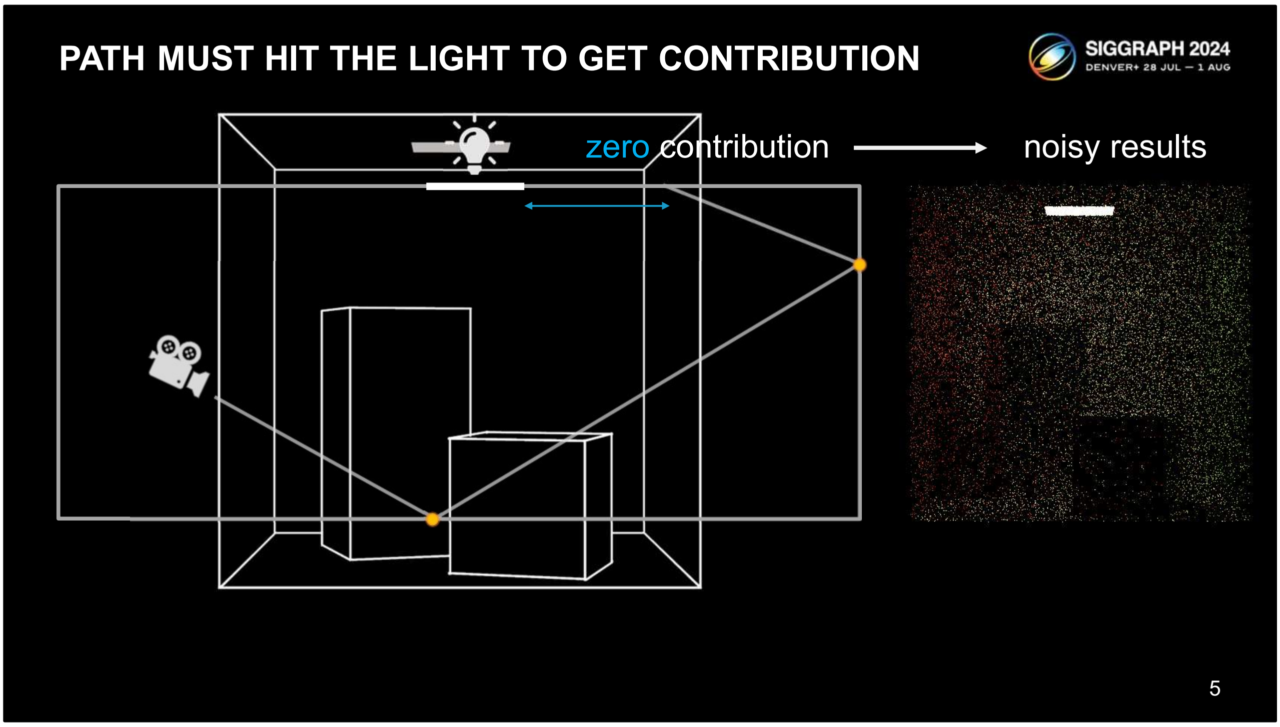
Real-time rendering is widely used in various interactive graphics applications. Recent advancements in hardware have also enabled real-time ray tracing.

Although we want to use path tracing everywhere,
it can be extremely noisy in some challenging scenes.

Therefore, we need a new solution to address these issues.

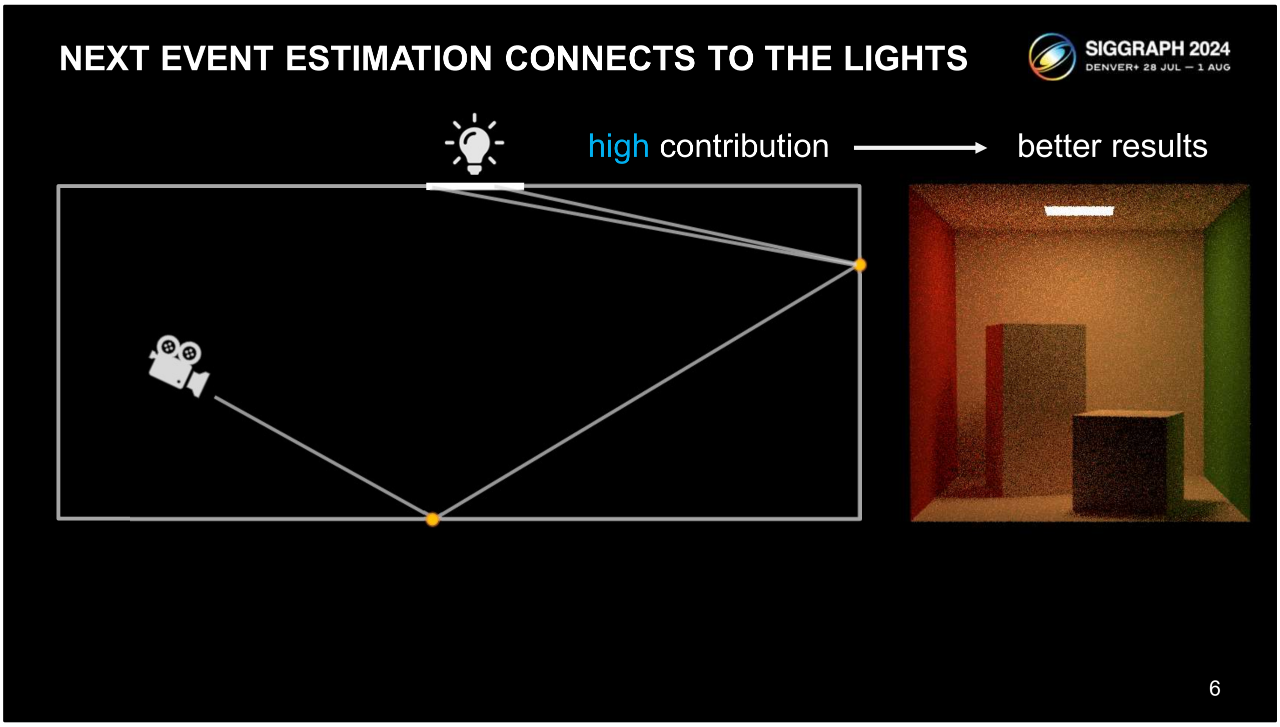


[click]
In path tracing, we compute pixel color by casting a ray into the scene
and evaluating the color at the shading point.
[click]
This involves recursively obtaining color
from other shading points
along the bounced ray
[click]
and, ultimately, from the light source.



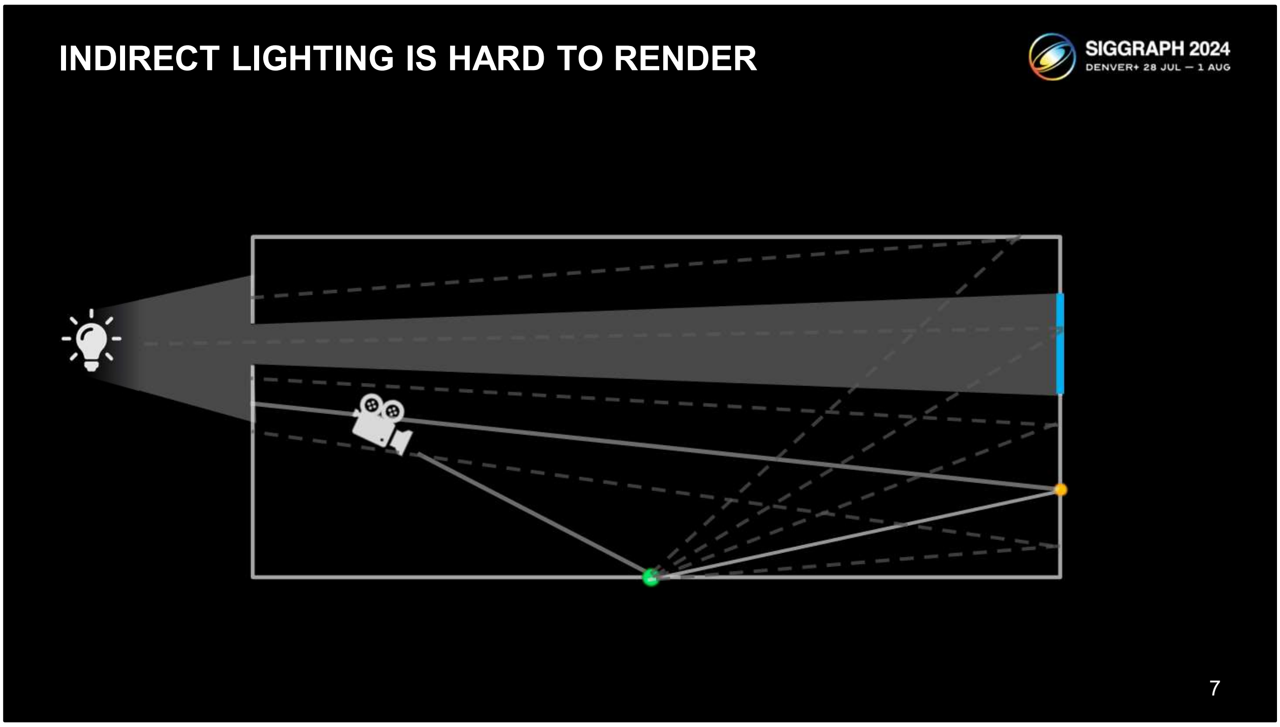
If a path cannot reach the light,
it adds zero contribution to the pixel;
which may lead to noisy results.

[click]
To avoid this, we use next-event estimation,
connecting the path directly to the light source.
which can improve the quality a lot.



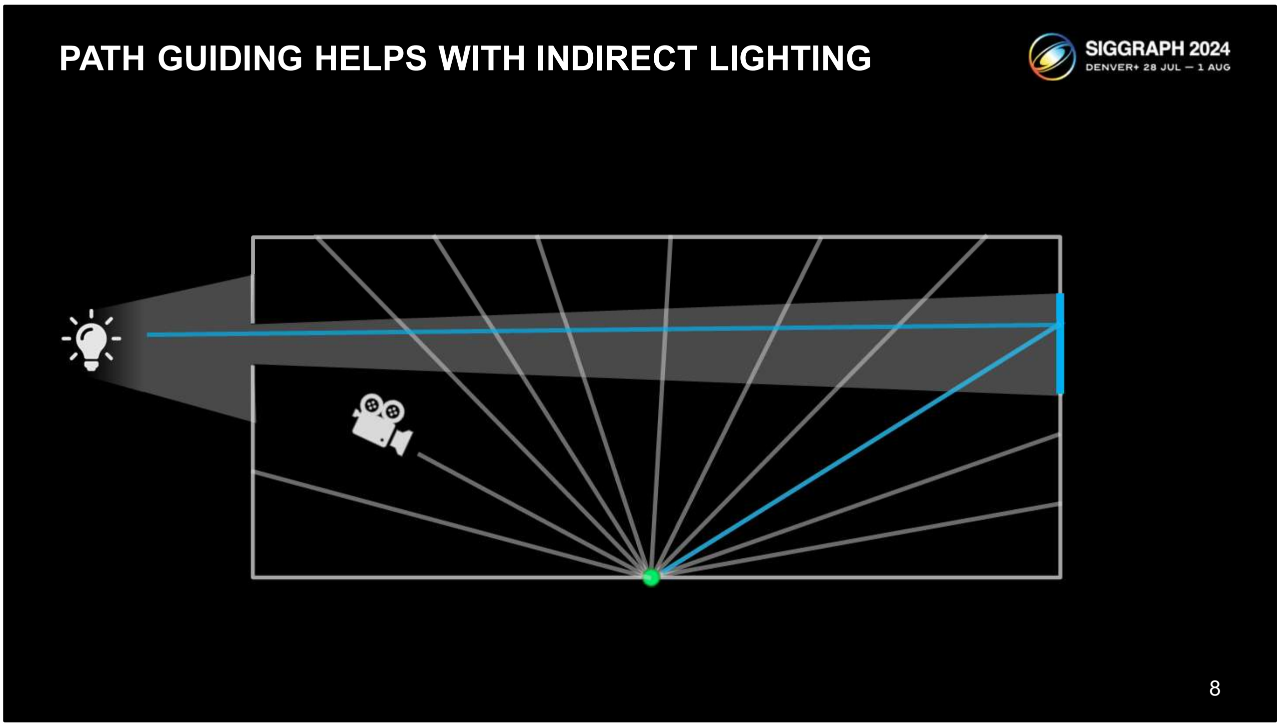
If a path cannot reach the light,
it adds zero contribution to the pixel;
which may lead to noisy results.

[click]
To avoid this, we use next-event estimation,
connecting the path directly to the light source.
which can improve the quality a lot.

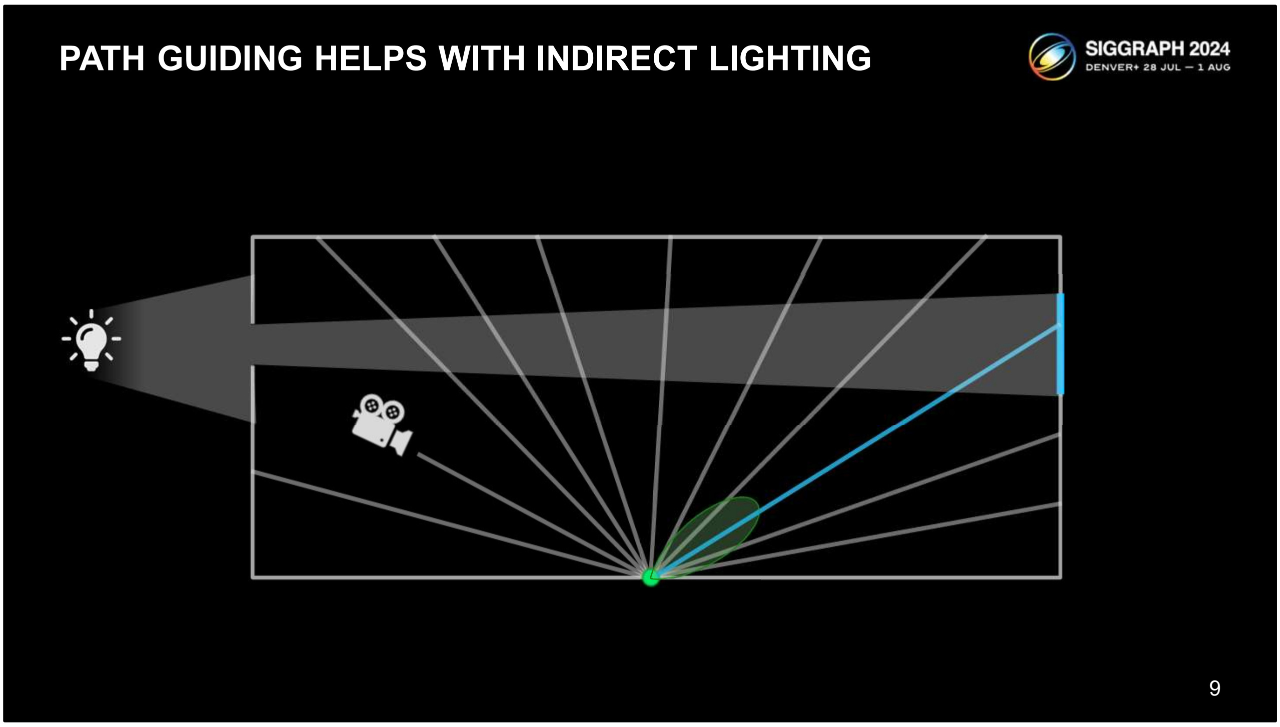


However, if we move the light outside of the box,
and illuminate the room through a small window.

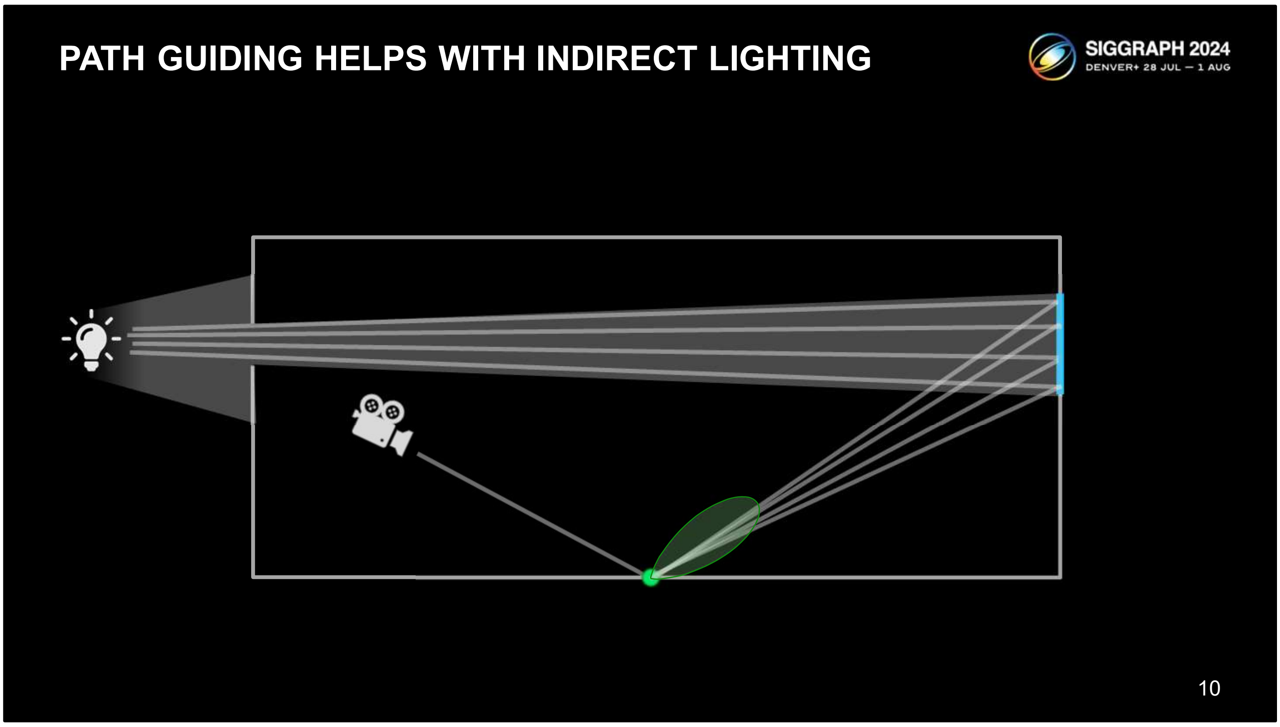
Next Event Estimation can no longer help,
[click]
As most connections will be blocked by the wall.
Therefore, we need another way out.



To solve this, path guiding starts with random ray tracing
[click]
and identifies directions that
[click]
carry a high radiance contribution



Then it cache the information as a local sampling distribution.



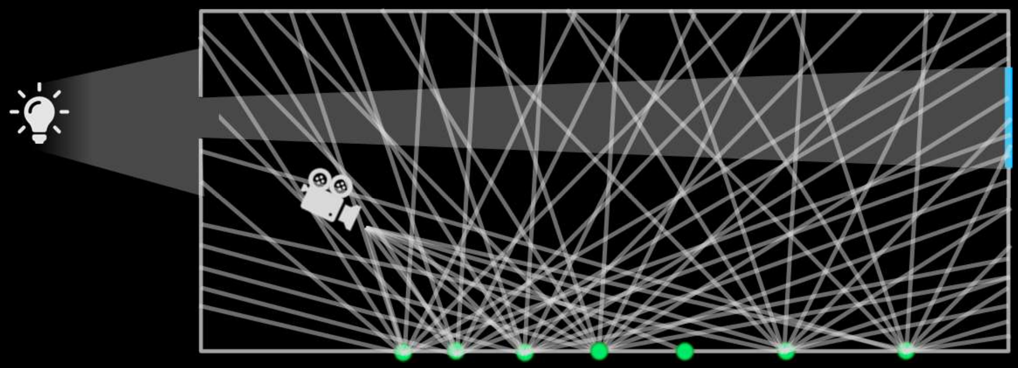
From the distribution, [click]
We can draw guided samples towards high-contributing directions,
Which helps us to reach the light source eventually.

Path-guiding methods for offline rendering work well,
but are too expensive to use in real-time rendering.
[click]
[02:00]
Building the sampling distribution
in the first place requires many samples.

EXISTING PATH-GUIDING METHODS ARE EXPENSIVE

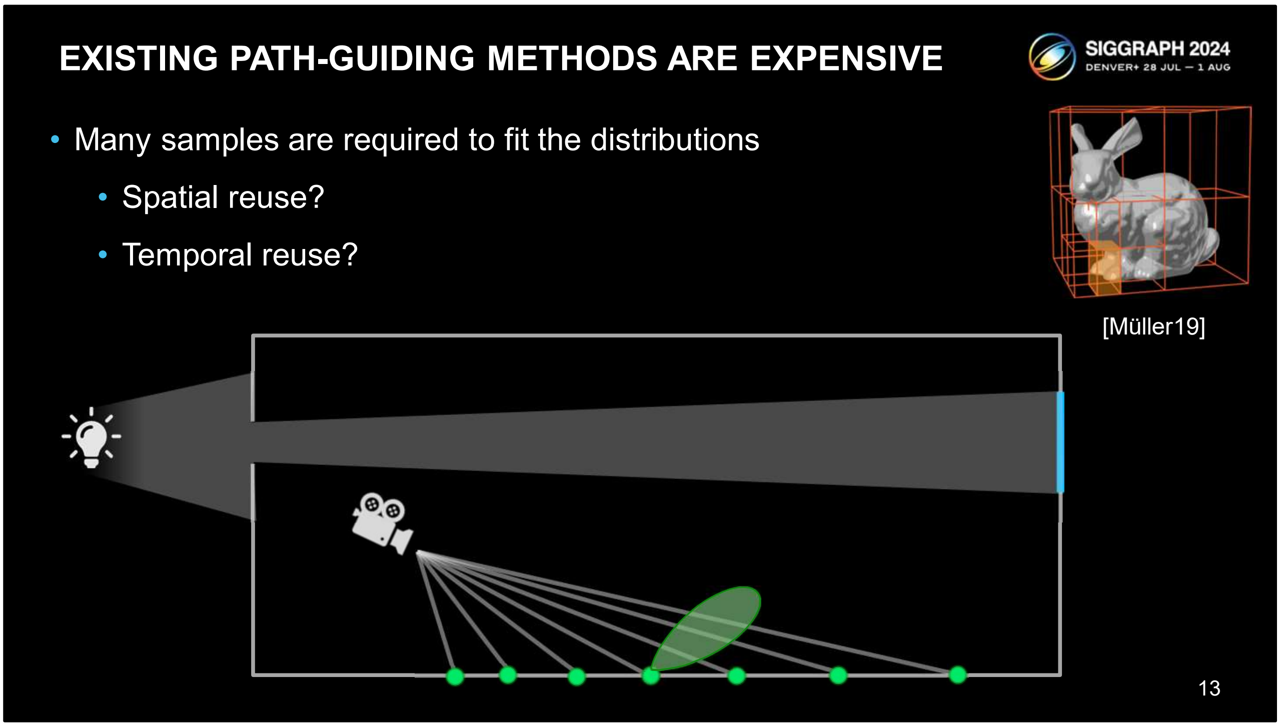
SIGGRAPH 2024
DENVER+ 28 JUL – 1 AUG

- Many samples are required to fit the distributions



12

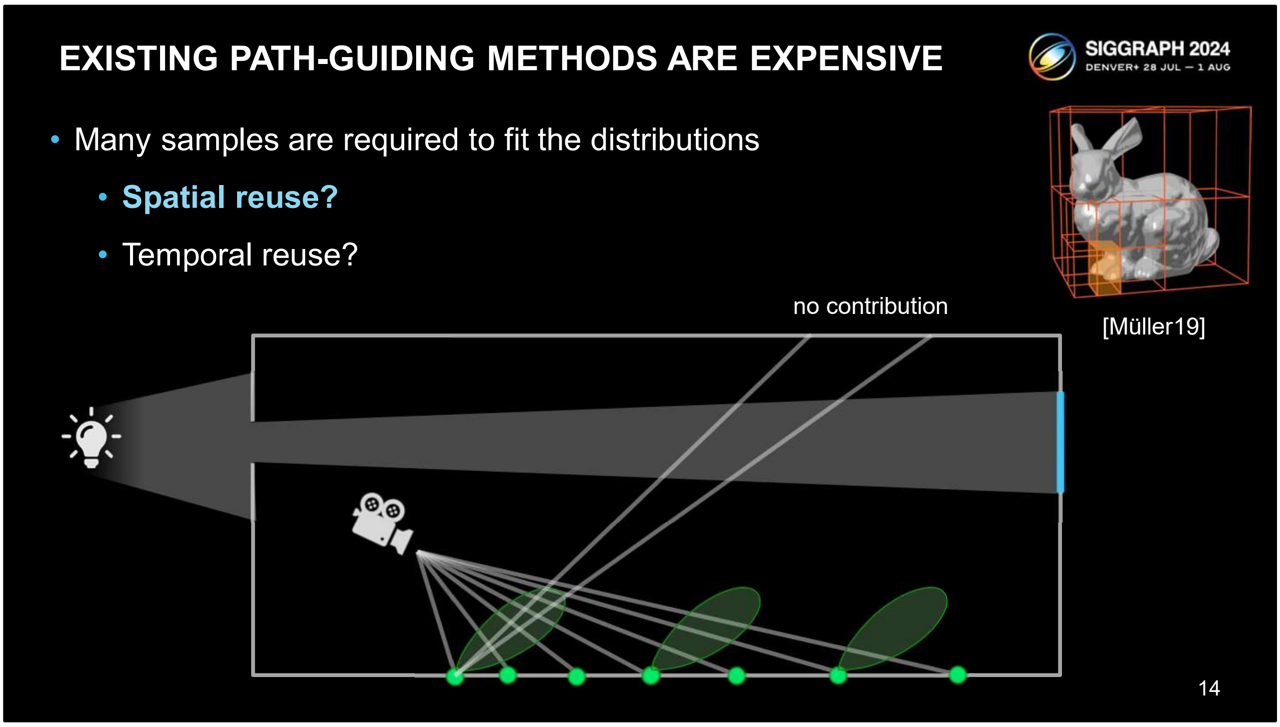
And this needs to be repeated at every shading point.
This is a problem;
because in real-time rendering,
we often only have time for 1-2 samples per frame.



Two common remedies for addressing sample count limitations
are spatial and temporal reuse.

[click]
Spatial reuse is widely used in offline path-guiding

[click]
by sharing one distribution within nearby regions.

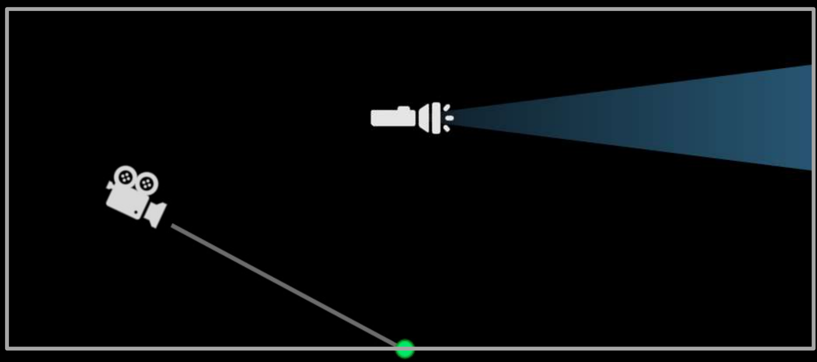


However, if we directly reuse the distribution for shading points far away,
[click]
they will not hit the same importance regions,
And leading to low or even zero contribution.
Therefore, spatial reuse is limited to very local neighbors.

EXISTING PATH-GUIDING METHODS ARE EXPENSIVE

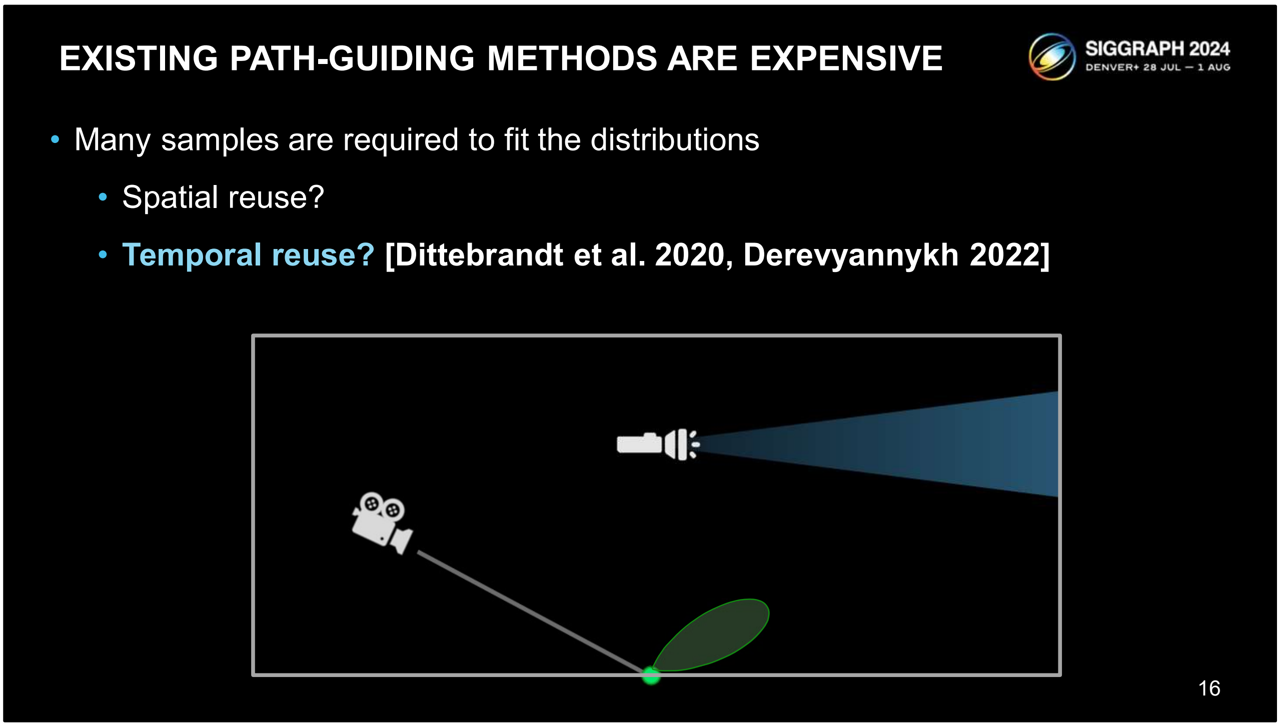
SIGGRAPH 2024
DENVER+ 28 JUL – 1 AUG

- Many samples are required to fit the distributions
  - Spatial reuse?
  - **Temporal reuse? [Dittebrandt et al. 2020, Derevyannykh 2022]**



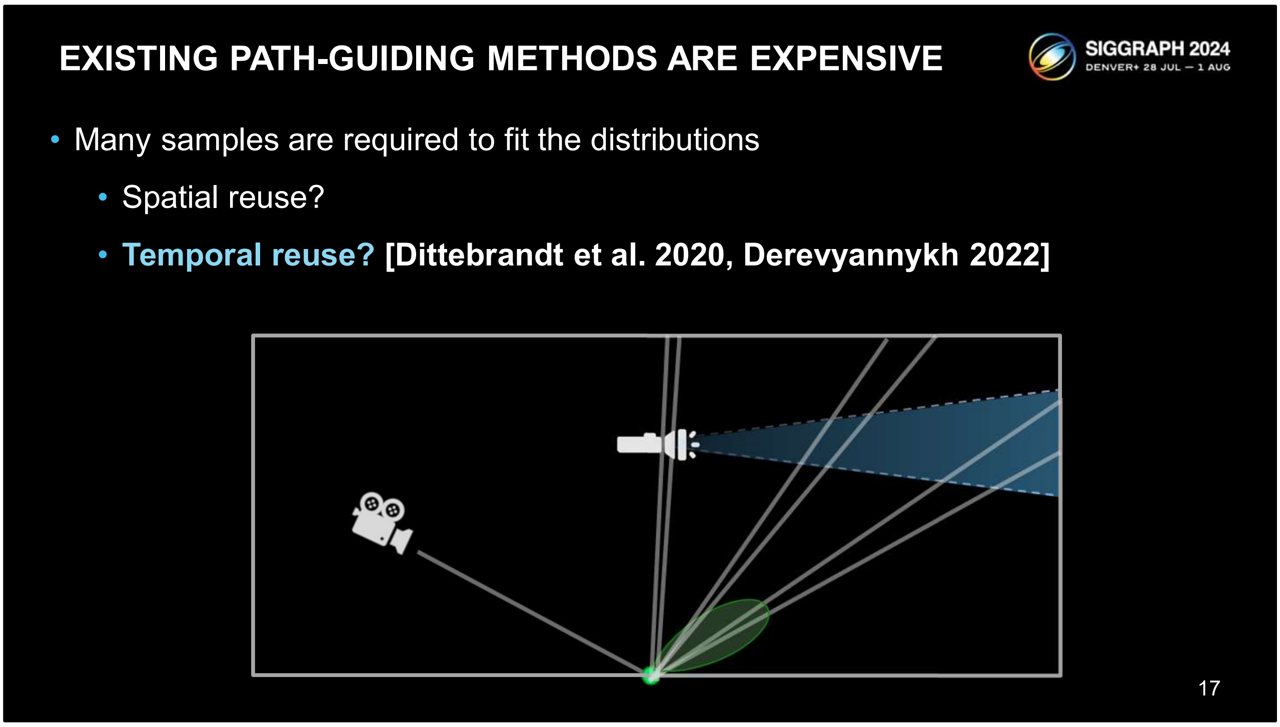
15

[02:45]
Previous work also widely uses temporal reuse,
which works great in static cases
but becomes problematic in dynamic settings



[click]
For example, consider a scene lit by a moving spotlight that rotates across frames.

[click]
And we have already learned a good distribution for the current frame.

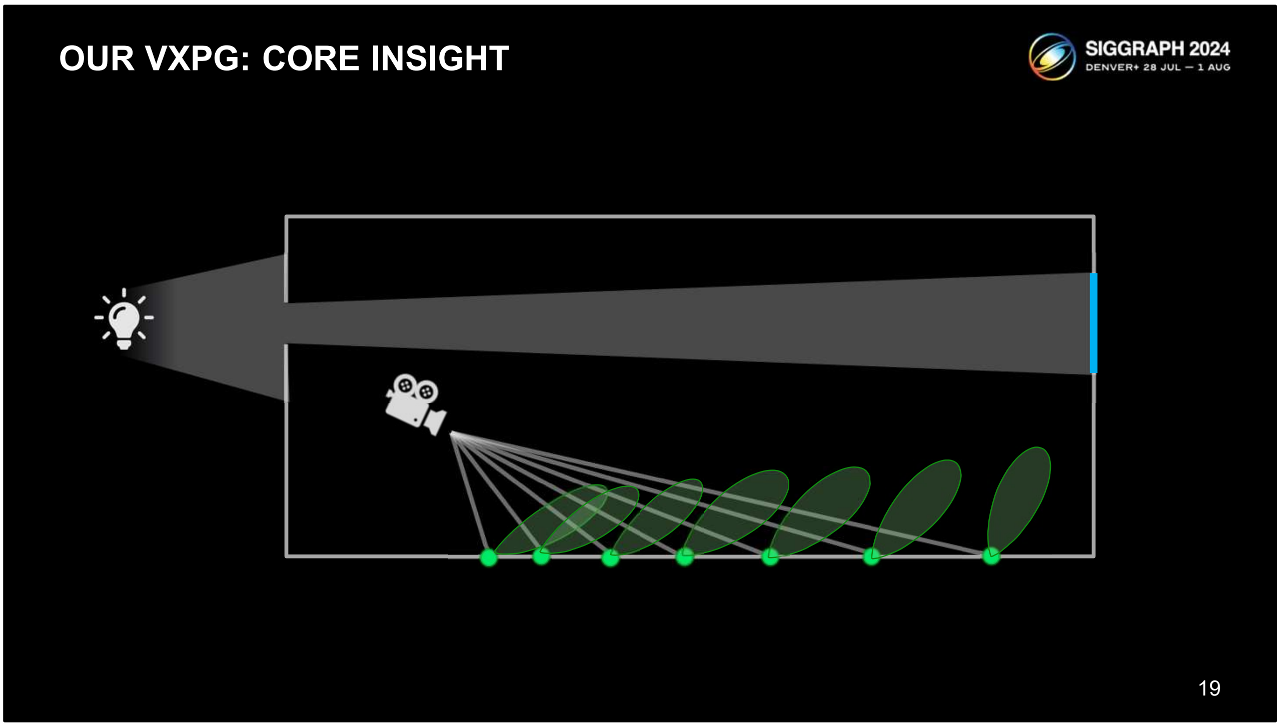


In a new frame, the spotlight will rotate a bit.
[click]

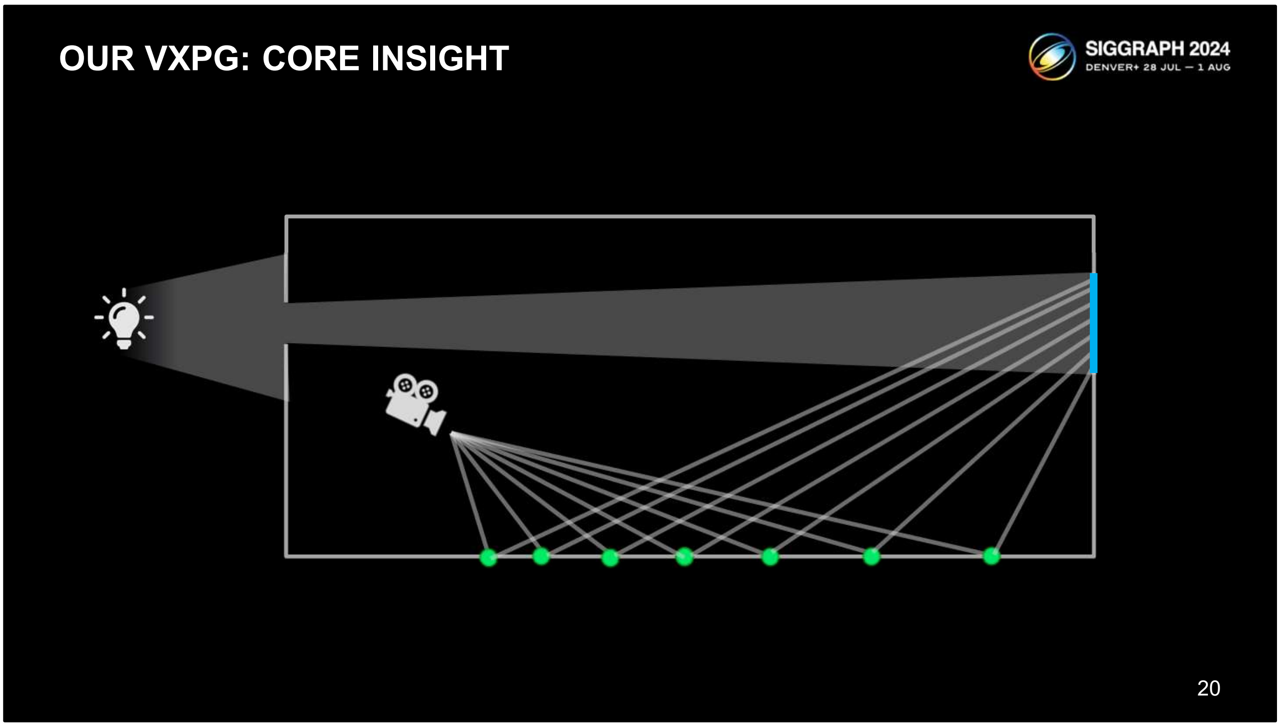
But due to temporal reuse, our distribution can hardly keep up,
[click]
so path guiding will then direct rays towards outdated regions.

[click]
And if the lighting is just constantly moving,
The guiding distribution can never catch up,
As a result, path guiding actually becomes path misguiding.

To achieve practical path guiding in real-time, we propose a new approach, called voxel path guiding.

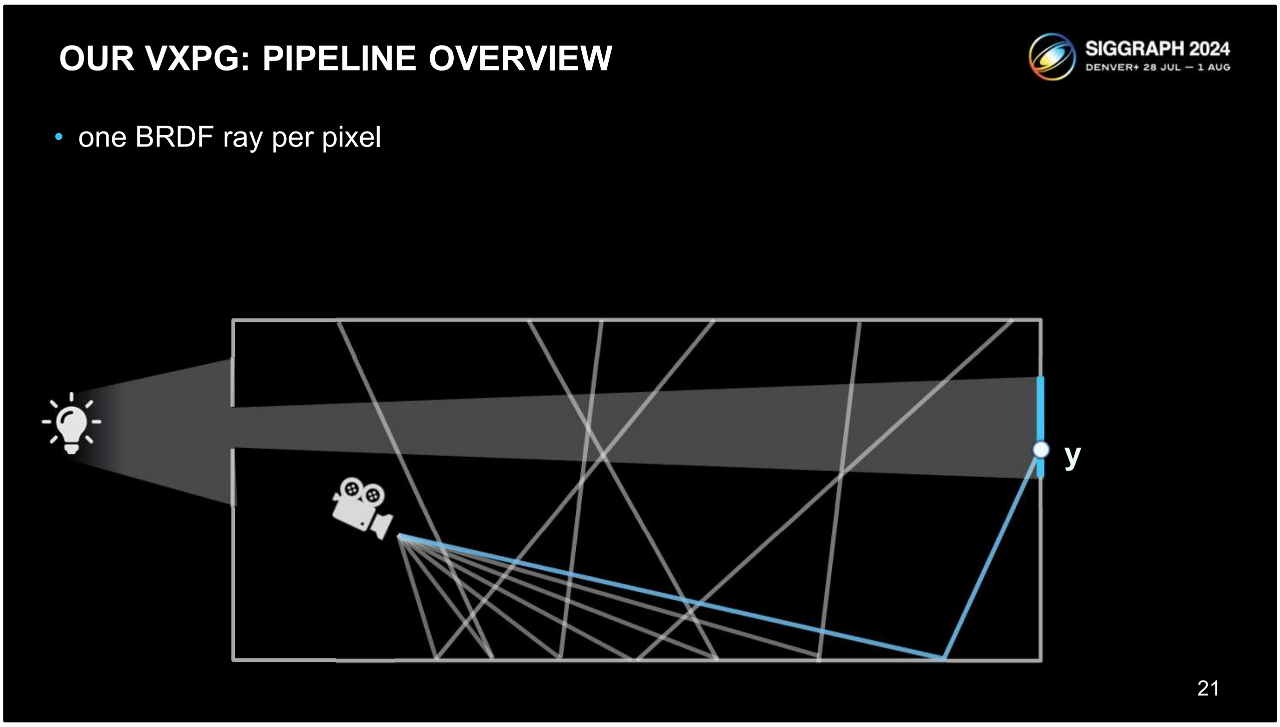


[03:33]
Our core insight is that learning all these
directional distributions locally
Is quite inefficient. As spatial reuse is so limited.

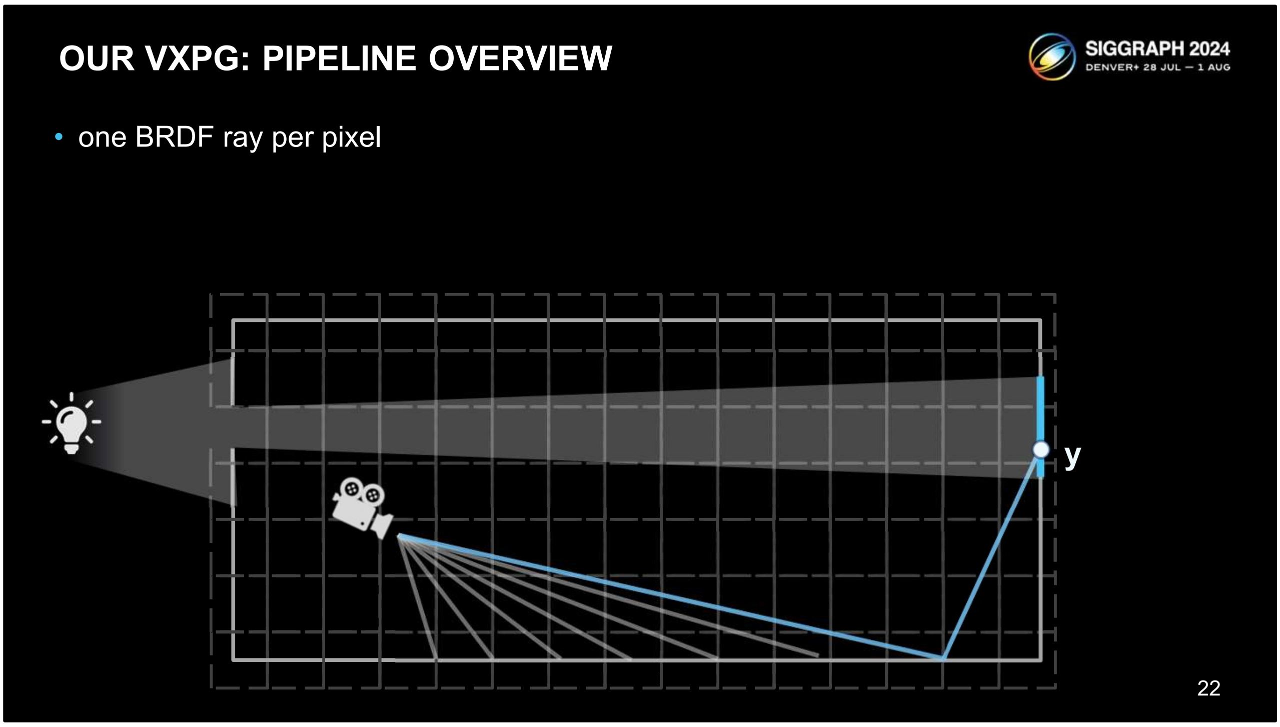


Instead, if we know
[click]
where this blue region is,
we can follow the spirit of Next Event Estimation,
And try to connect all shading point towards this region.

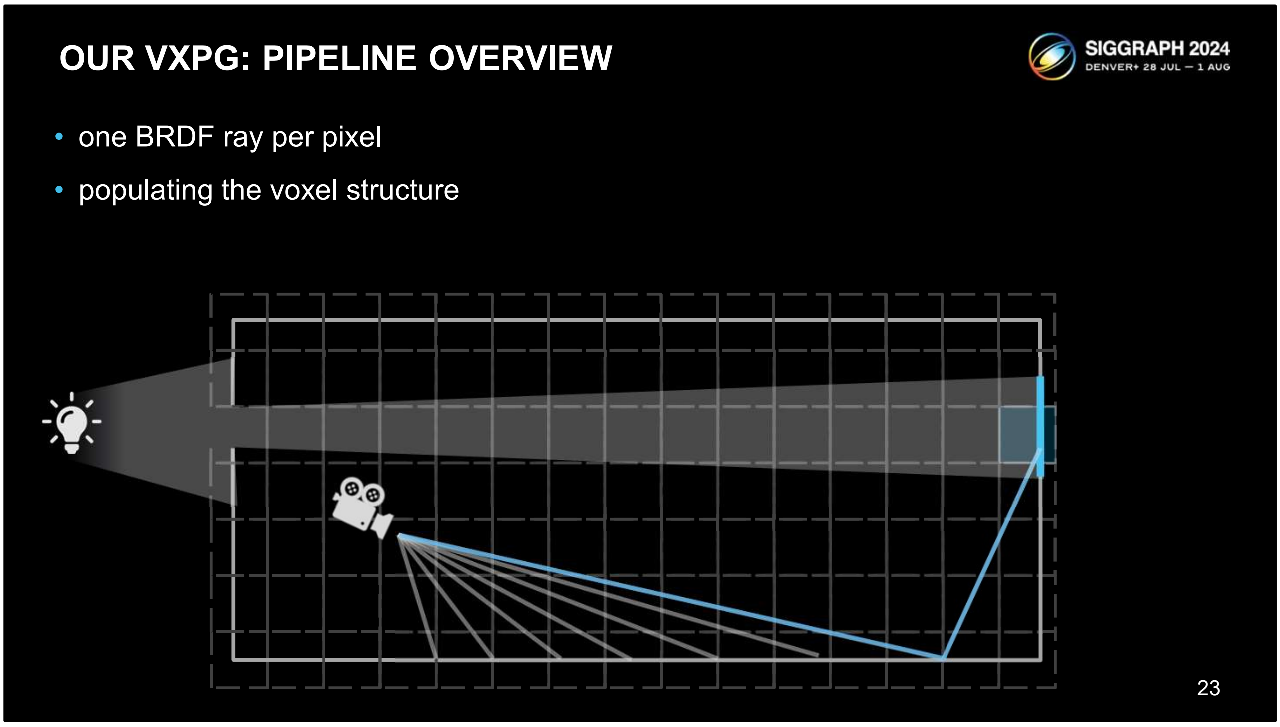
And it turns out that finding this blue region is much easier.



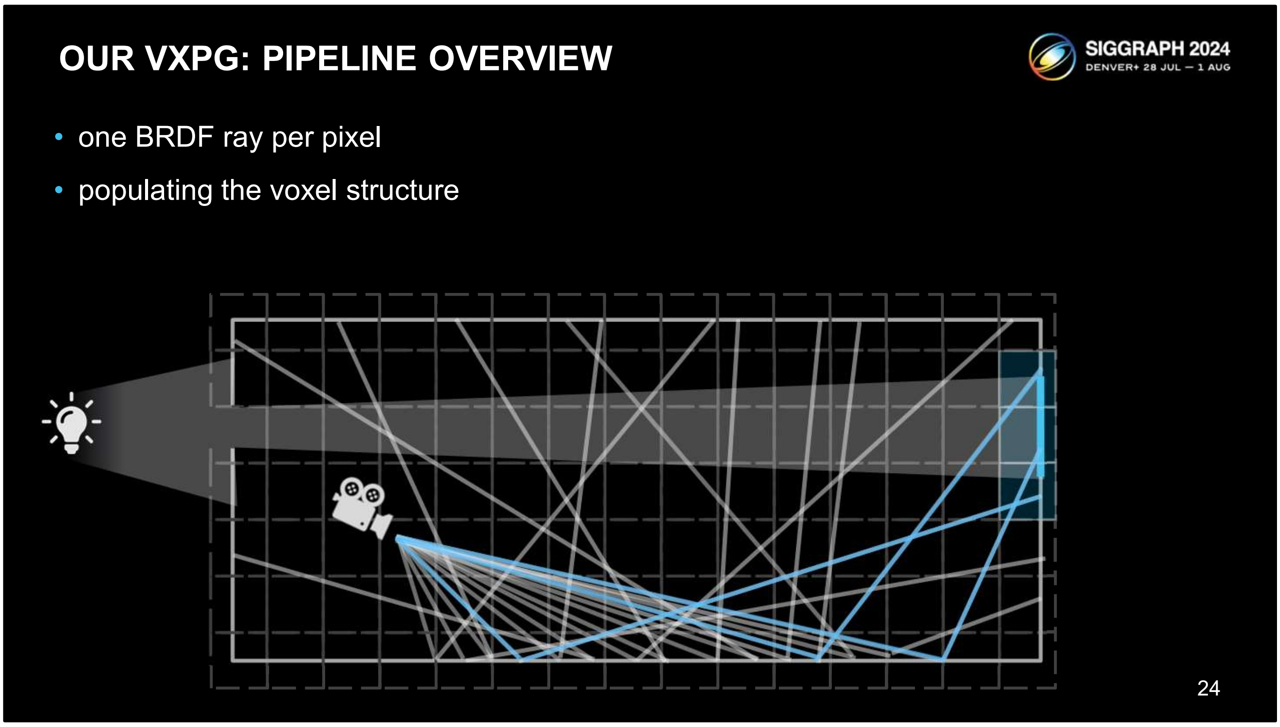
Our pipeline is a lightweight realization of the core insight.
To begin with, we draw one sample per pixel using
[click]
BRDF importance sampling.
Although hitting the blue region is challenging, as we have millions of pixels,
[click]
there is a good chance that at least one or two lucky paths will hit it,
For example, at path vertex y.
[click]



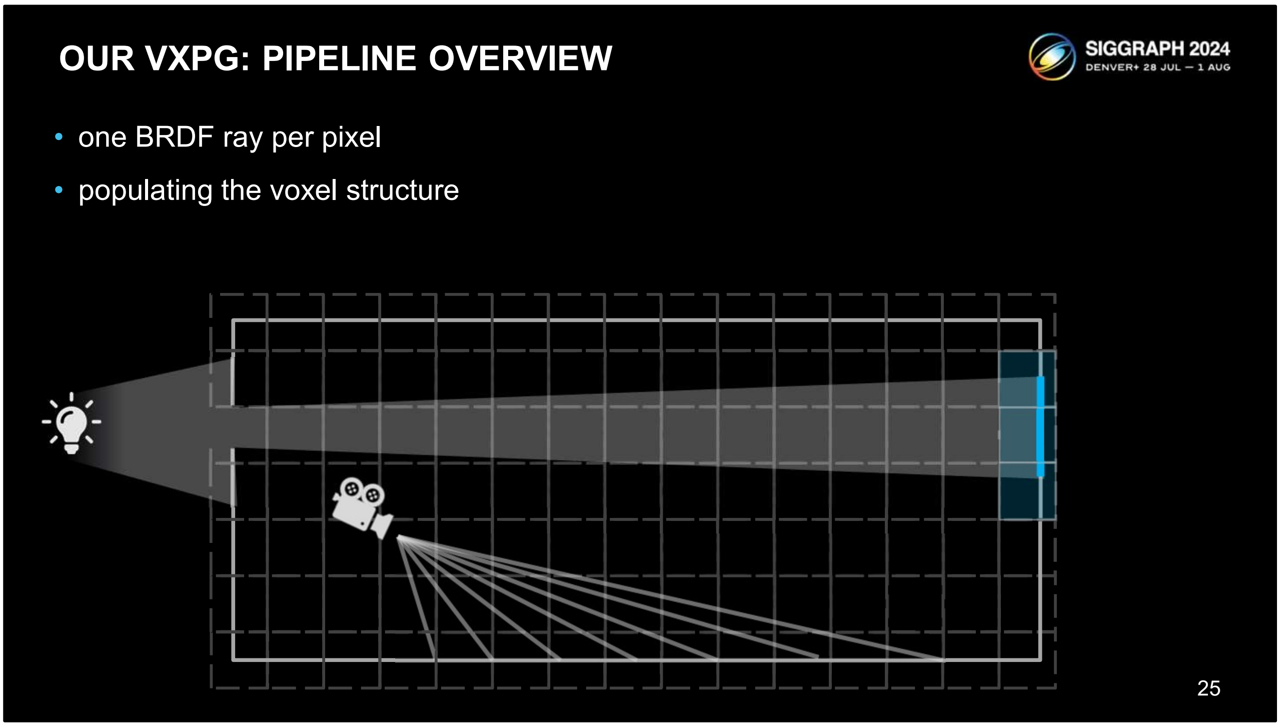
Obviously, we cannot recover the entire blue region from individual points just like y.
However, we can generalize this by embedding the scene with a voxel data structure.
[click]
Then, we simply populate the voxel that contains the vertex y.



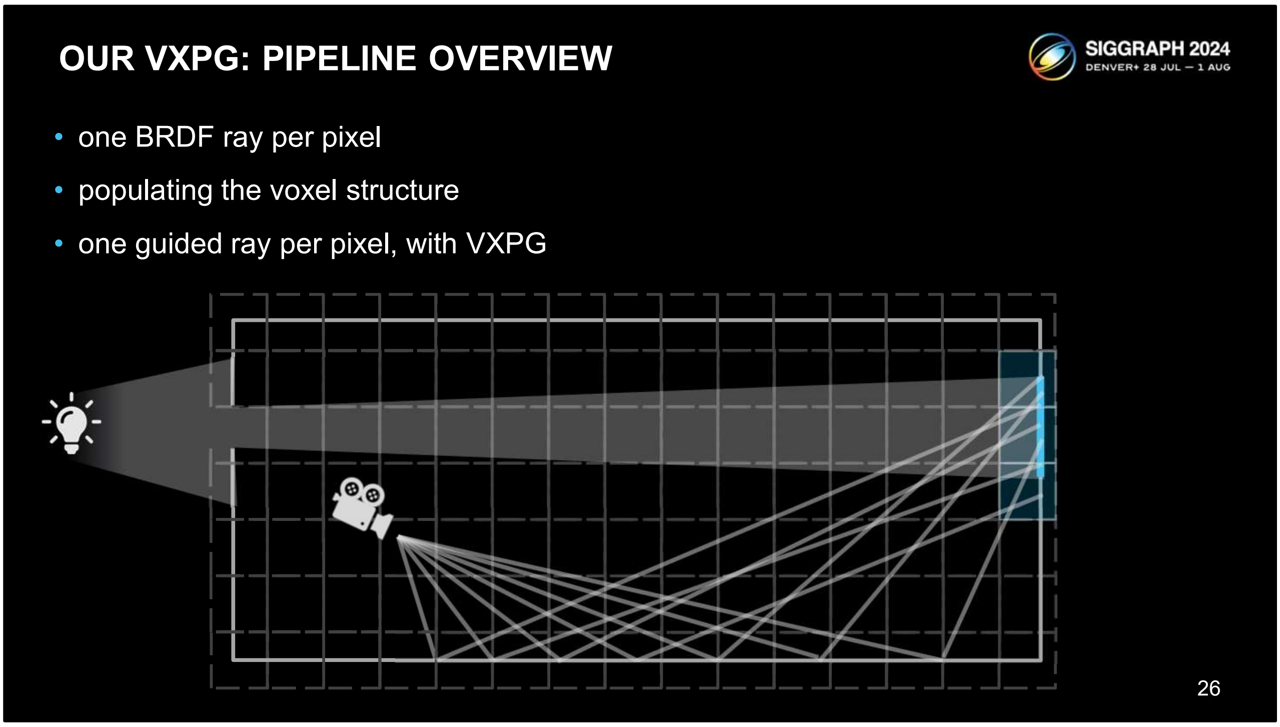
Obviously, we cannot recover the entire blue region from individual points just like y.
However, we can generalize this by embedding the scene with a voxel data structure.
[click]
Then, we simply populate the voxel that contains the vertex y.



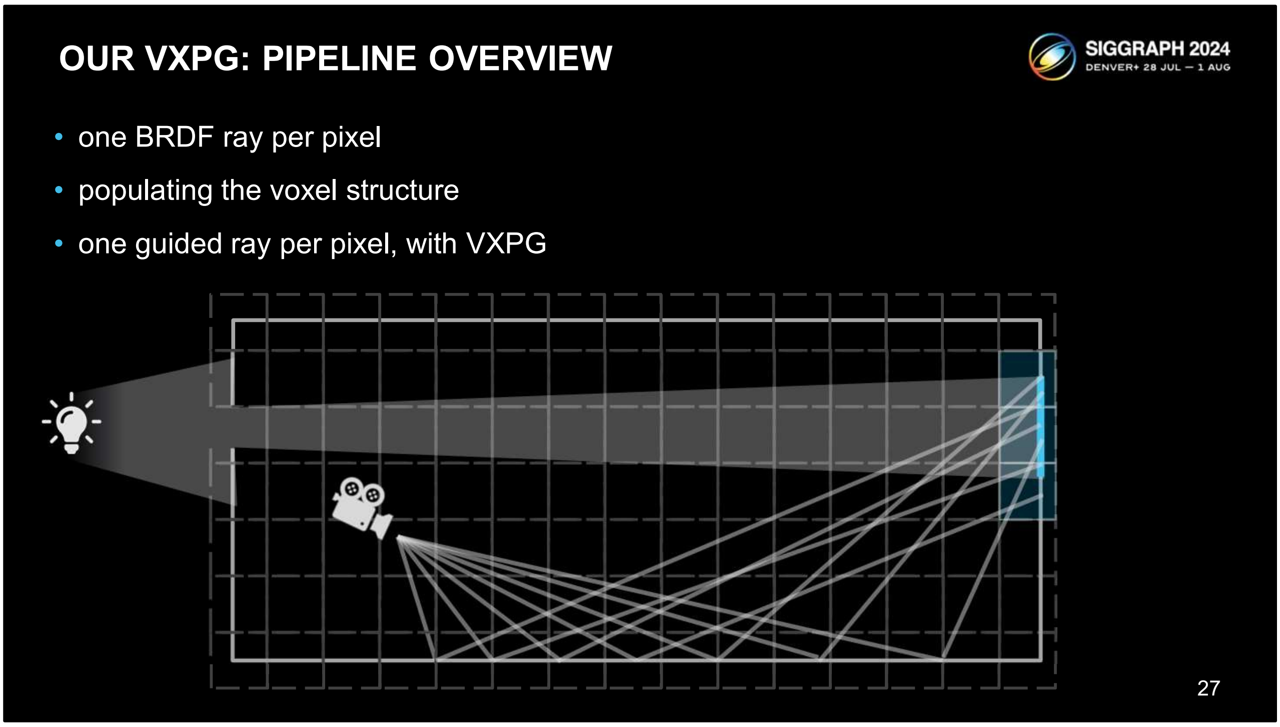
In practice, we have numerous pixels,
[click]
so hopefully most related voxels can get populated.



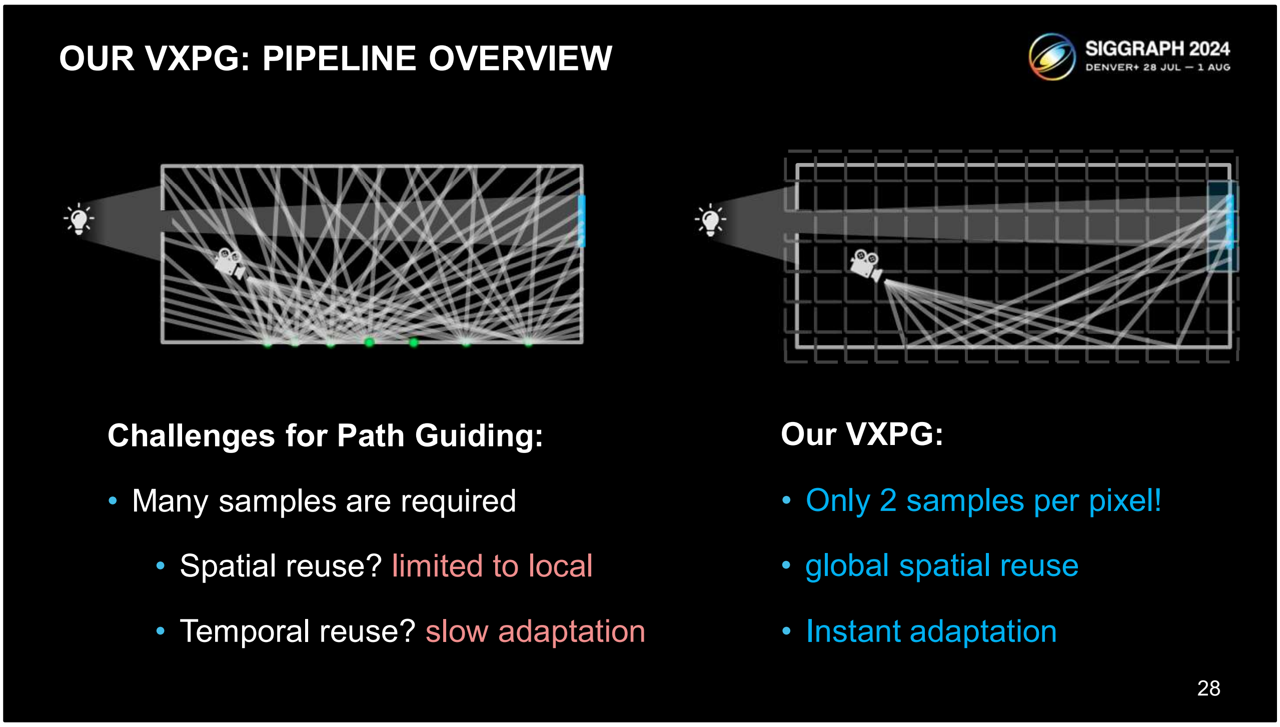
In practice, we have numerous pixels,
[click]
so hopefully most related voxels can get populated.



Finally, we can draw another guided sample,
trying to connect the shading points with the populated voxels.



Finally, we can draw another guided sample,
trying to connect the shading points with the populated voxels.
[skip]

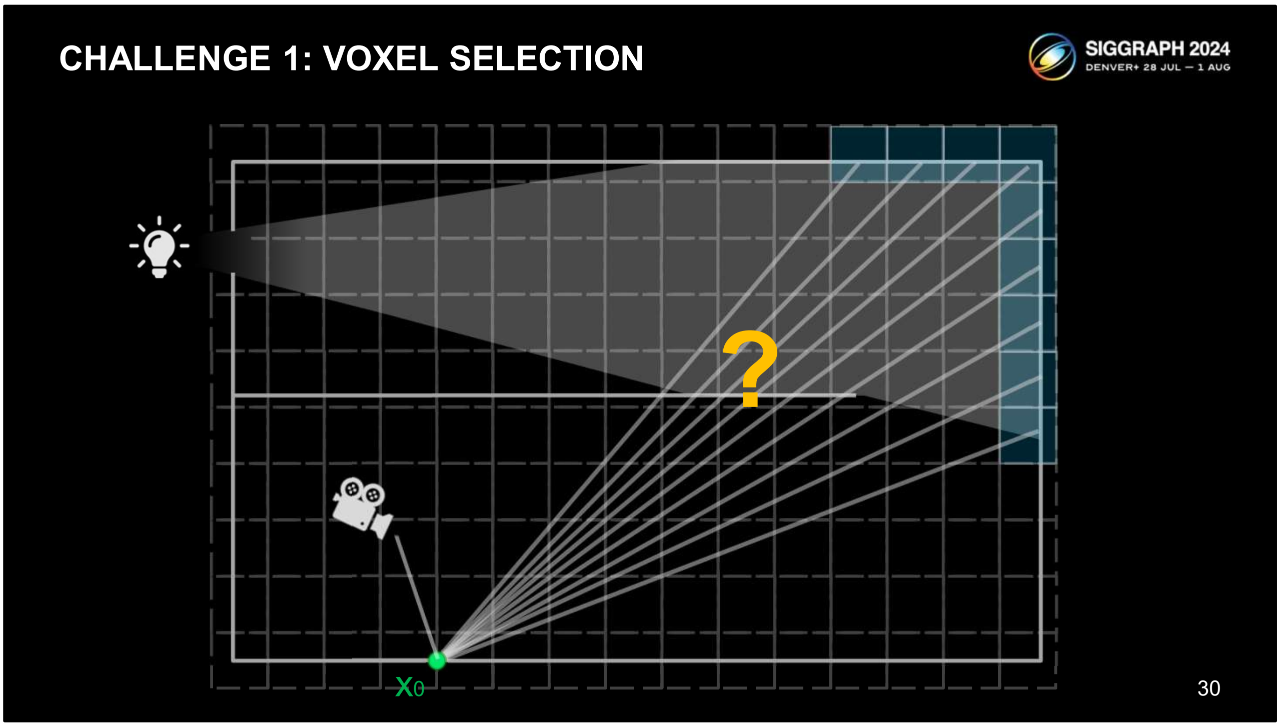


[05:00]
With VXPG, we have addressed the posed challenges in a novel way.

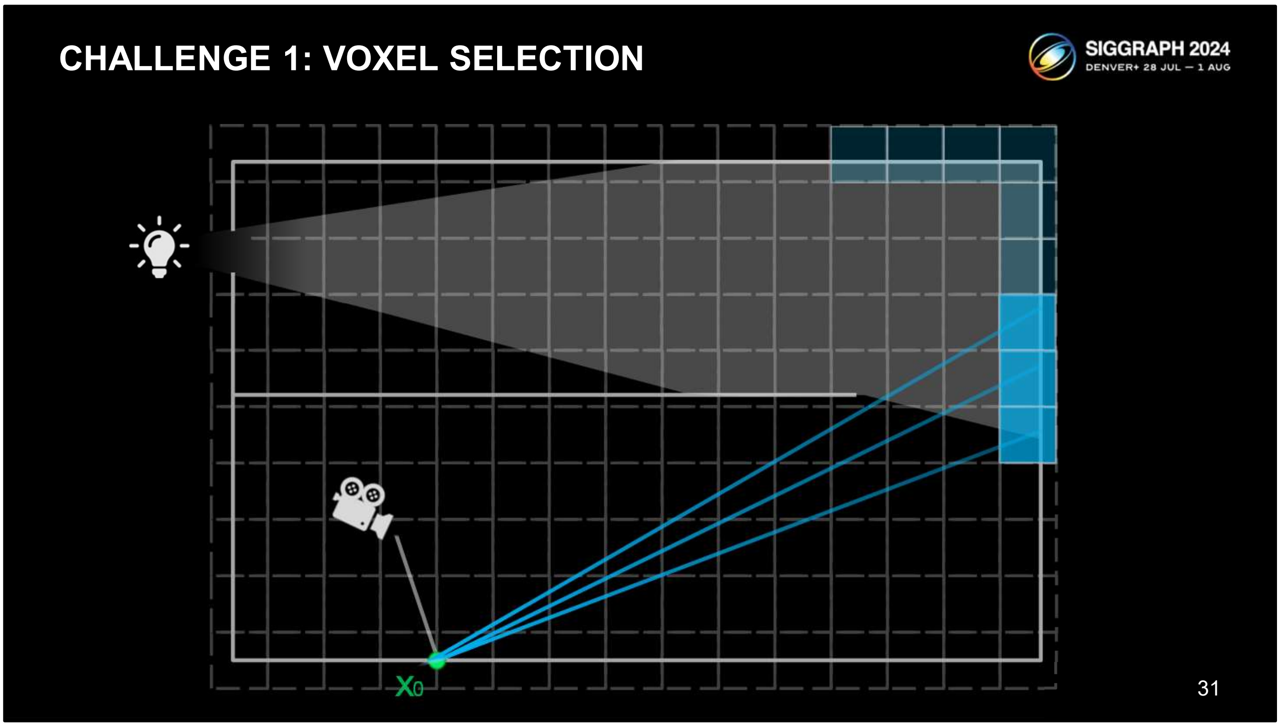
We promote global spatial reuse, which is no longer limited to nearby regions, [click]
and achieve instant adaptation without the need for temporal reuse. [click]

Most importantly, our entire pipeline uses only two samples per pixel, [click] making real-time usage possible.

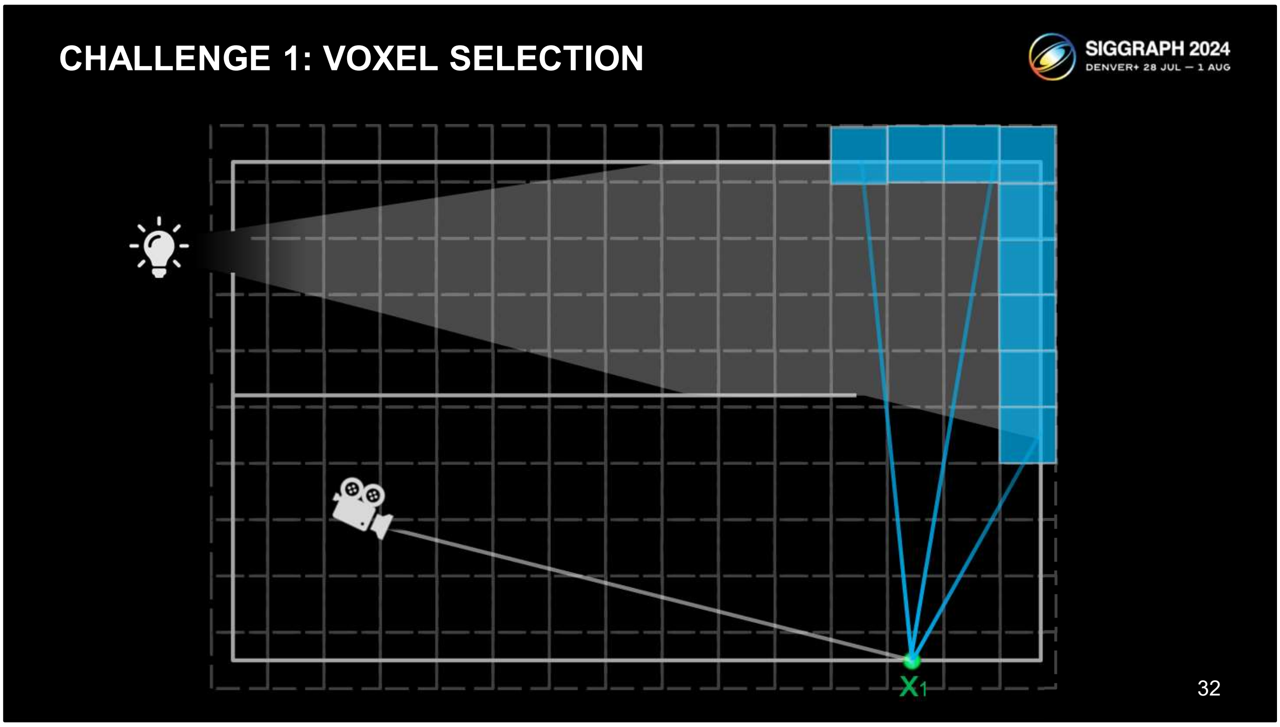
[05:20]
However, this new approach also introduces new challenges



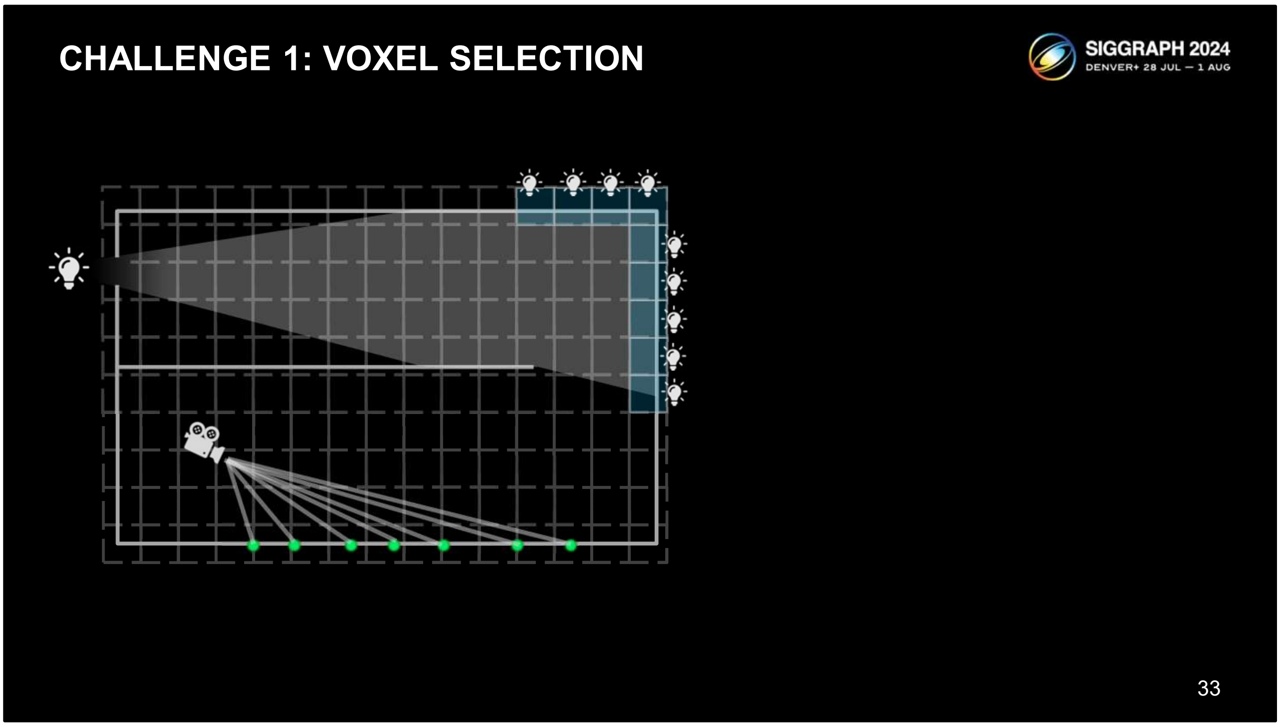
After populating the structure, we need to select one voxel to connect with for the shading point



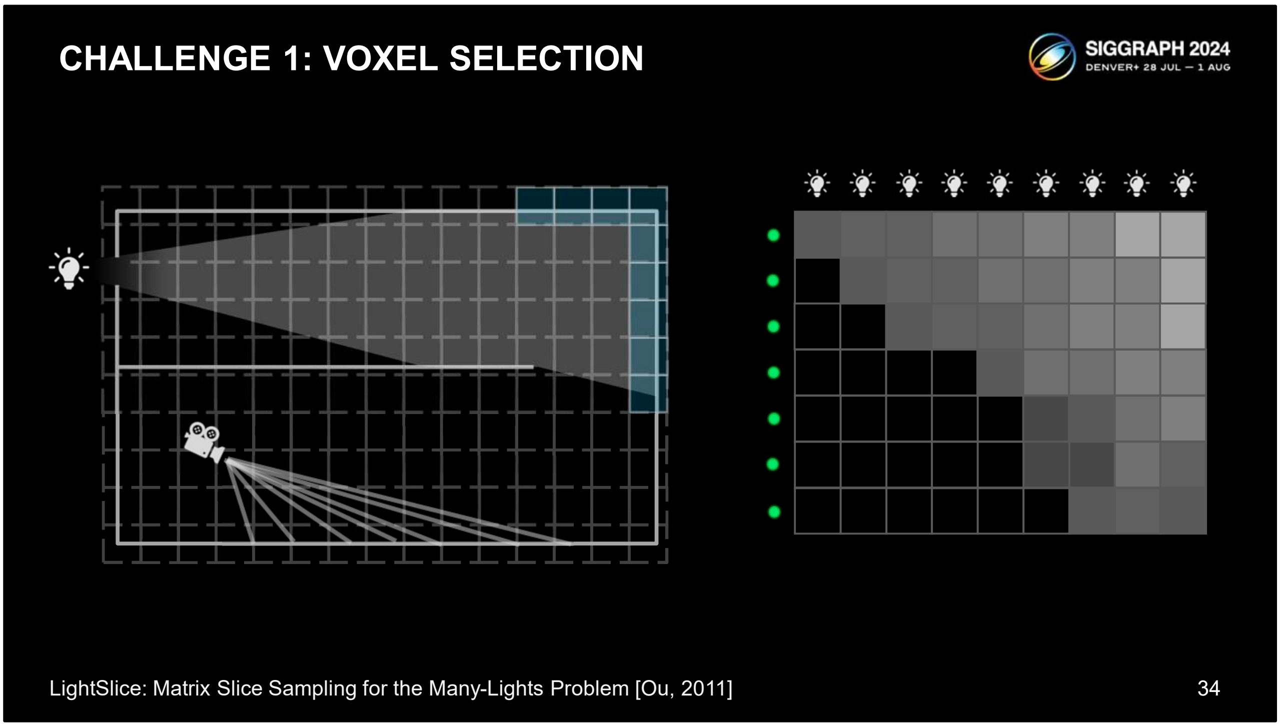
For x0, we might prefer the rightmost 3 voxels, as the rest are occluded.



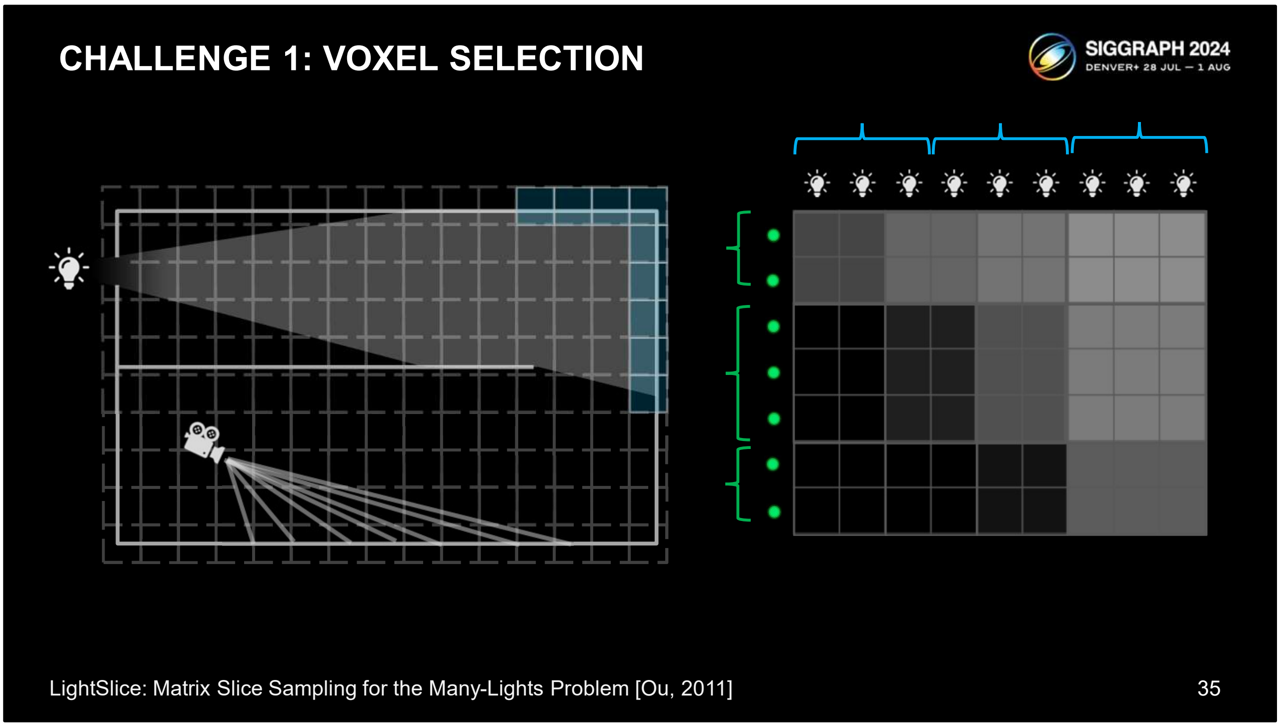
But for x1, all voxels are reasonable choices.
Thus, different shading points may prefer different voxels due to BRDF and visibility differences.



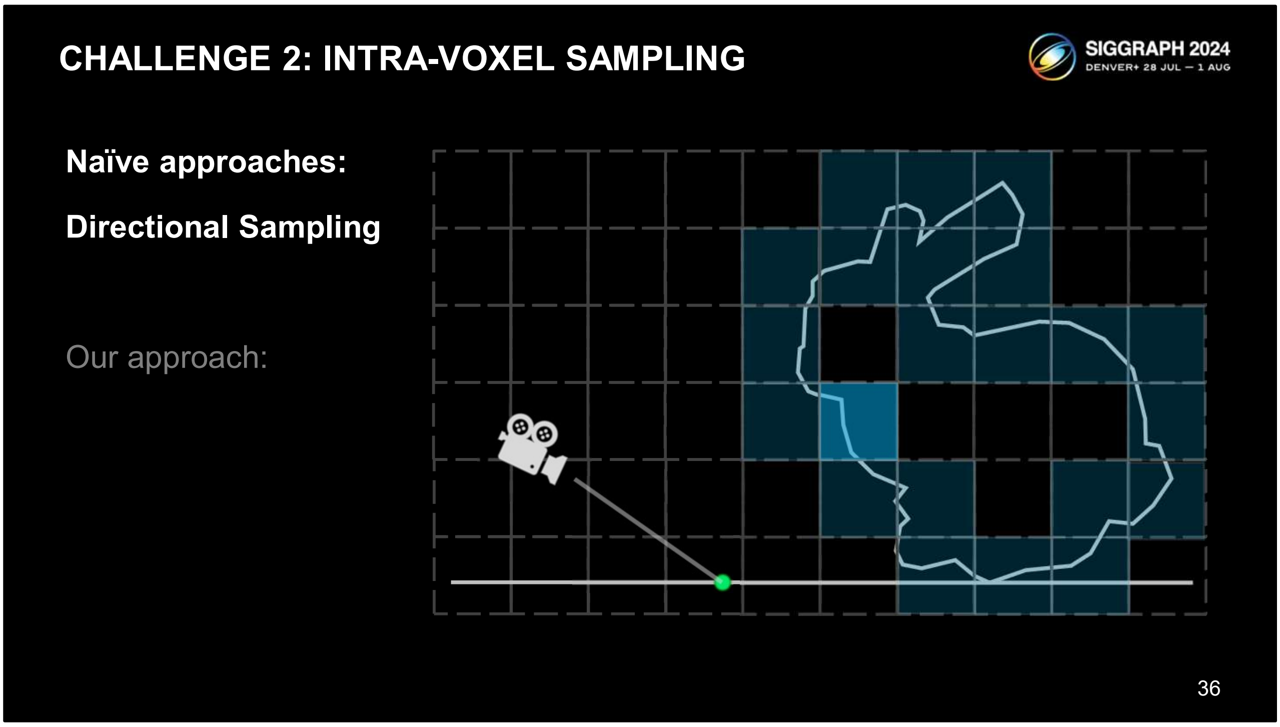
By regarding all populated voxels as individual lights,
[click]
We can realize that this is essentially a many-light sampling problem.



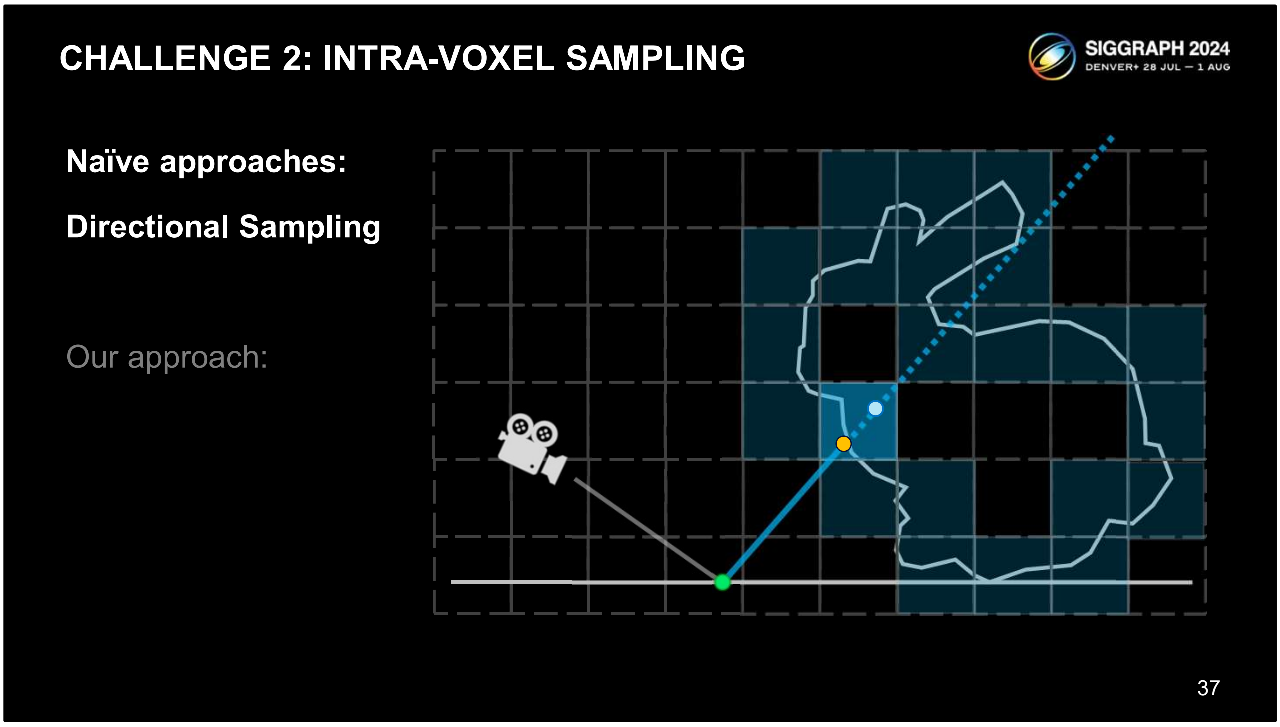
We adapt the light-slice technique for real-time use with significant simplifications. This method tabulates mutual contributions between lights and shading points.



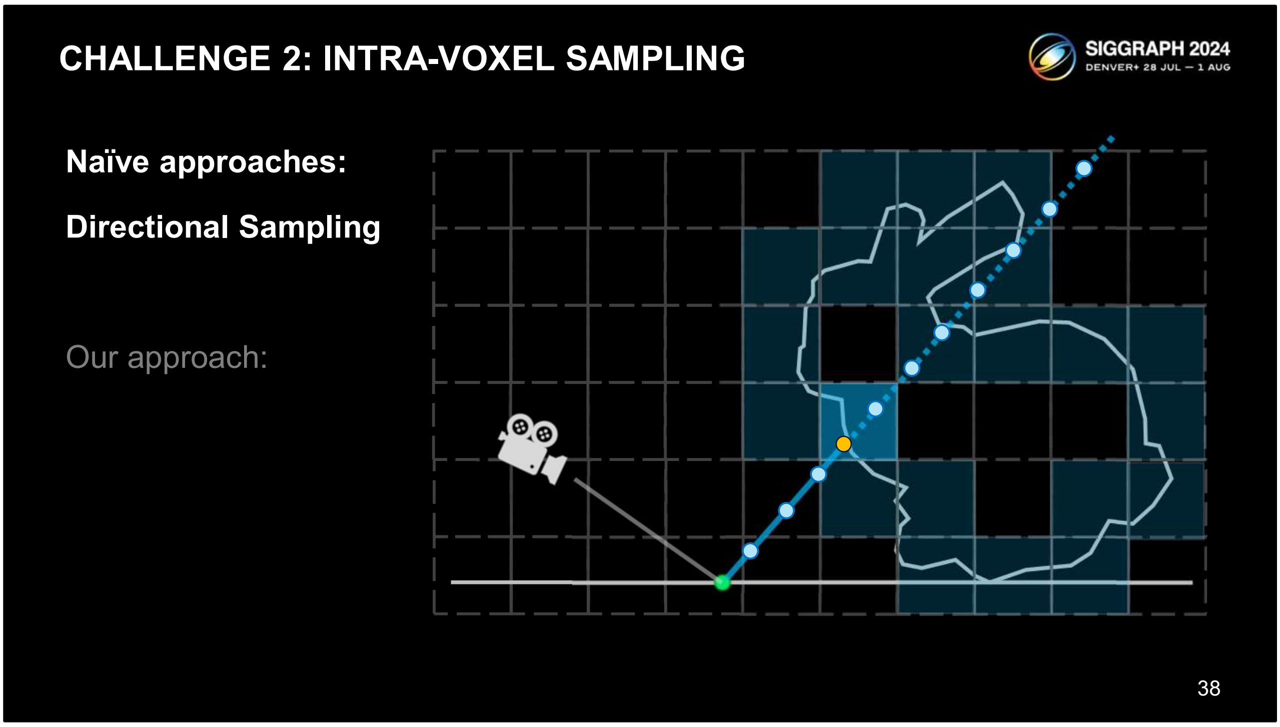
By wisely clustering both, we can sparsely evaluate a low-resolution table. This approximation can significantly improve sampling quality with minimal overhead.



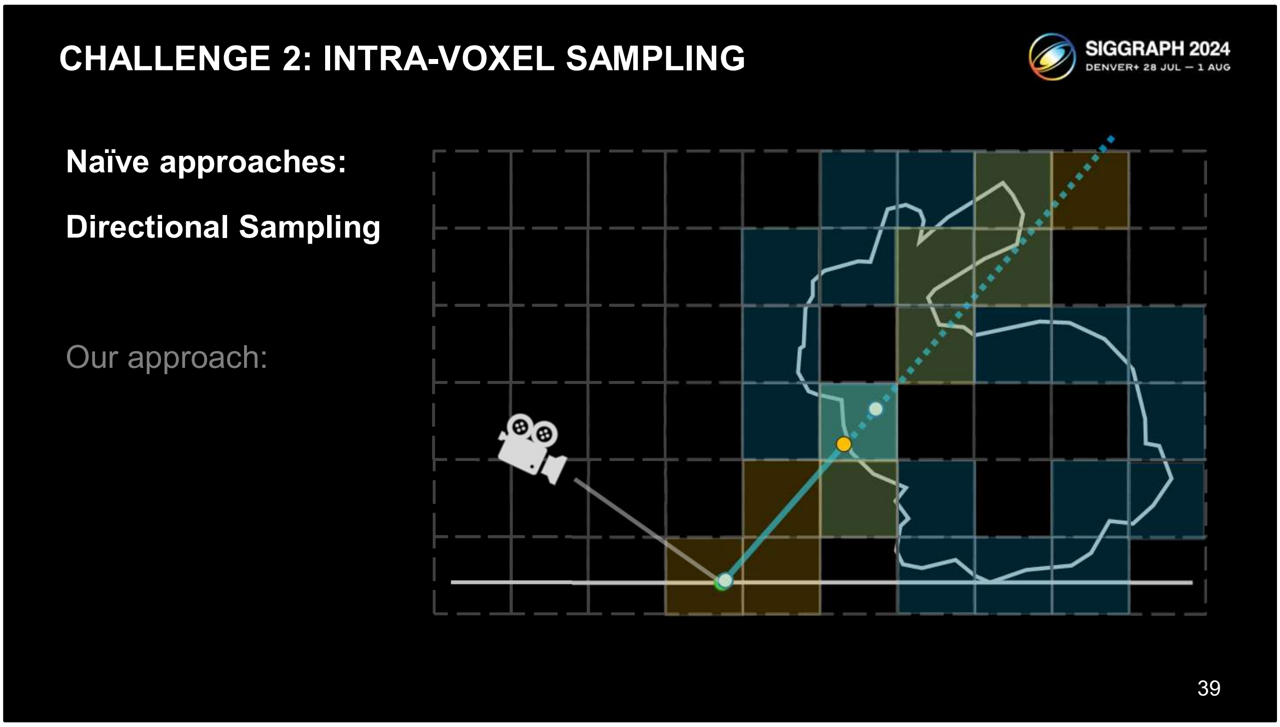
[06: 20]
Then, once a voxel is selected,
[click]
we need to perform the connection.
[click]
We first look at a naïve approach which is imperfect,
And describe our solution later.



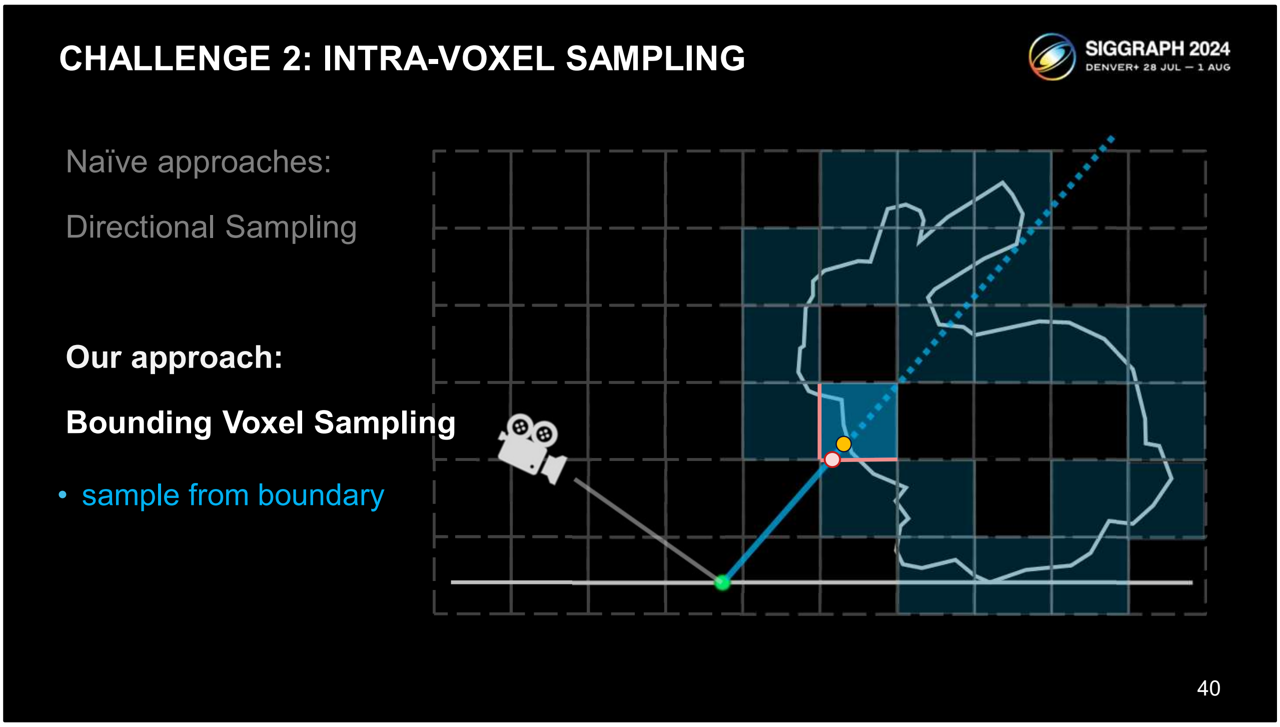
In the naïve approach,
[click] We first sample a point inside the selected voxel;
[click] Then we shoot a ray toward it,
[click] and find the first intersection point as the next vertex.



But the problem is that
[click]
all points along the ray will be mapped to the same direction

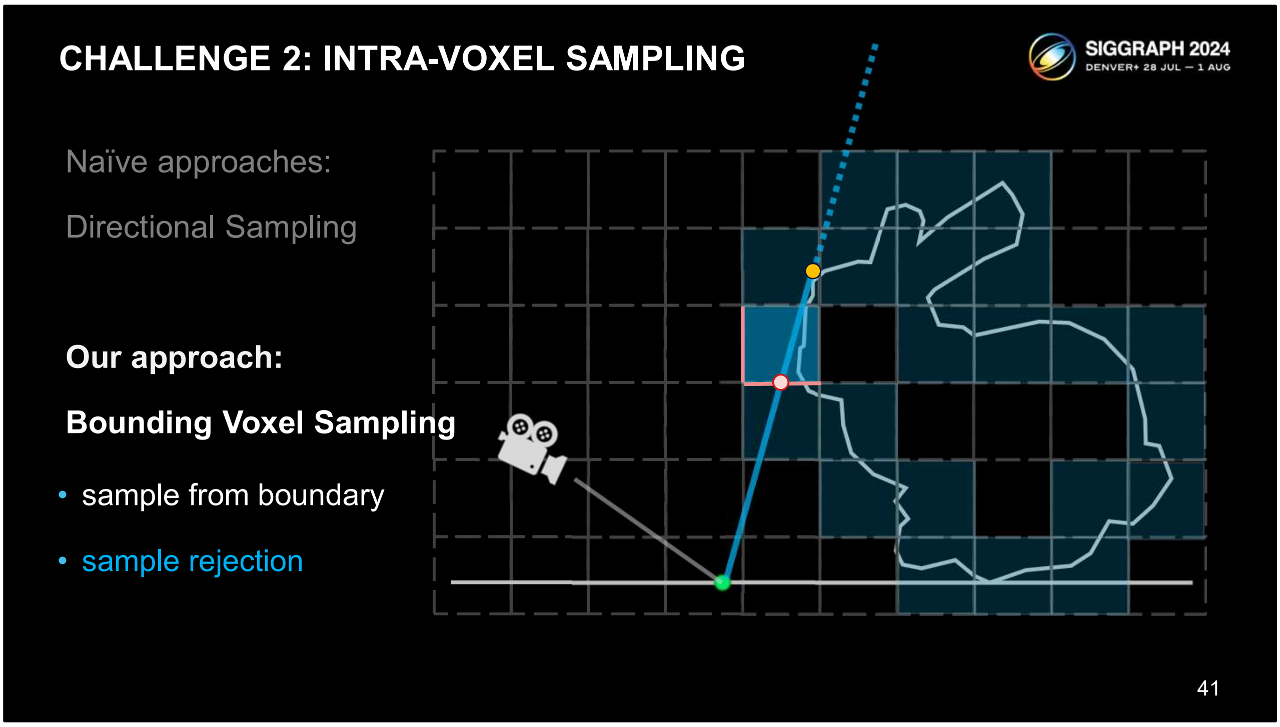


This means we must integrate over the entire ray to compute the PDF,
[click]
making it computationally expensive

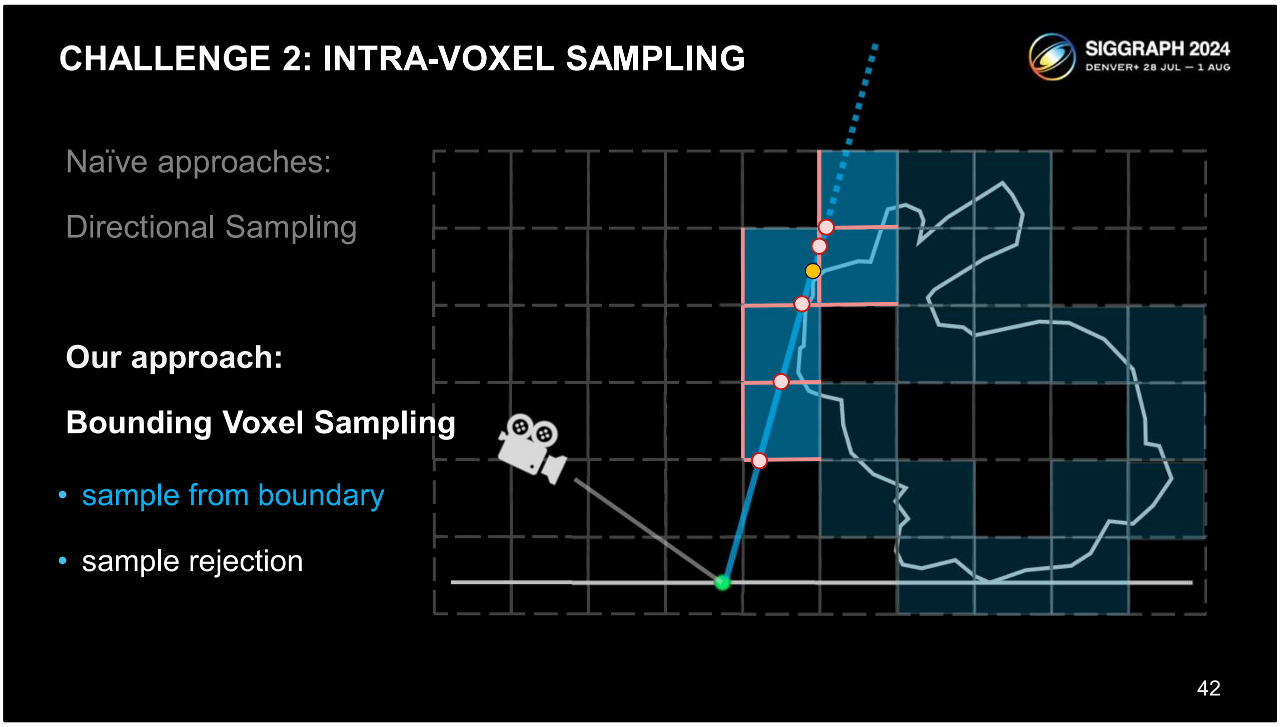


Instead, we propose bounding voxel sampling.
We first sample a point on the voxel boundary facing our shading point.
[click] shoot a ray toward it,
[click] and still use the first intersection.

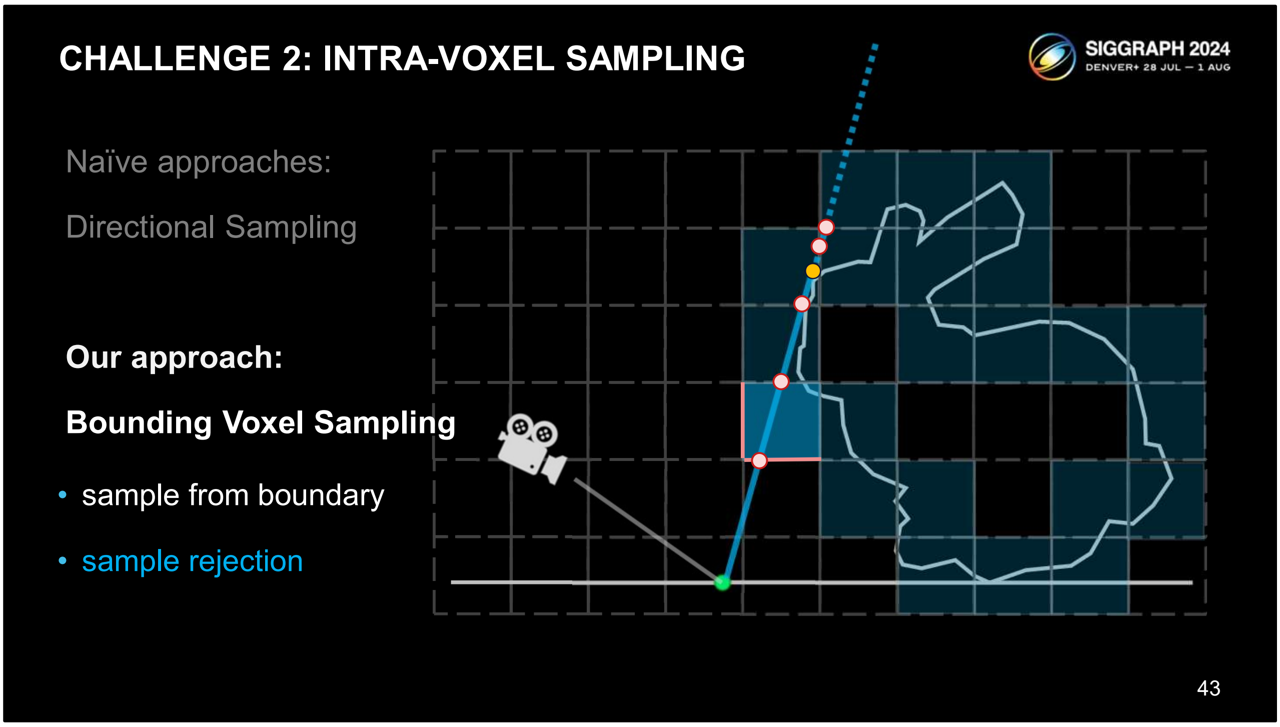
If this intersection is within the voxel, we accept it.



Otherwise, we will directly discard it.

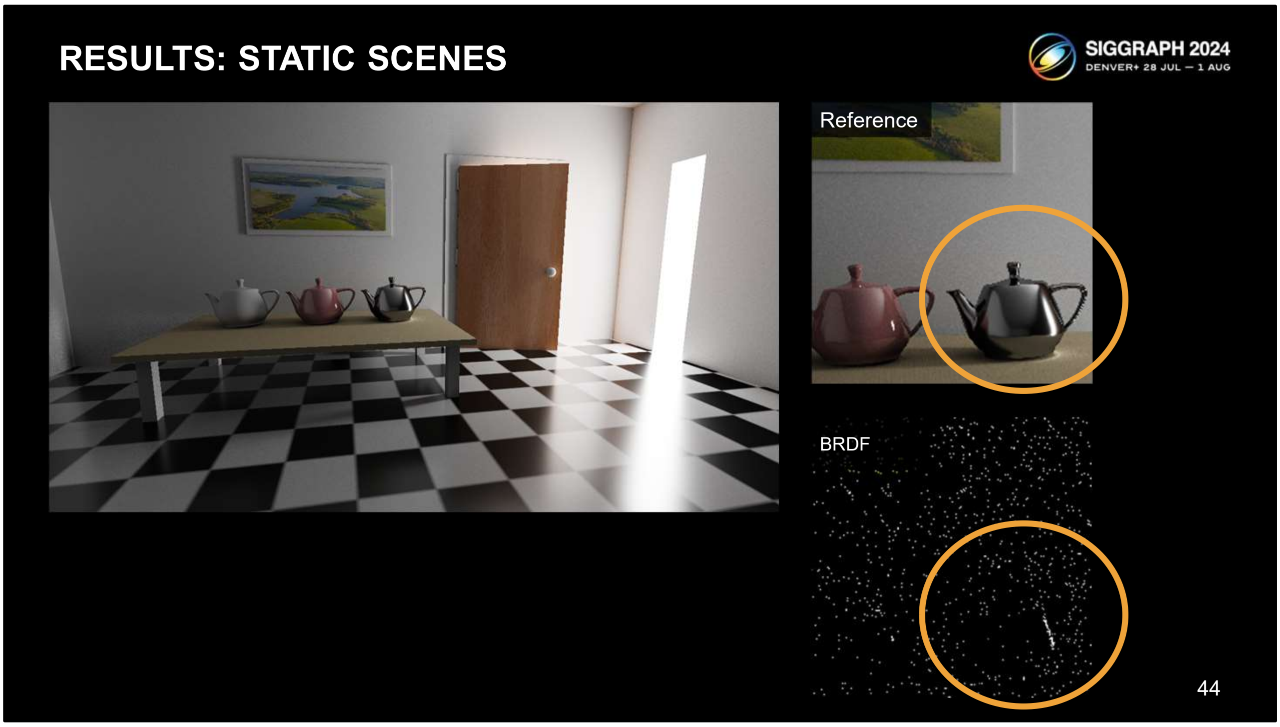


Notice that, as we sample from the closer boundary,
[click]
Each voxel has at most one point that can be mapped to the direction.

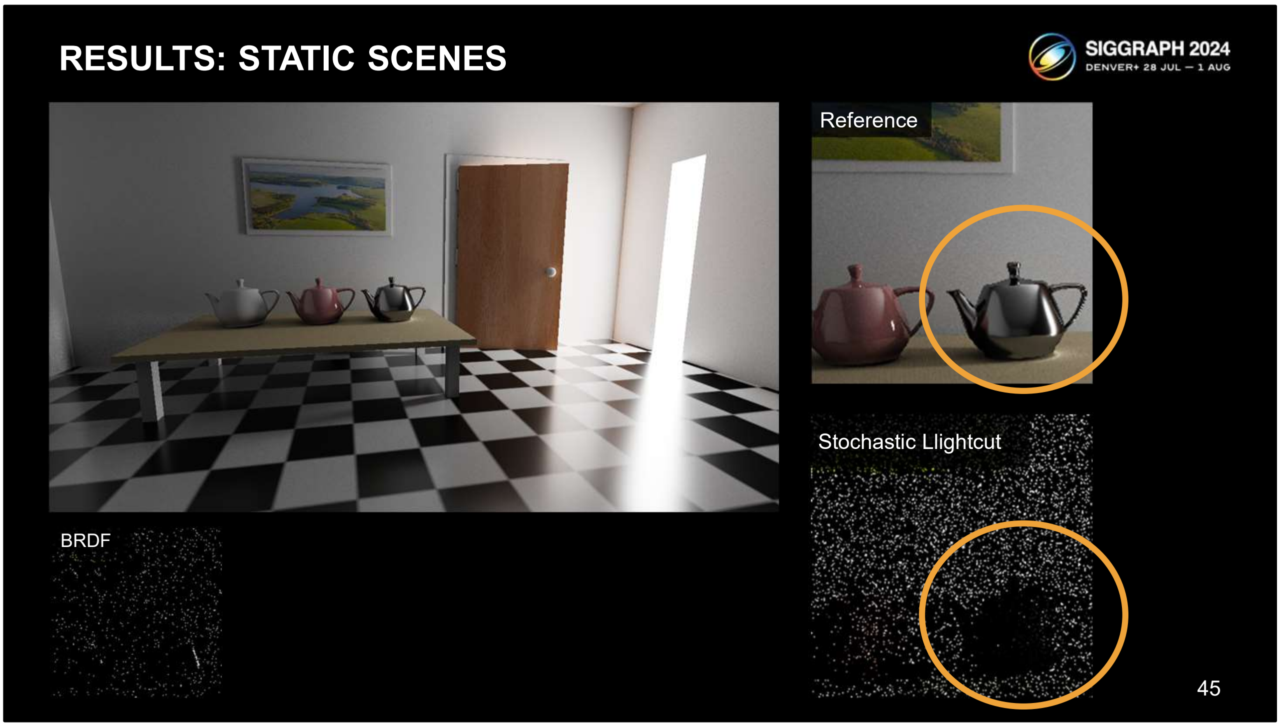


And as we use sample rejection,
any samples from voxels [click] that do not enclose the next vertex [click] will be rejected[click],

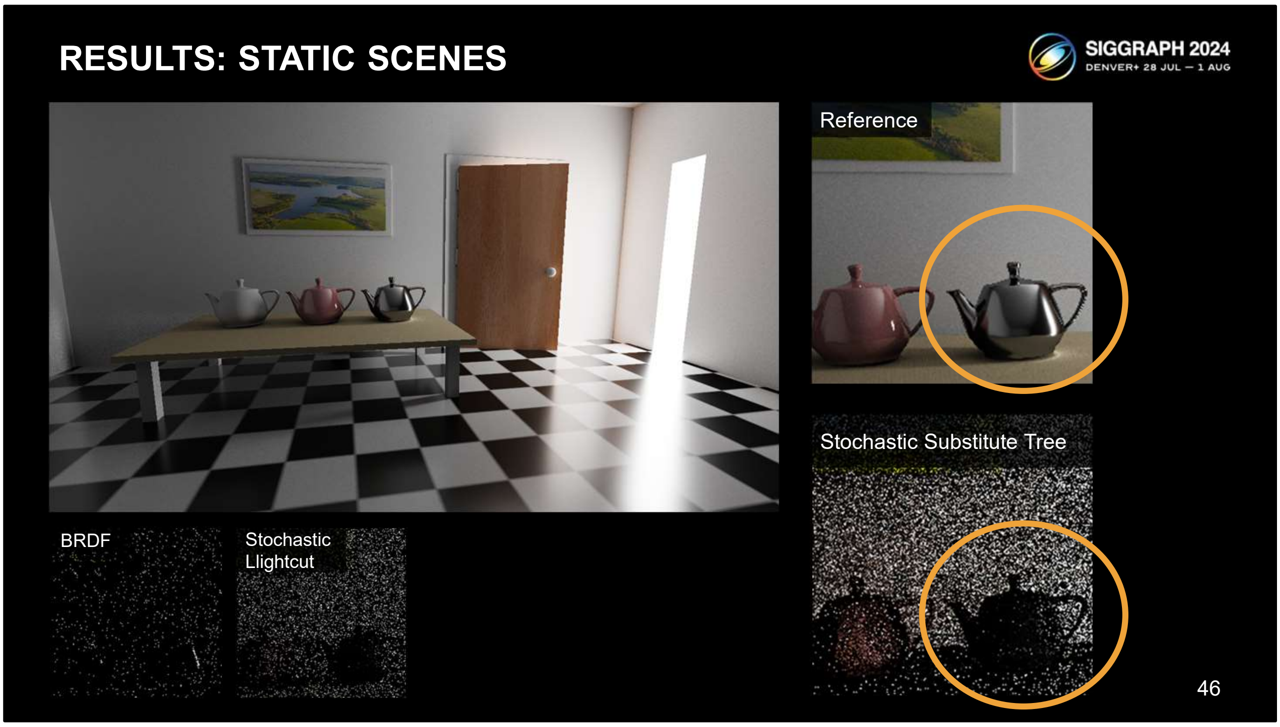
Thus, it leaves a one-to-one mapping between the sample point and the next vertex,
making the PDF computation much easier.



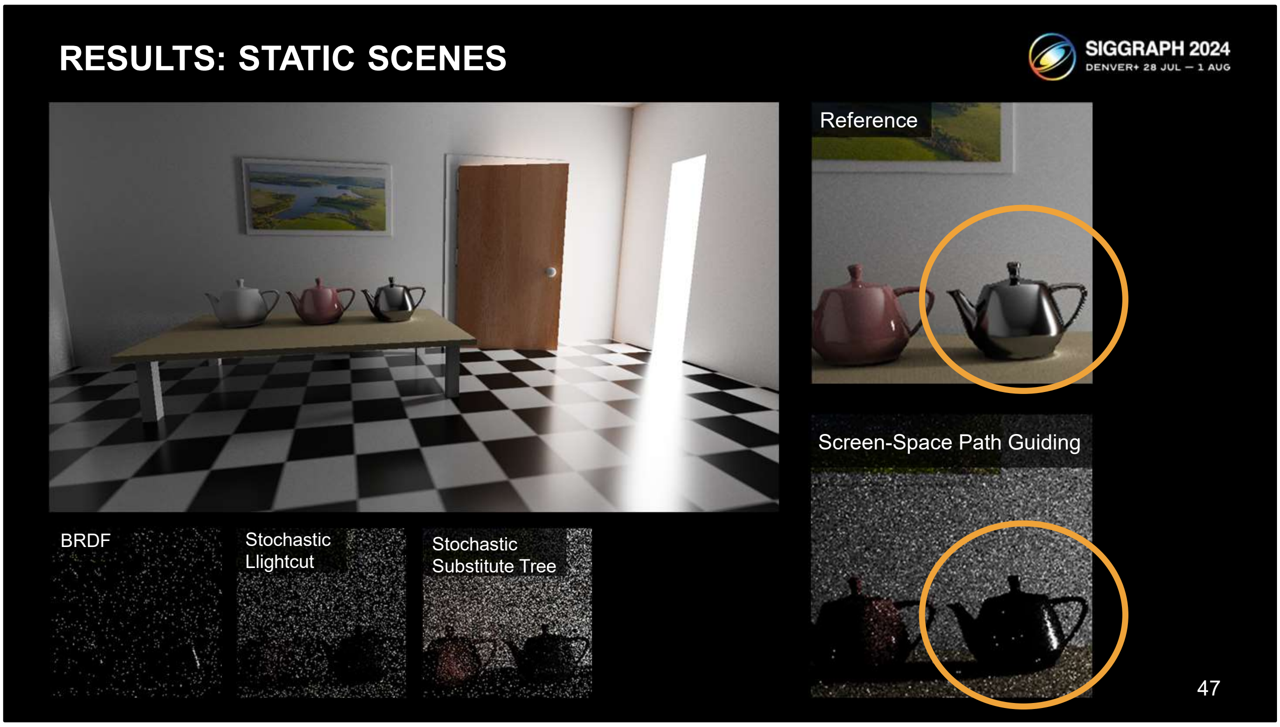
[07:50]
We validated our approach in static scenes by comparing it with BRDF importance sampling



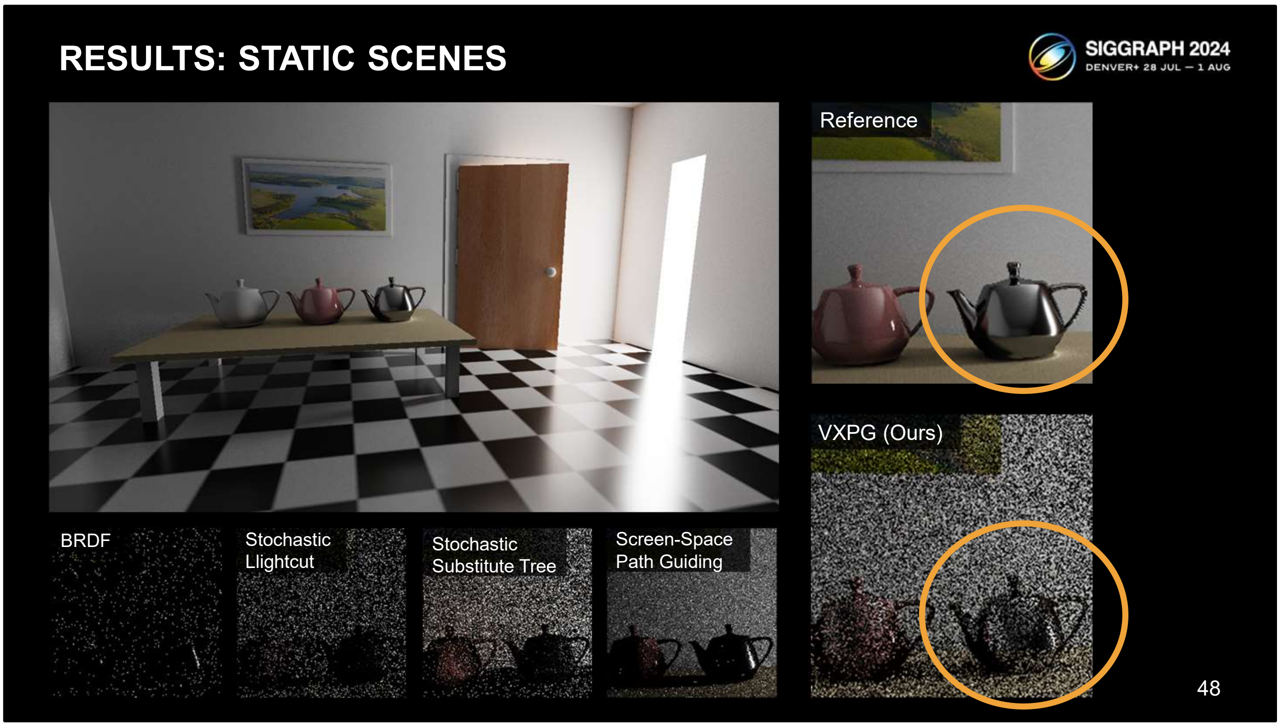
both unbiased



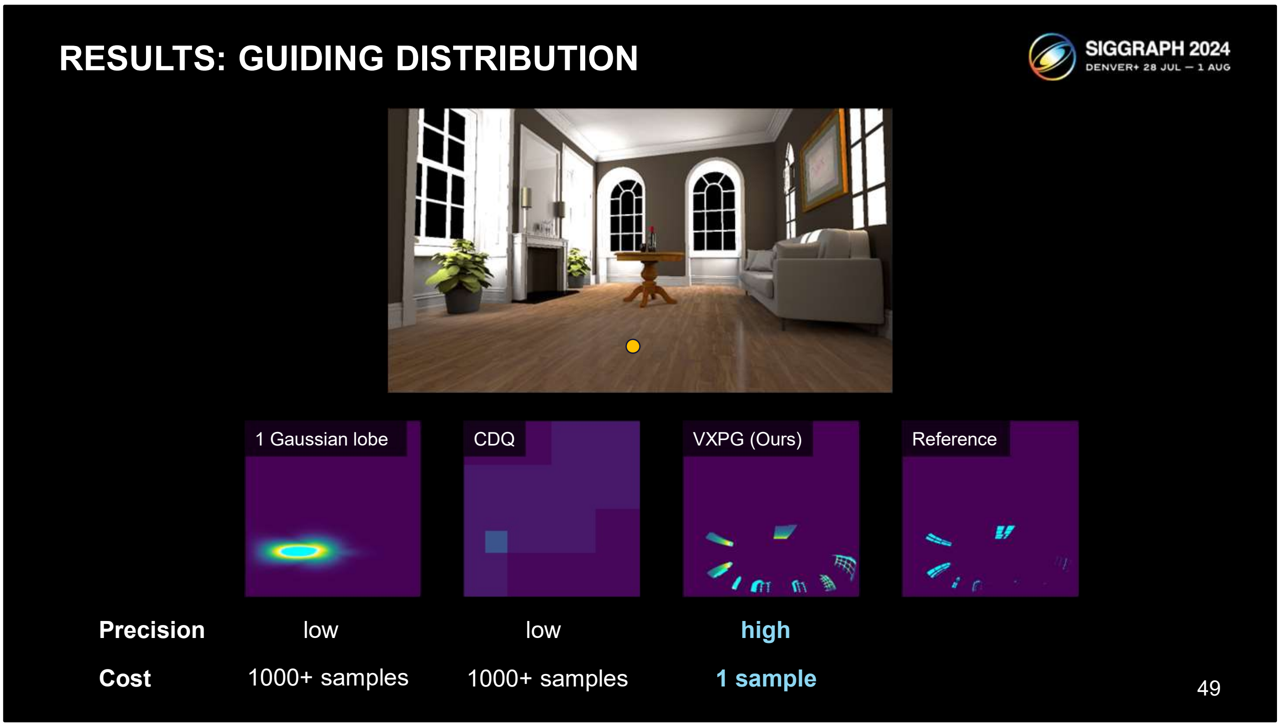
and biased virtual point light methods



and a previous real-time path guiding method that relies heavily on temporal reuse.



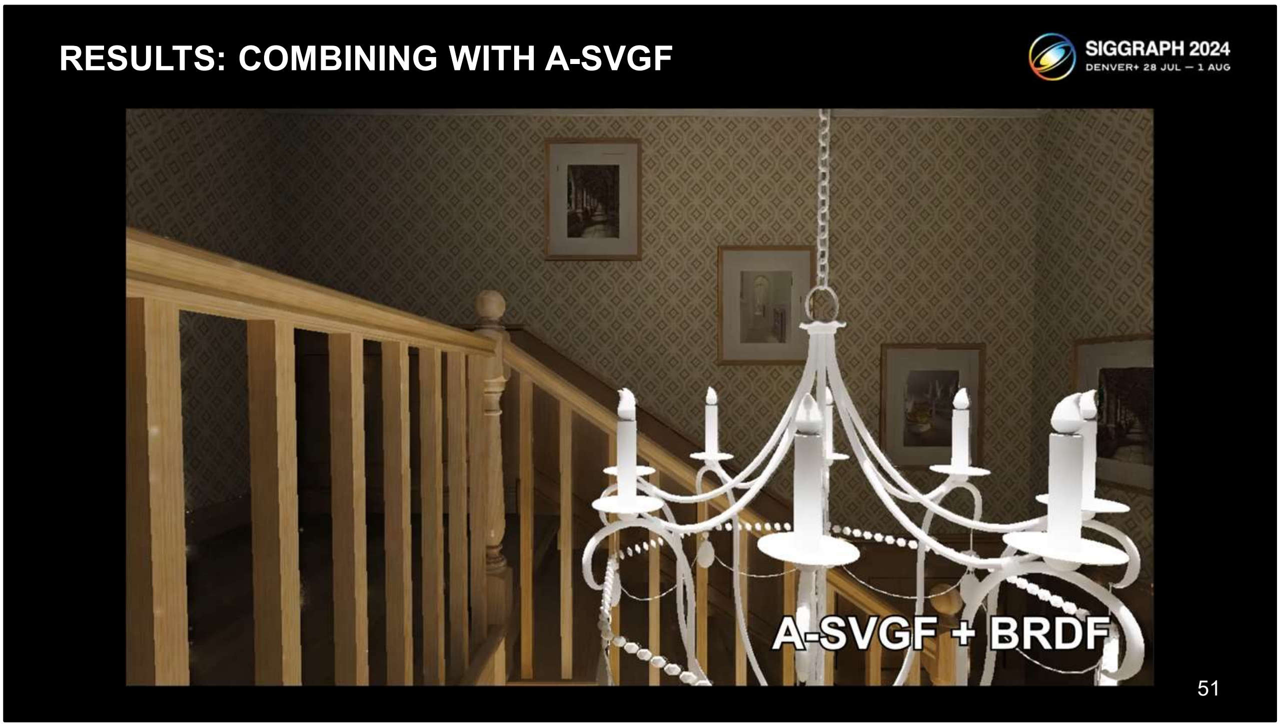
Our experiments show that VXPG achieves significantly better results without temporal reuse. (wait)



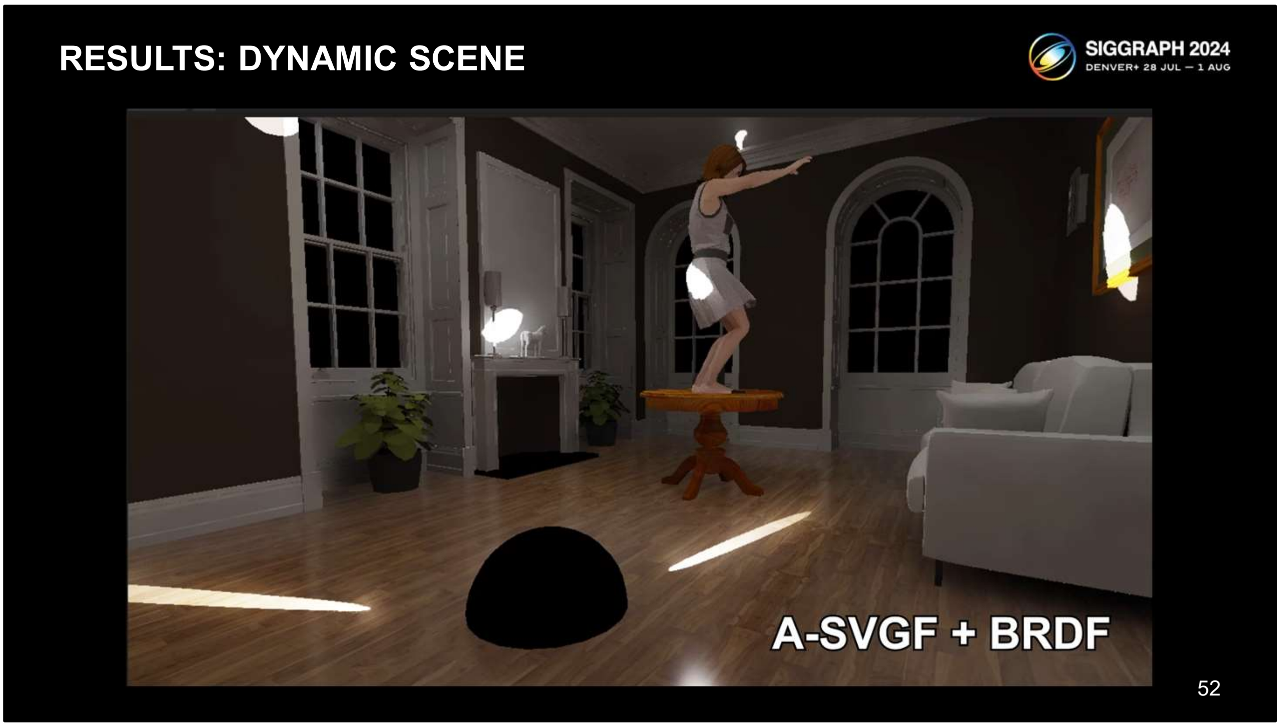
We also compare our learned distribution, with existing real-time path-guiding methods.
Their representations are low-resolution,
and cannot fit good distributions to challenging lighting,
even with thousands of samples.

By sharing samples across all shading points, our method can learn challenging distributions with only 1 spp.

[click]
Our approach can be effectively combined with ReSTIR, which relies on temporal reuse.
By combining ReSTIR with VXPG, adaptation becomes faster, and the stable quality also gets improved.



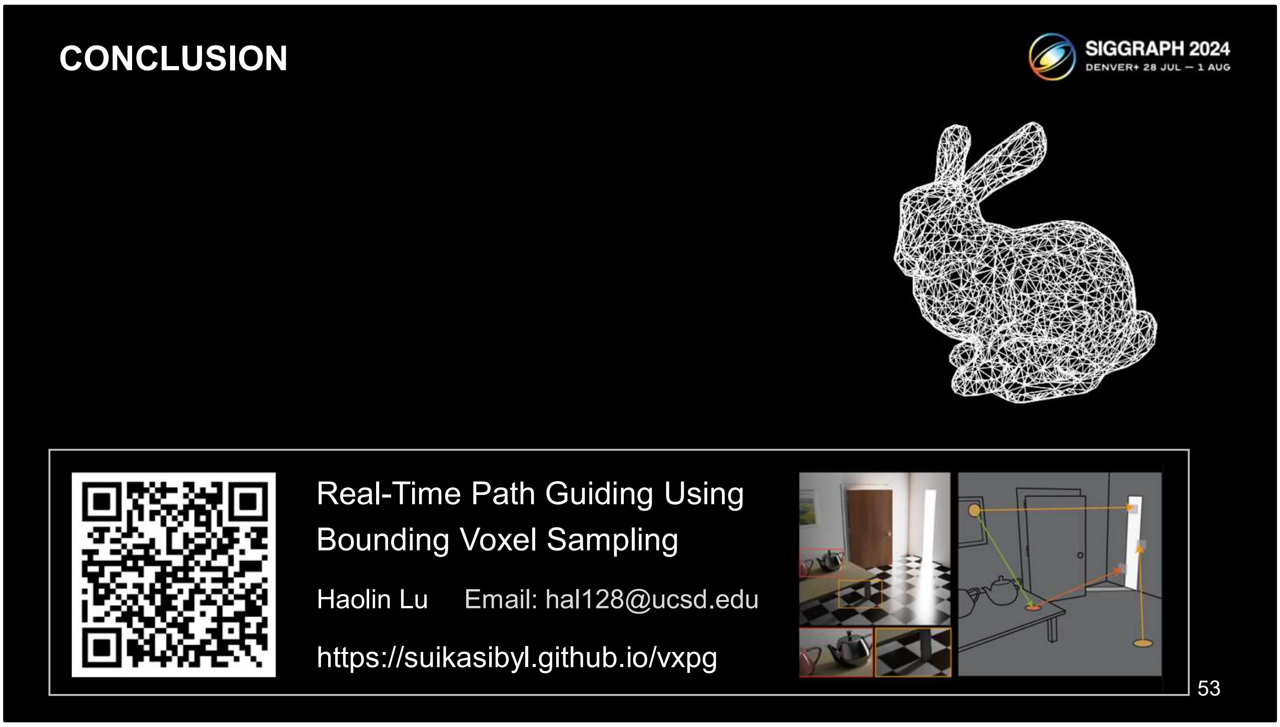
[click]
Denoisers such as A-SVGF also benefit from combining with VXPG.
VXPG can reduce detail loss and color distortion and, crucially,
help eliminate temporal artifacts like disocclusion



[click]
Additionally, since our method doesn't rely on temporal reuse,
it can naturally handle dynamic scenes.

To show its effectiveness, we present a challenging case with multiple fast-moving regions.
In this scenario, A-SVGF and unbiased ReSTIR-GI will all fail, but VXPG provides reasonable results.

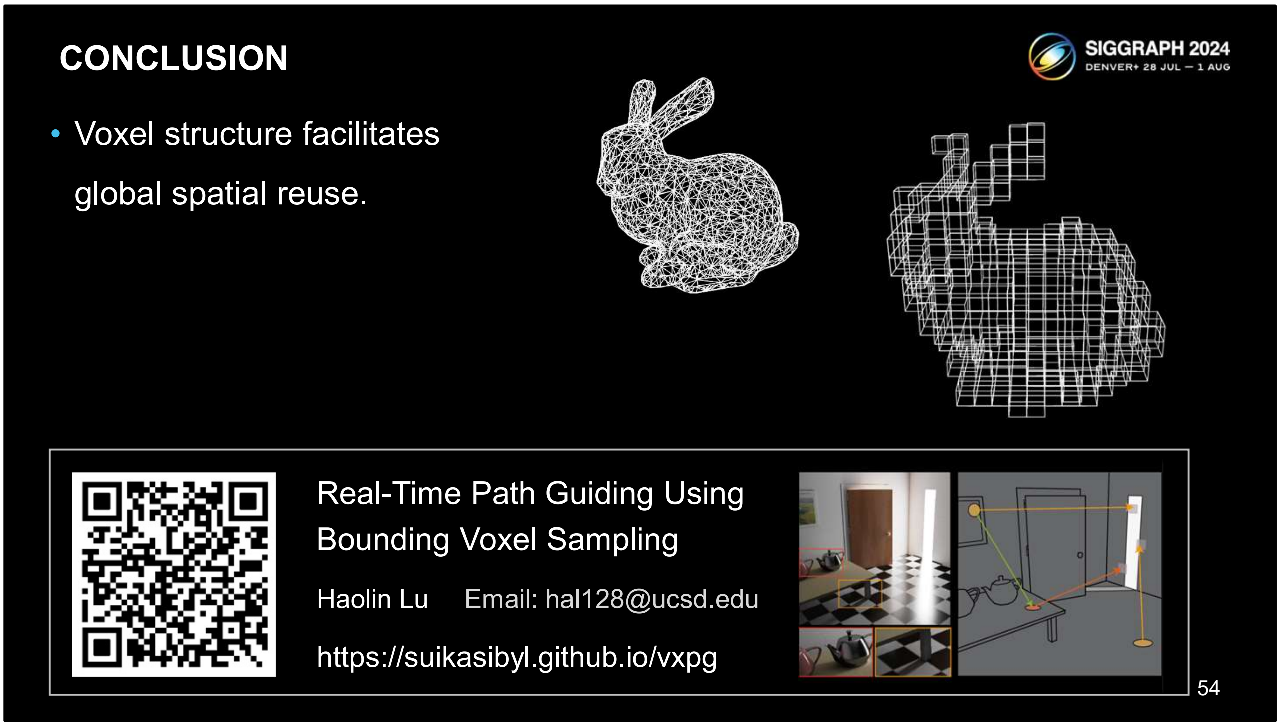
[9:30]



To summarize, [click] we propose using a voxel structure for path guiding, which allows for global spatial reuse and real-time reconstruction.

We implement a real-time variation of Lightslice for high-quality voxel selection. And introduce bounding voxel sampling to find the next path vertex.

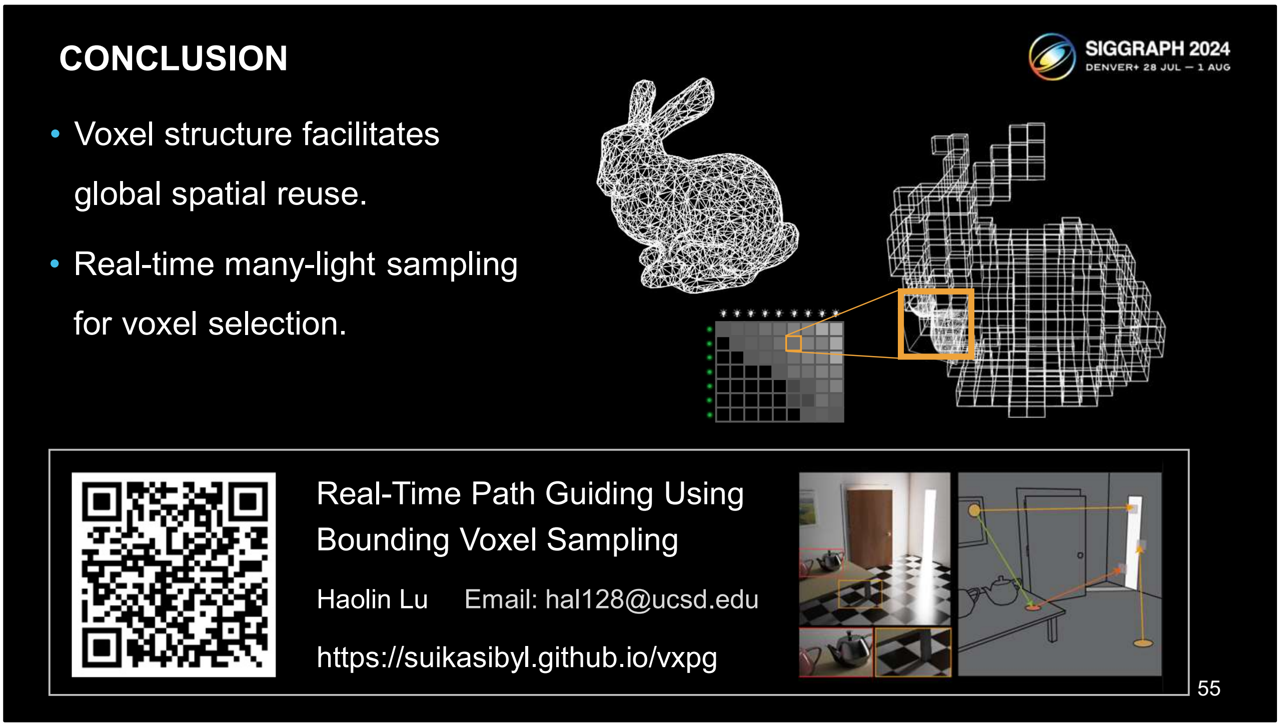
These components together enable real-time path-guiding.
Thank you for listening.



To summarize, [click] we propose using a voxel structure for path guiding, which allows for global spatial reuse and real-time reconstruction.

We implement a real-time variation of Lightslice for high-quality voxel selection. And introduce bounding voxel sampling to find the next path vertex.

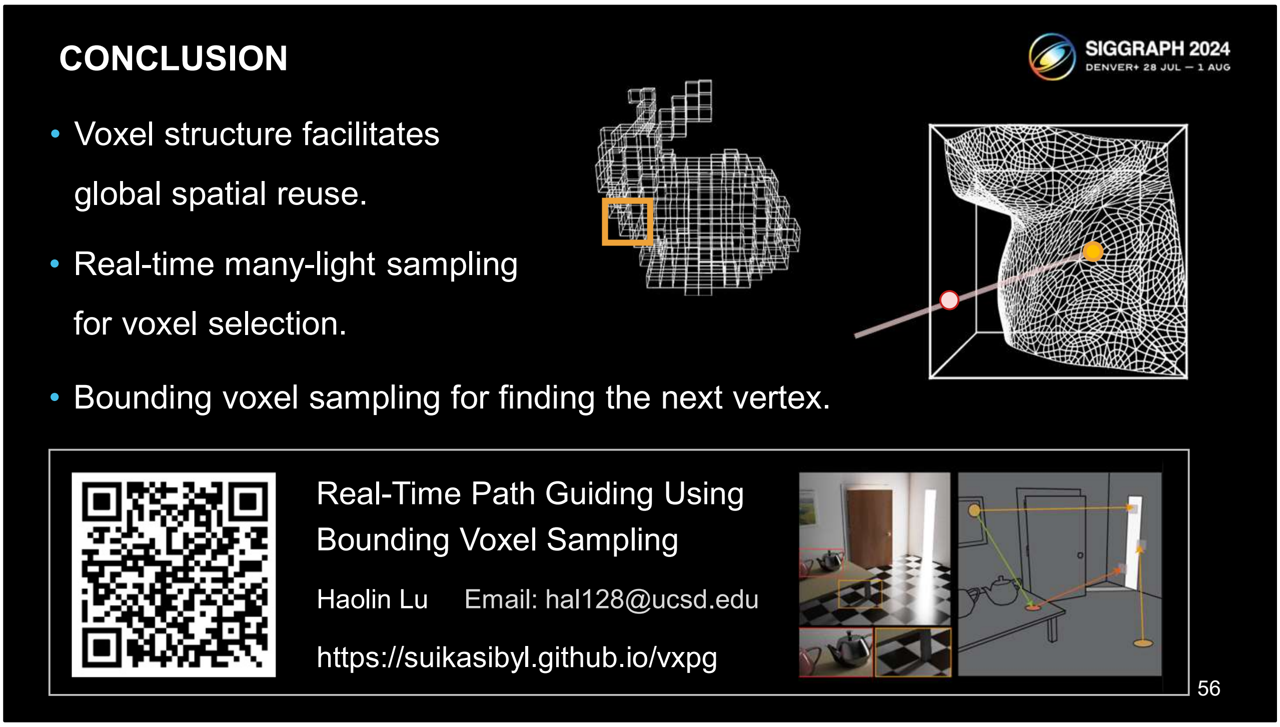
These components together enable real-time path-guiding.
Thank you for listening.



To summarize, [click] we propose using a voxel structure for path guiding, which allows for global spatial reuse and real-time reconstruction.

We implement a real-time variation of Lightslice for high-quality voxel selection. And introduce bounding voxel sampling to find the next path vertex.

These components together enable real-time path-guiding.
Thank you for listening.



To summarize, [click] we propose using a voxel structure for path guiding, which allows for global spatial reuse and real-time reconstruction.

We implement a real-time variation of Lightslice for high-quality voxel selection. And introduce bounding voxel sampling to find the next path vertex.

These components together enable real-time path-guiding.
Thanks for listening.