(Optimizing) Realistic Rendering with Many-Light Methods

Improved VPL Distribution

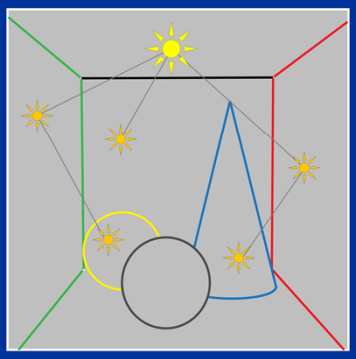
(part of the "Handling difficult light paths" section)

Jaroslav Křivánek
Charles University in Prague

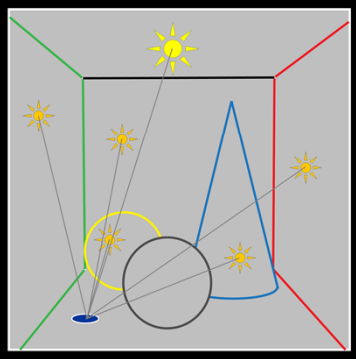
In this part of the course, I will discuss various approaches for generating VPLs where they are most needed for a given camera view.

VPL rendering

1. Distribute VPLs

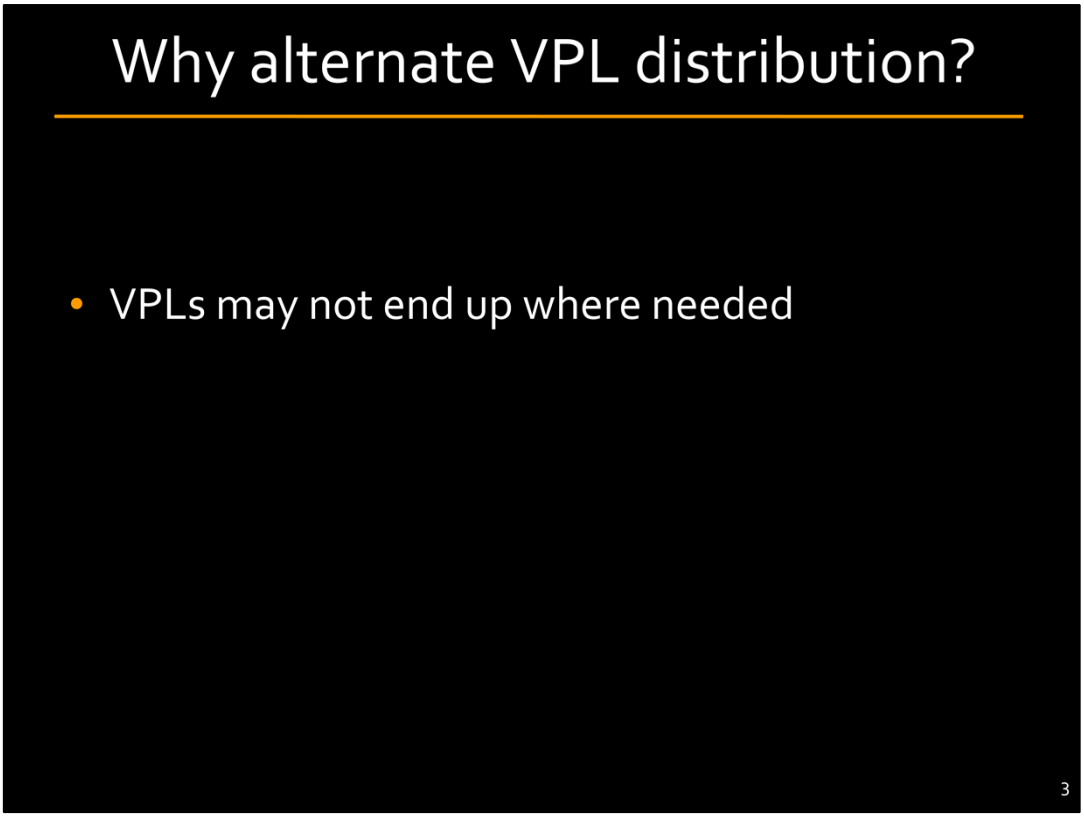


2. Render with VPLs

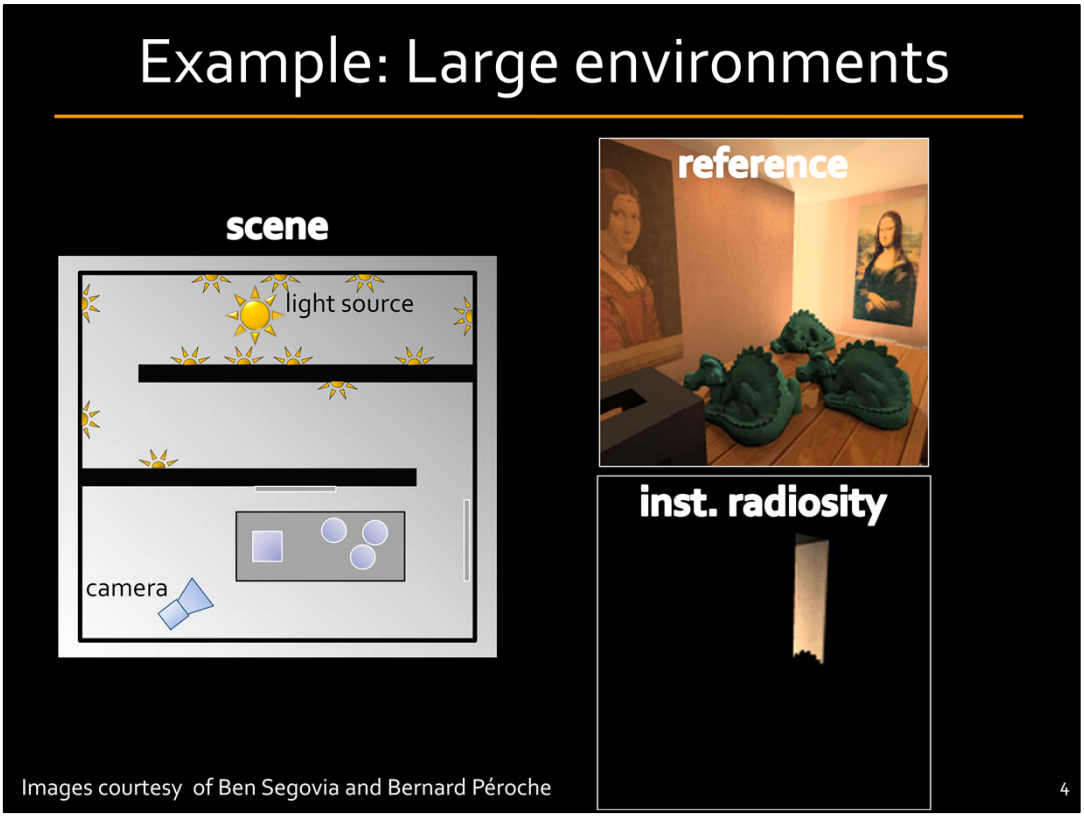


2

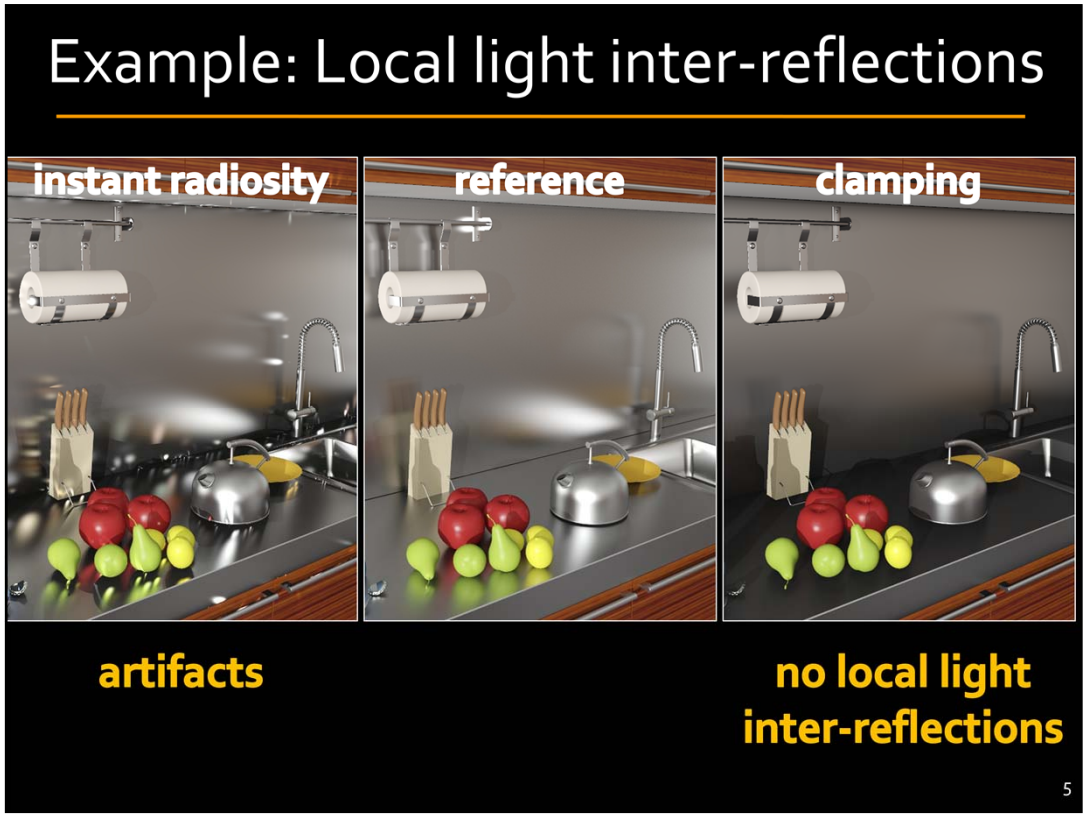
Let me start by reviewing the classic many-lights rendering algorithm, Instant Radiosity.
In the first step, the VPLs are distributed on scene surfaces by tracing particles from light sources.
In the second step, the image is rendered by summing contributions from all the VPLs.
In this part of the course, I will focus on various approaches to distributing the VPLs.



The need to develop alternate VPL distribution approaches follow from the fact that with the basic VPL tracing algorithm, the VPLs may end up in regions where they do not contribute significantly to the image.



This will be the case especially in large environments where the camera is looking at a small portion of the scene.
In the example shown here, the light source is far from the portion of the scene seen by the camera. The usual VPL tracing algorithm will generate many VPLs that do not contribute to the image at all.



In addition, VPL distributions generated by the basic VPL tracing algorithm cannot be used to render local light inter-reflections.

Here is an example. The number of VPLs along the edges is insufficient to render the local inter-reflections, resulting in artifacts in the form of light splotches.
The usual way of dealing with these artifacts the *clamping* that we discussed previously, where we clamp the contribution of a single VPL to a prescribed maximum value.
But this selective energy removal can severely change material appearance, as you can see in the image on the right.
A better solution would be to ensure that more VPLs are generated in the visible areas along the edges.

Purpose & approach

- Purpose
  - Ensure VPLs end up where needed
- Approaches
  - Rejection of unimportant VPLs
  - Metropolis sampling for VPL distribution
  - Distribute VPLs by tracing paths from the camera

6

So, the goal is to get the VPLs where they are most needed. A number of approaches have proposed for this and I will discuss the following three in the remaining part of my presentation.
The simplest approach is to apply rejection sampling, where VPLs that do not significantly contribute to the image are rejected.
Second, we can use a more advanced sampling algorithm such as Metropolis sampling.
And finally, we can distribute the VPL by tracing paths from the camera instead of from light sources.

Rejection of unimportant VPLs

Rejection of unimportant VPLs

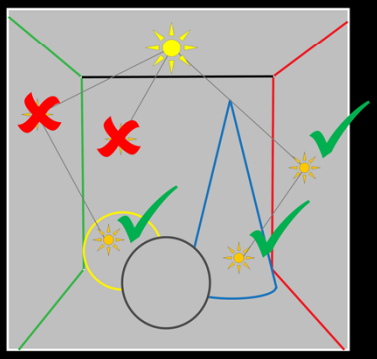
- Autodesk 360 Rendering
  - Covered by Adam later in the course
- [Georgiev et al., EG 2010]
  - Covered on the following slides (courtesy of Iliyan Georgiev)
- Good for large environments but not for local interactions

8

A form of rejection sampling is used for VPL distribution in the Autodesk 360 Rendering solution that will be later described by Adam Arbree.
So I will only briefly mention the approach presented by Georgiev et al. in their EG 2010 short paper.

VPL rejection – Idea

- Accept VPLs proportionately to their total image contribution
  - Reject some of those that contribute less than average

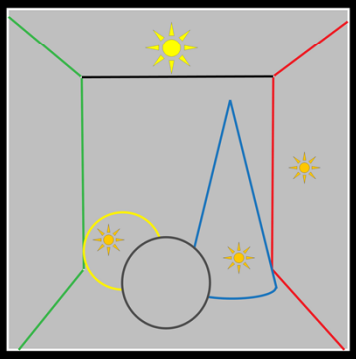


9

The main idea is very simple:
They use the exact same VPL tracing algorithm as in Instant Radiosity but they probabilistically reject the VPLs if their image contribution is less than average.

VPL rejection – Idea

- Accept VPLs proportionately to their total image contribution
  - Reject some of those that contribute less than average



VPL rejection – Algorithm

- Want VPLs with equal image contribution Φ_v
- For each VPL candidate i
  - Estimate total image contribution Φ_i
  - Accept w/ probability $p_i = \min\left\{\frac{\Phi_i}{\Phi_v} + \varepsilon, 1\right\}$

    (divide energy of an accepted VPL by p_i)

11

Estimating image contribution

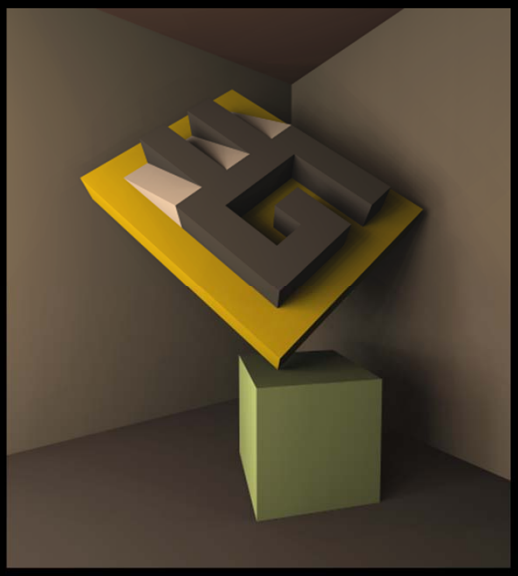
- No need to be accurate
- Estimating Φ_v (average VPL contribution)
  - Based on a few pilot VPLs
- Estimating Φ_i (contribution of VPL candidate i)
  - Contribution to only a few image pixels

12

How do we estimate the target "average" VPL contribution? A simple way to do it is to run a number of pilot VPLs and render a low-resolution image. Another possibility is to use information form the previous frame, if rendering an animation.
To estimate the image contribution of a candidate VPL, we simply render a "low-res" image by picking only a couple of pixels.
There's no need to be very accurate: It's no use to spend much time on estimating the VPL contributions because the algorithm will produce correct results no matter how accurately the VPL contribution is estimated.

VPL rejection – Results

Instant Radiosity



[Georgiev et al. 2010]
(7% acceptance)

VPL rejection – Conclusion

- Cheap & simple
- Can help a lot
- "One-pixel image" assumption
  - Not suitable for local light inter-reflections

14

To conclude, VPL rejection sampling is cheap and simple and can help a lot. There's really no reason for not using it, especially in mostly diffuse scenes, unless one wants to apply a more advanced VPL distribution techniques.

The problem is that it makes the "one-pixel image" assumption – it will not help us to resolve the local inter-reflection problem.

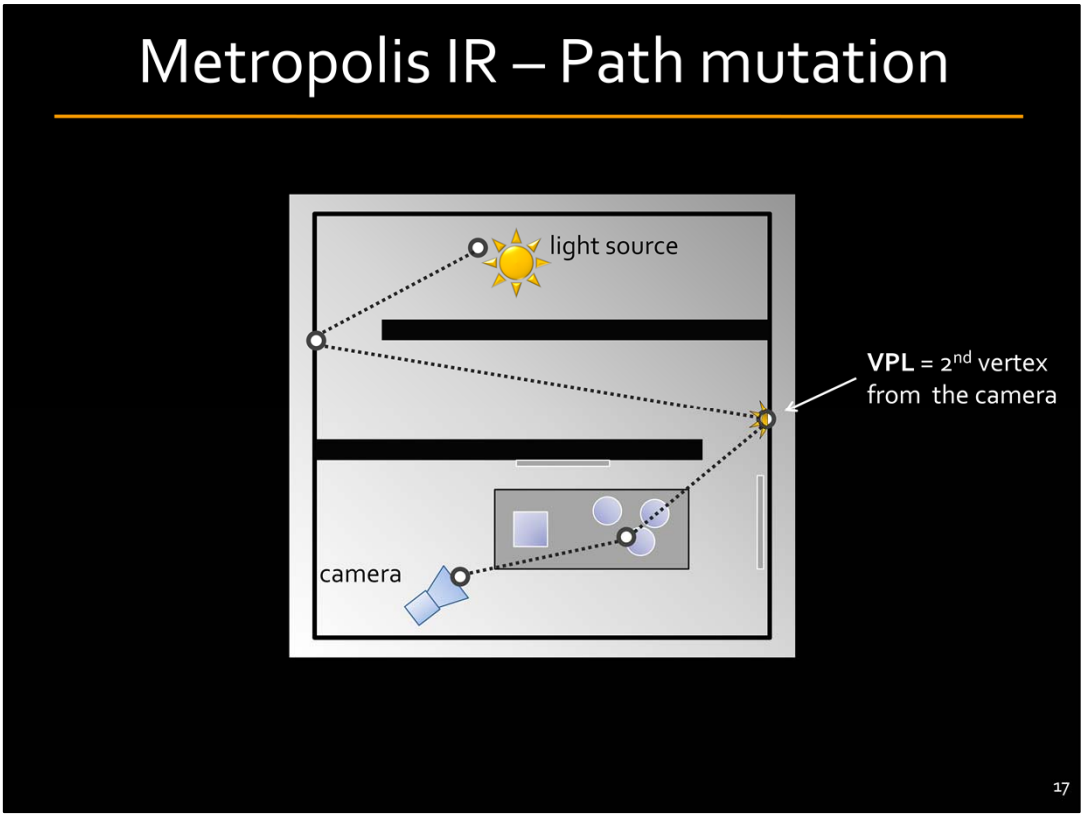
Metropolis sampling for VPL distribution

Metropolis sampling for VPL distrib.

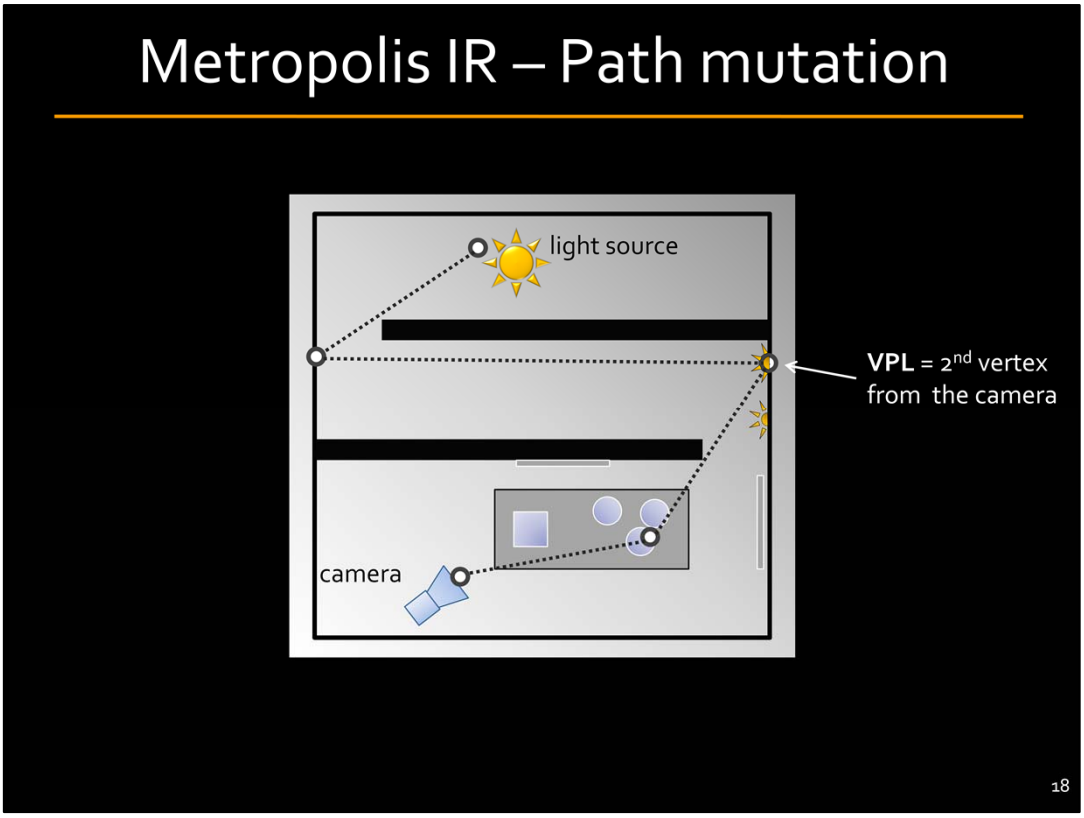
- "Metropolis instant radiosity" [Segovia et al., EG 2007]
- Good for large environments but not for local interactions

16

Another approach, presented by Ben Segovia et Eurographics 2007, relies on Metropolis-Hastings sampling to distribute the VPLs.

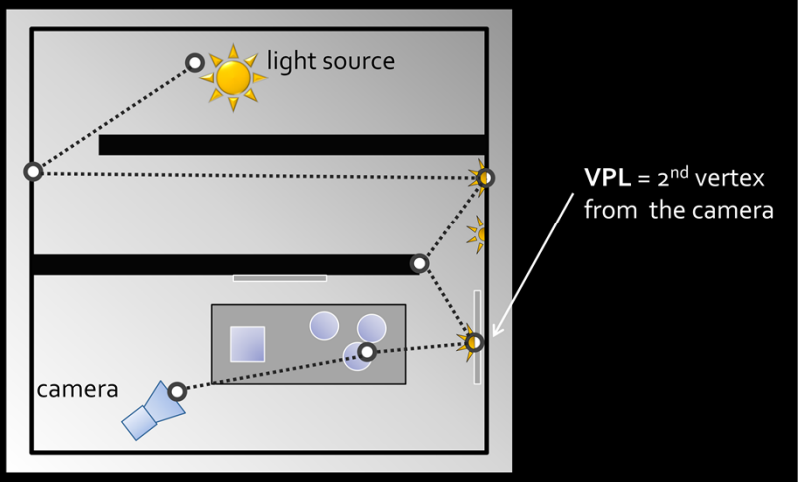


Given a light path that connects a light source to the camera, the 2nd vertex from the camera can be interpreted as a VPL (and stored and used for illuminating the scene when all other VPLs have been distributed).
Because the path connects the light to the camera, a VPL generated this way is very likely to have an important image contribution.



We can now use the Metropolis-Hastings algorithm, as in Veach's Metropolis Light Transport, to explore the space of all possible light paths by proposing local path mutations, as shown on the slide. Again, the 2nd vertex from the camera of the mutated path yields a VPL.

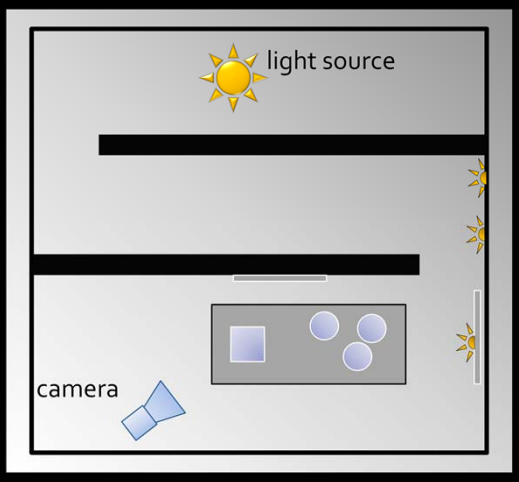
Metropolis IR – Path mutation



19

In this way, we can keep on mutating the path and generating new VPLs.

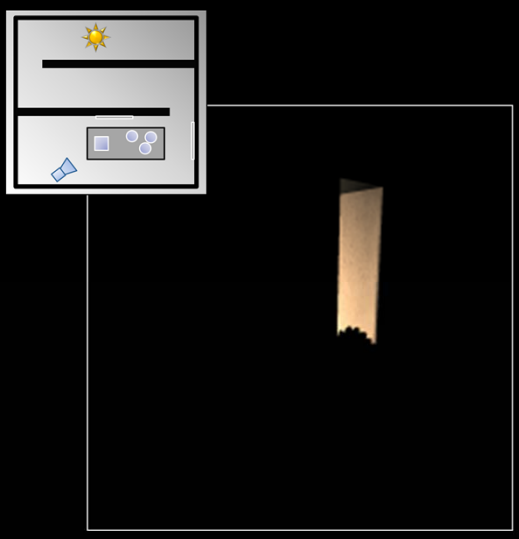
Metropolis IR – Resulting VPL set



20

All VPLs generated this way end up having important image contribution. In fact, it can be proven that they all contribute the exact same total power to the image.

Metropolis IR – Results

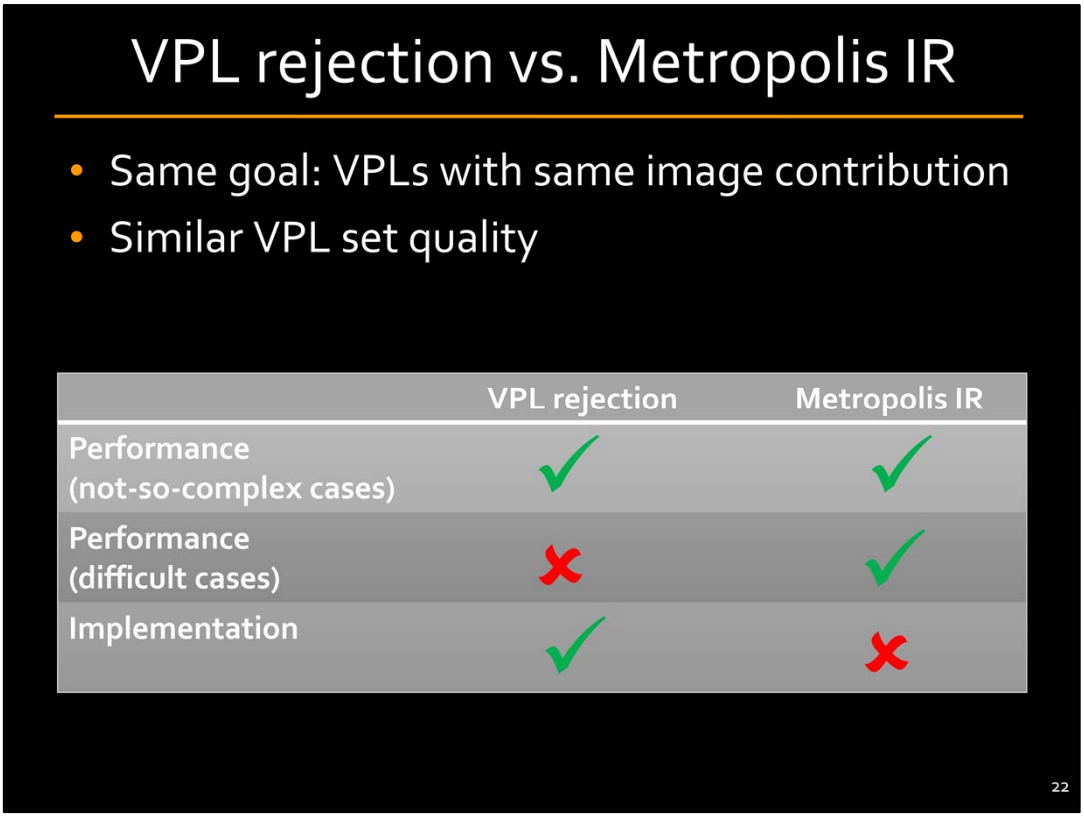


Instant Radiosity

Metropolis Instant Radiosity

Images courtesy of Ben Segovia and Bernard Péroche

21



Though very different, both the Georgiev et al.'s VPL rejection algorithm and Metropolis Instant Radiosity will generate a set of VPLs where each VPL has roughly the same contribution to the image. From this, we can expect that the VPL set generated by both algorithms will be of similar quality.

Performance of VPL generation in reasonably complex scenes is likely to be similar for both algorithm. However, for very complex scenes, where light must bounce many times to reach the camera, the rejection algorithm will perform poorly because it will have to reject many VPLs.

On the other hand, the implementation of the rejection algorithm is trivial but MIR requires substantial implementation effort.

Sampling VPLs from the camera

(Local VPLs)

Sampling VPLs from the camera

- Address the local inter-reflection problem
- Guaranteed to produce VPLs important for the image

24

None of the two previously discussed approaches help with the problem of local light inter-reflection.

To deal with this problem, it is more reasonable to distribute the VPLs by tracing paths from the camera instead of from the light sources. This approach is bound to produce VPLs in locations important for the image to be rendered, but there are some important technical issues that need to be taken care of:

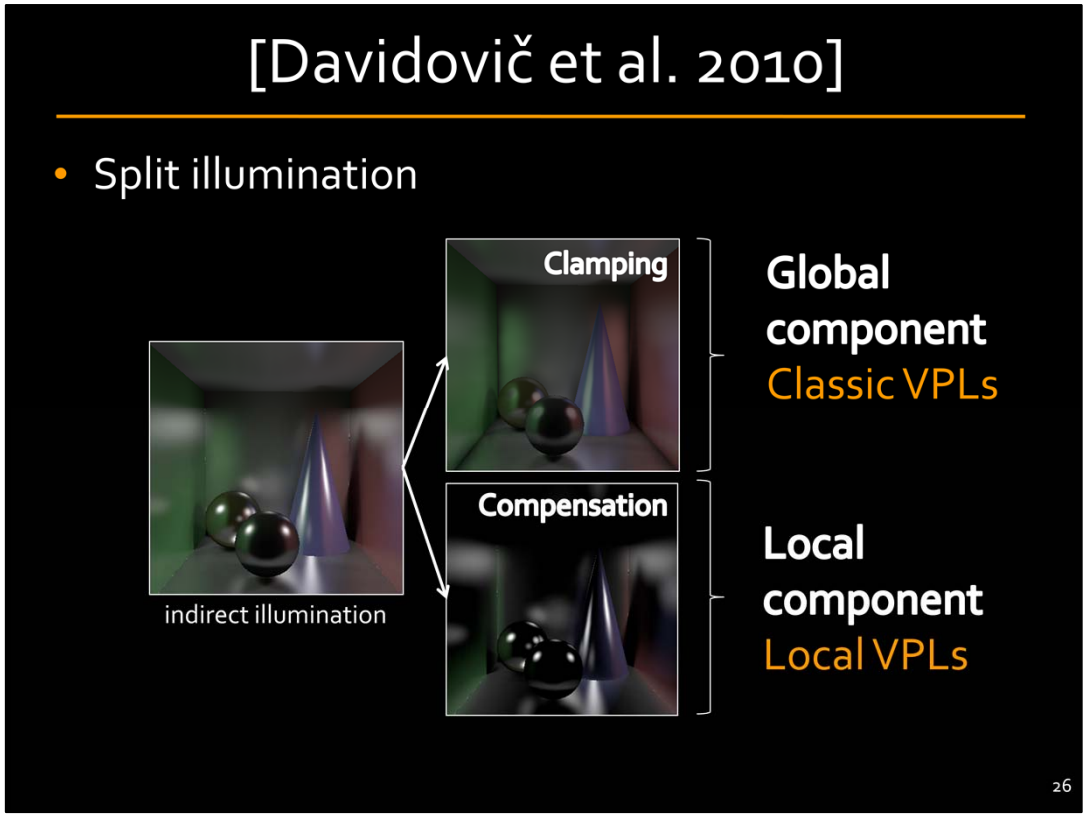
- First, we need to explicitly connect these VPLs to the light sources so that they can form complete light transport paths.
- Second, computing the VPL intensity involves the evaluation of the probability density of generating the particular VPL position. The probability calculation is significantly more complex (and costly) when the VPLs are generated from the camera.

Sampling VPLs from the camera

- "Bidirectional instant radiosity"
  [Segovia et al., EGSR 2006]
- "Local VPLs"
  [Davidovič et al., SIGGRAPH Asia 2010]

25

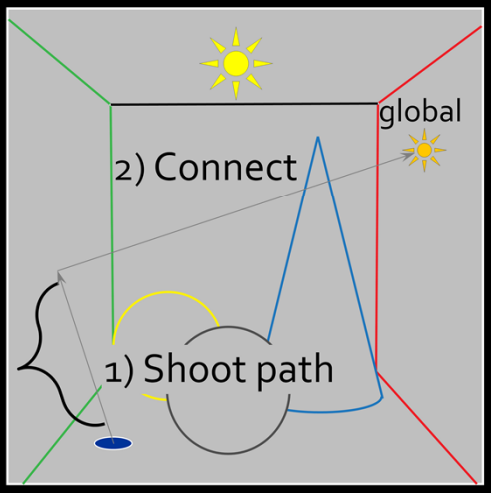
This idea has appeared in the two papers on the slide. In the following, I will discuss the method proposed by Davidovič, myself, Miloš Hašan and Kavita Bala in our SIGGRAPH Asia 2010 paper.



We use the idea of separating the light transport into the clamped, global component, and the local component, as previously discussed by Alexander and Miloš.
The global component accounts for the long-distance light transfer, while the local component corresponds to the short-range inter-reflections, and indirect glossy highlights.
We take advantage of the specific structure of each of the two components to design a solution for each of them that is substantially more efficient than a general GI solution.
Specifically, we handle as much energy as possible in the global component which leaves only the local inter-reflections for the local component, which we handle by the so called **local VPLs** that are distributed by tracing paths from the camera. I will only focus on the local component.

Review of compensation

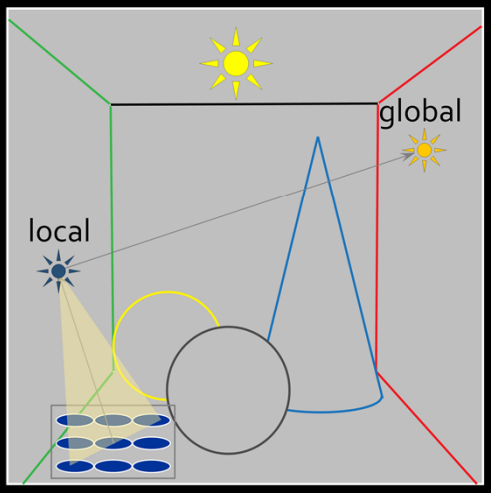
- Kollig & Keller compensation



27

Local VPLs – Idea

- [Davidovič et al. 2010]

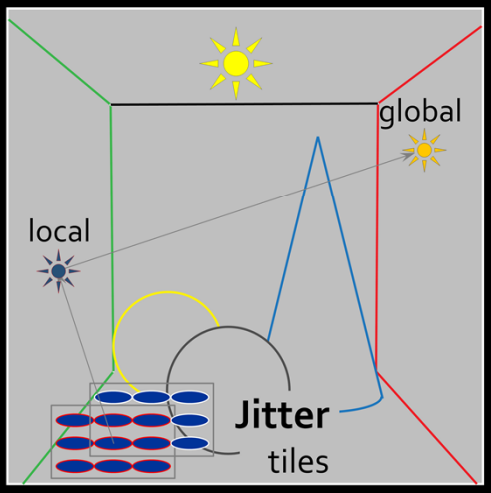


28

Our idea is to create a **local VPL** at the end point of the camera path. This way, the cost of tracing that path is amortized by letting it contribute not just to the pixel that generated the VPL, but also to a tile of its neighboring pixels.
With this basic idea, let us take a closer look at what is necessary to actually make it work.

Local VPLs – Technical solution

- [Davidovič et al. 2010]



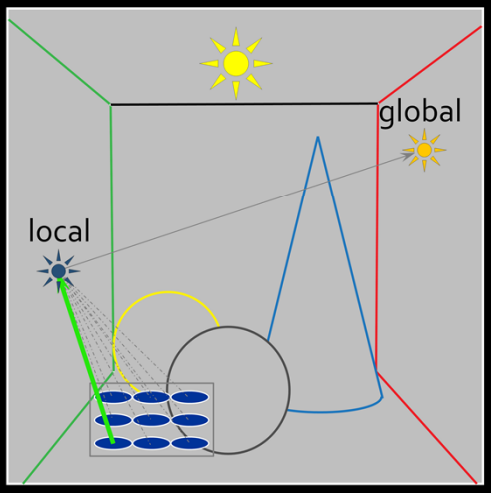
Probability density from **tile** pixels

29

The first important thing we have to compute is the probability density of the generated local VPL.
We cannot simply use the probability with which it has been generated, but we have to sum the probabilities over all the pixels in the tile it contributes to. The reason is that all the neighboring pixel could potentially have generated the VPL at this particular location.
The second important thing is that if we used a fixed tile grid, the boundaries would be fairly visible. To break this coherence, we jitter the tiles for each VPLs.

Local VPLs – Technical solution

- [Davidovič et al. 2010]



One-sample visibility

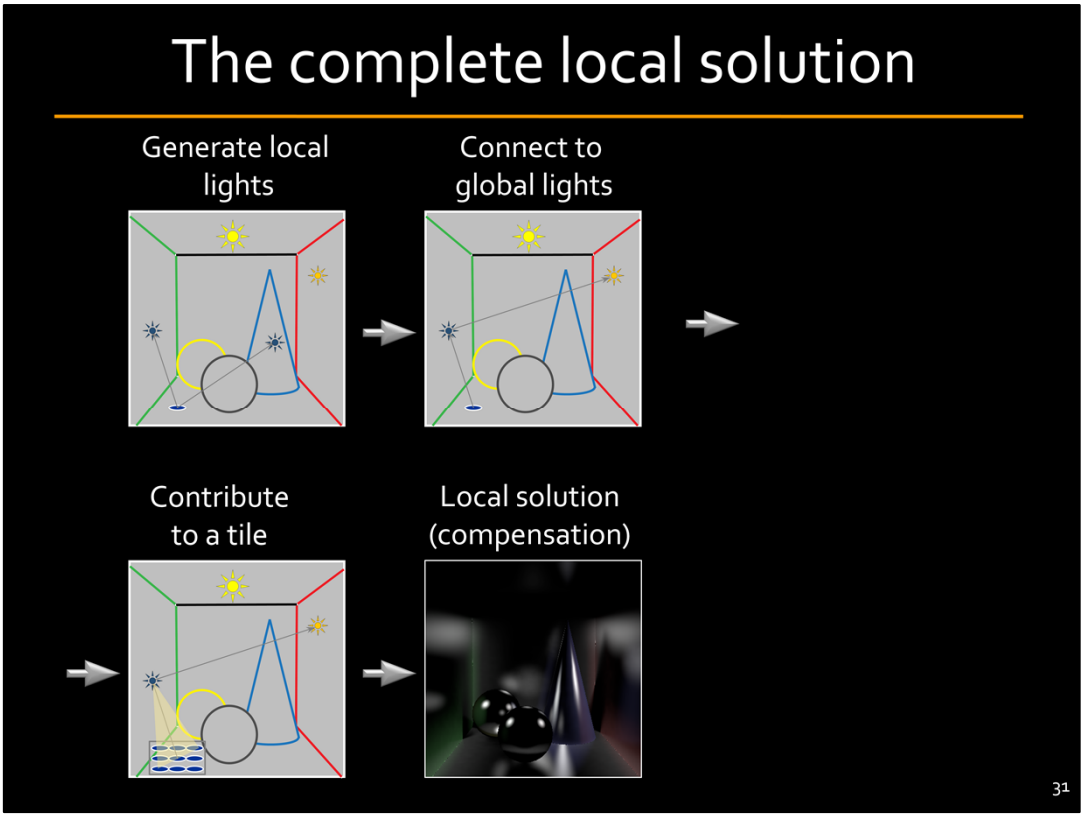
- Key idea: Tile visibility approximation

30

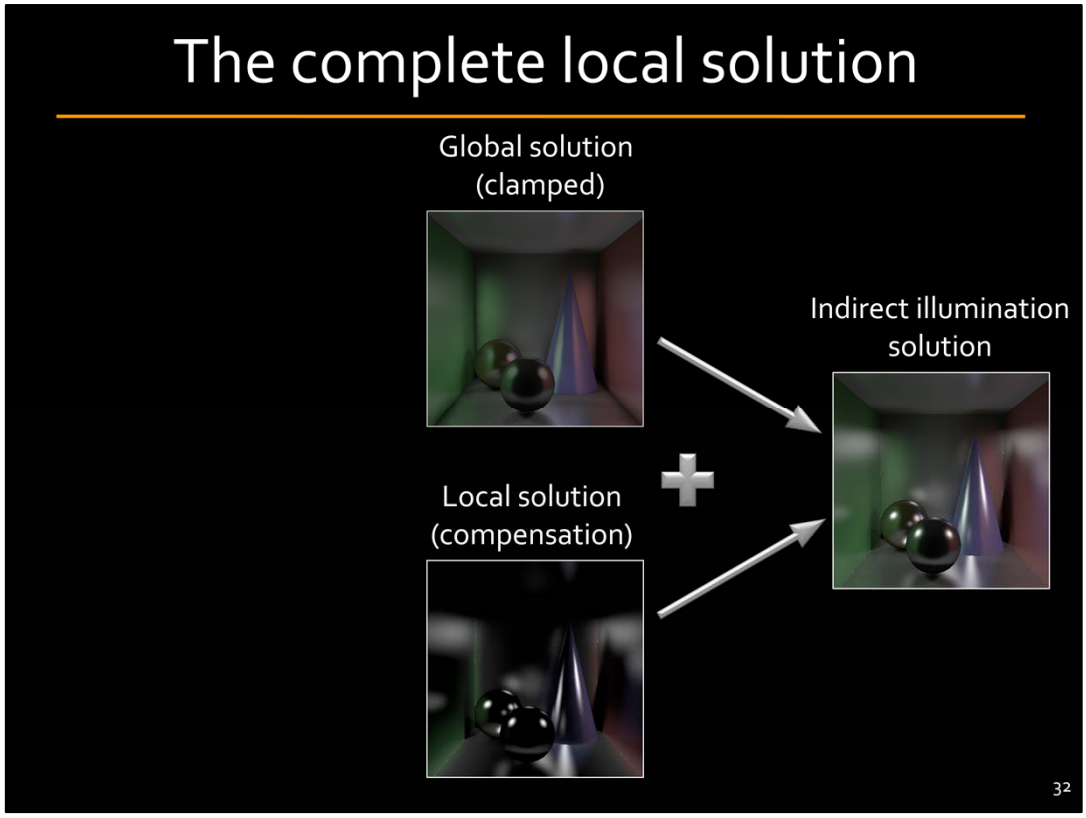
Now we come to the part that would simply not be possible without splitting the light transport.
To compute the full probability density for our light, we would normally need to compute visibility to all pixels, which would be prohibitively expensive.
However, we do have the global component that contains most of the energy and handles most of the shadows.
The local lights can then have their visibility approximated because they only handle local inter-reflections.

We approximate it by just one visibility sample, which is actually the ray that generated the light in the first place.

So the key insight here, the light transport split made tile visibility approximation possible.



So, for the overview of the whole process.
We generate local lights
Reject lights with zero contribution
Connect the surviving local lights to global lights
Have them contribute to a tile
And after repeating this about 20 million times, we get the final local solution.



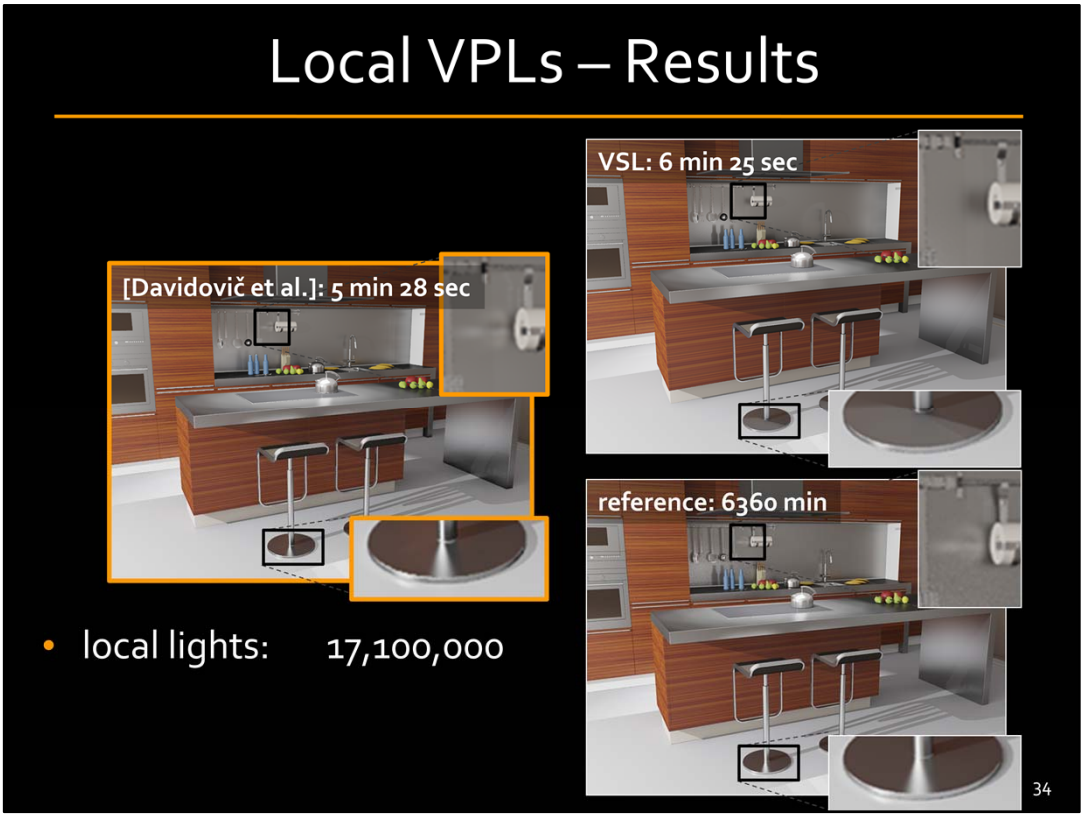
Now we have the result of the local solution. We simply add the result of the global solution and obtain the final indirect illumination solution.

The global solution can be computed by any VPL method, for example Lightcuts. In the original paper we used used a visibility clustering algorithm.

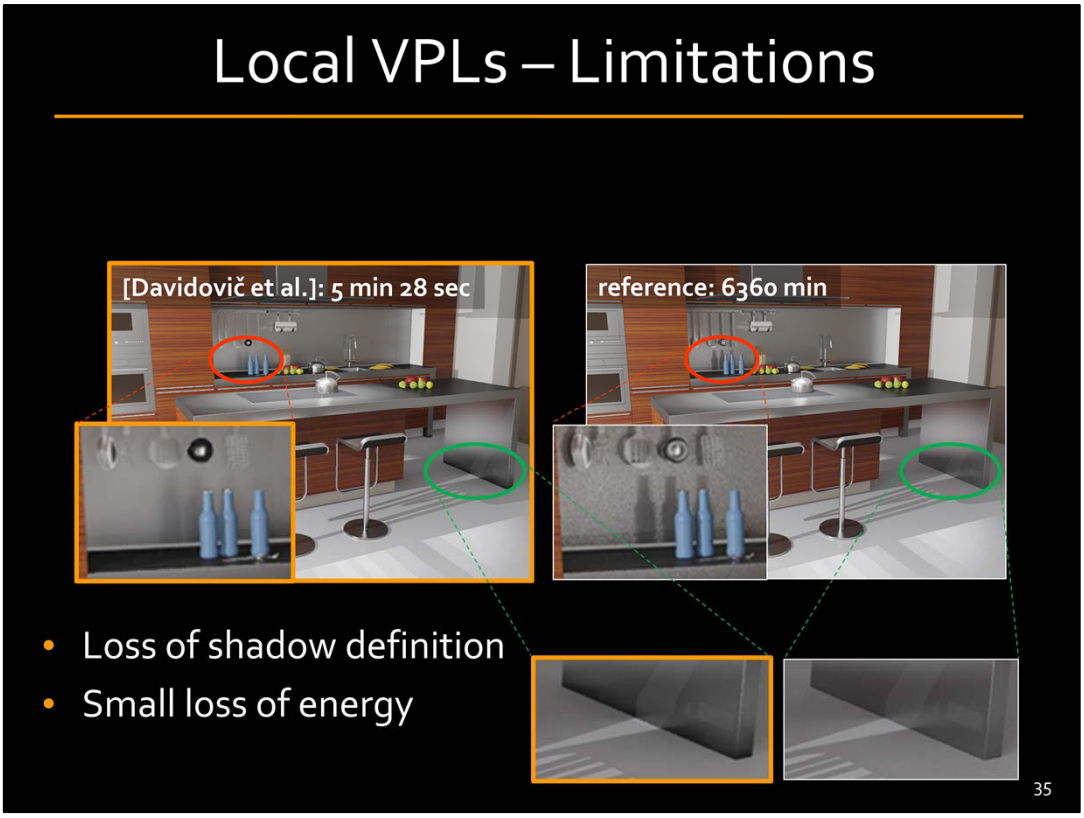
Local VPLs – Results

- local lights: 17,100,000

33



Here we see that the local lights nicely capture this highlight from metal stool leg or the reflection of the paper towel on the metal back wall, something that cannot be done with the “globally distributed” VSLs.



However, this scene also shows some of the limitations the method has.
There is a loss of definition of the shadows behind the bottles.
This is caused by the fact that the local lights on the kettle in the front contain too much energy. Pushing more energy into the global component could resolve this problem.

One detail we did not mention is that in some of the scenes we still need slight clamping even on the Local VPLs, causing some darkening here. This can be solved by interpreting the Local VPLs as Local VSLs.

Local VPLs – Conclusions

- Good for local inter-reflections
- Really useful only when used in conjunction with a separate "global" solution

36

To conclude, distributing VPLs by tracing paths from the camera is very useful for resolving local inter-reflections.

They are best used in conjunction with a separate "global" solution which can take care of the smooth, long distance light transport in the scene.

Thank you