Brady Ledger
Supervisor: Michael Doggett

Minimising Light Leakage in Real-Time Photon Splatting

Hi everyone, thanks for staying to watch. My name is Brady Ledger and I am here to present my thesis entitled Minimising Light Leakage in Real-Time Photon Splatting.

Summary of Presentation

- Motivation
- Background
  - Global Illumination
  - Photon based GI
  - Problem
- Algorithm
- Results
  - Visuals
  - Performance
- Conclusions

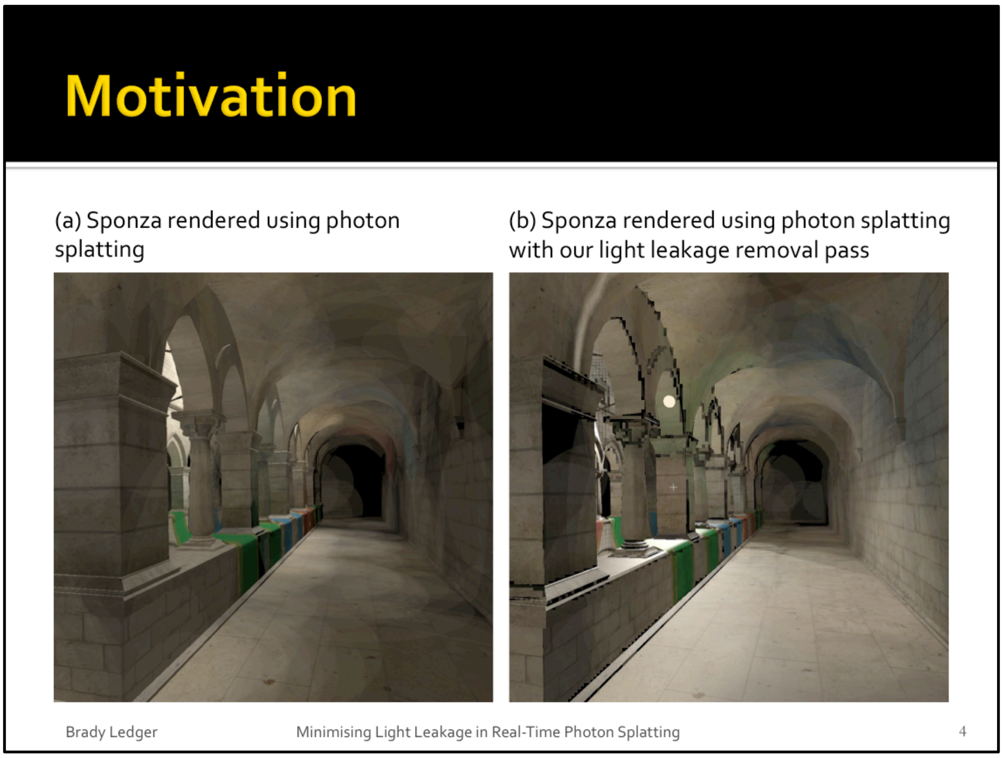
Brady Ledger Minimising Light Leakage in Real-Time Photon Splatting 2

Here is a summary of what I'll be talking about today.
I will start with the motivation for the thesis.
I shall then talk a little about background topics to better understand the work I am presenting today.
I will explain the problem that this work addresses before getting into how the algorithm I present operates.
We will then take a look at the results, how it performs and what conclusions can be drawn from this.

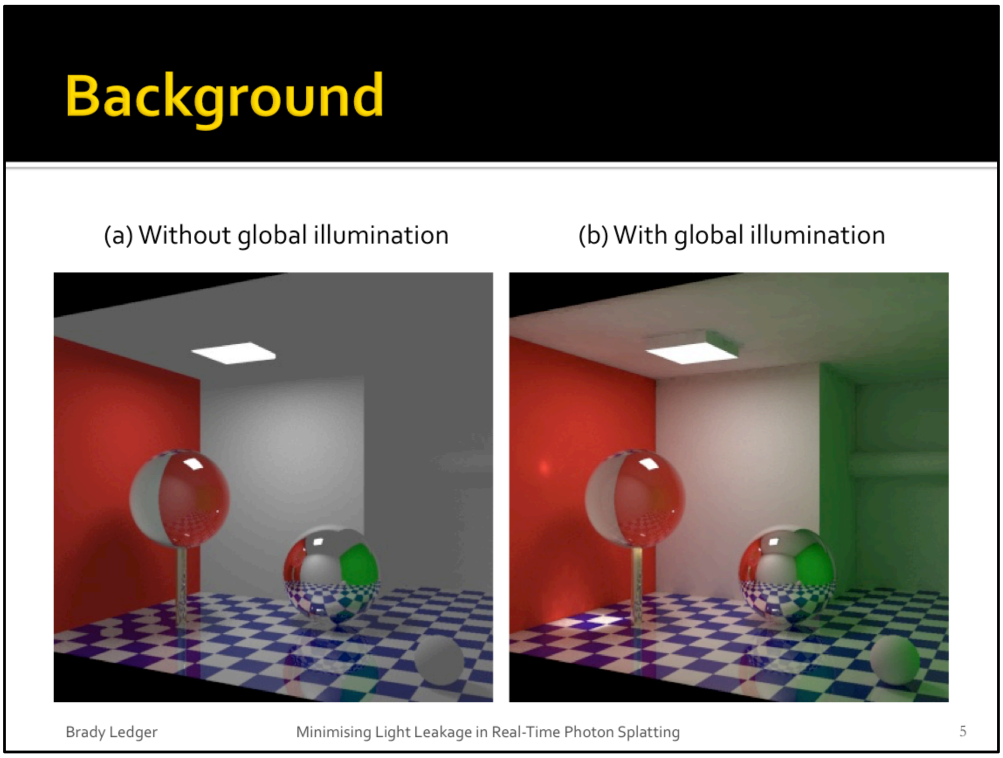
Motivation

Figure: Screenshot from Star Wars Battlefront II (Release Nov 2017)

Modern computer games continue to push at the boundaries of photorealism to differentiate themselves from other games in the very competitive gaming market. Part of this push has been to add more physically based lighting by using indirect interreflection of light. The dynamic nature of computer games requires leverage of graphical realism with computational efficiency on current hardware; this is necessary in order to have the scene update in real-time. Despite the parallelism available on the GPU to compute physically based lighting, numerically accurate global illumination on current hardware remains computationally expensive; thus approximate algorithms, that are able to render scenes in real-time for interactive applications, have become an area of research.

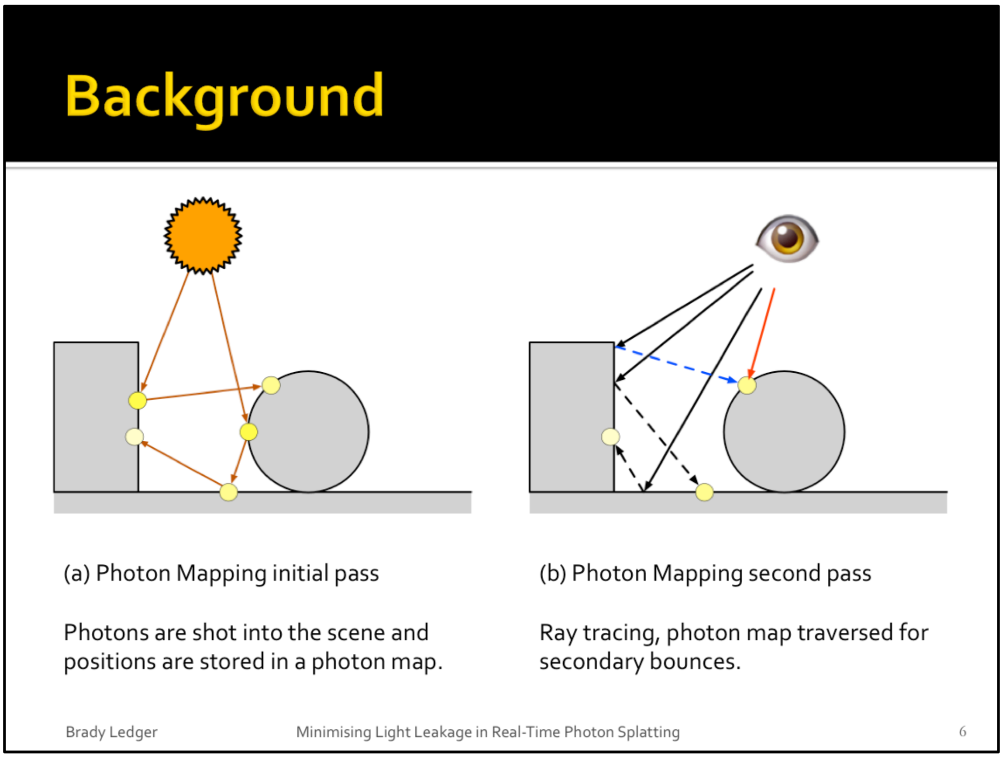


A real time global illumination technique we wanted to improve upon was photon splatting. It can produce physically based lighting in real time by splatting photons into the scene. A problem it carries however is light leakage, towards which we researched and now present a solution.



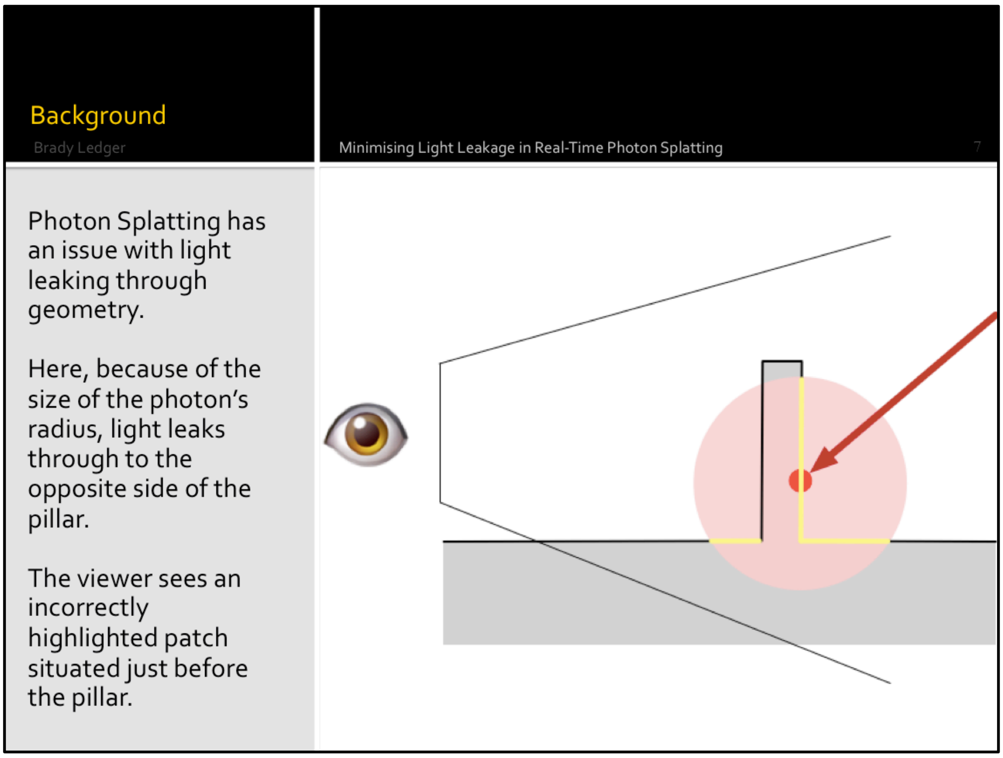
I will talk a little about background, useful in understanding the topic, and what I'm doing. Global Illumination is adding indirect light to a scene, that is to say to render the light- that has bounced off-and-between surfaces as it does in real life.

The left image is rendered without global illumination. Areas that lie outside of the ceiling lamp's direct light lack definition and the lamp itself appears 2-dimensional. Without ambient lighting in the scene, this entire image would be black. The right-hand image is rendered with global illumination. Light is reflected by the surfaces in the scene. We can also see how colour from the red wall and green wall (only visible in the reflection of the sphere) reflects onto other surfaces in the scene. Another thing global illumination enables are caustics (a type of light refraction), which have been projected onto the red wall from light passing through the glass sphere.



Photon mapping is a popular global illumination technique that photon splatting, the algorithm we use, is based off. A lot of concepts are the same so I will explain first how photon mapping works and then photon splatting.

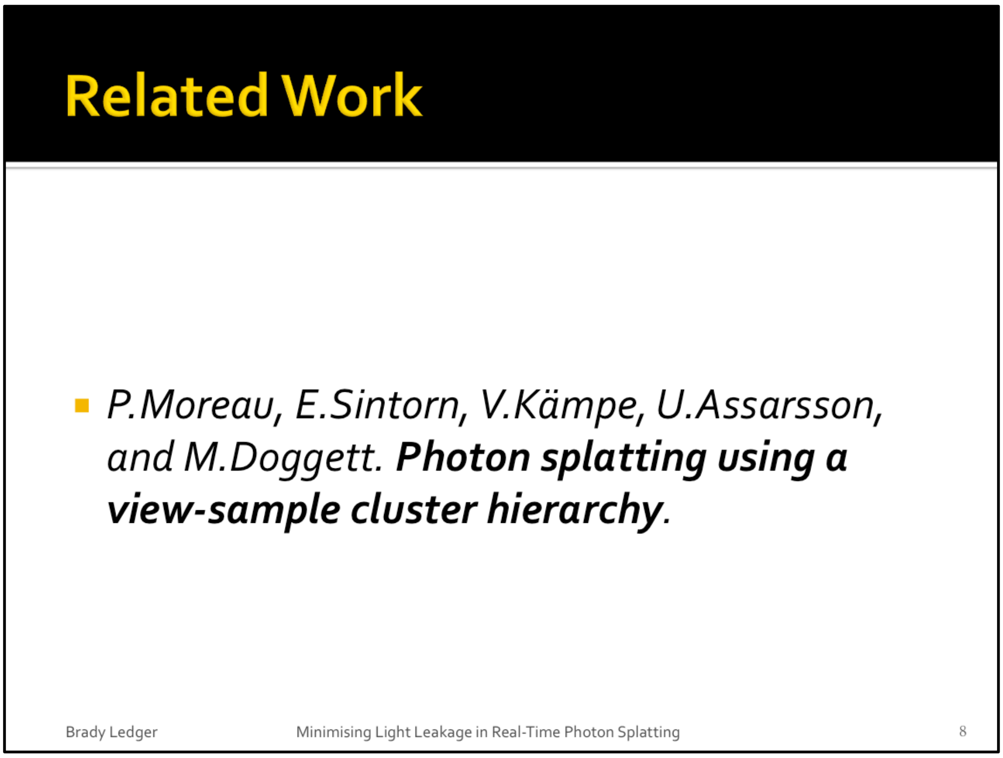
As picture (A) details, in photon mapping's initial pass, light rays (or photon paths) are shot into the scene from the light source, and, where they become incident with geometry, the location is stored in a photon map.
(B) Shows photon mapping's second pass whereby ray tracing is applied. Ray tracing is where we trace a path of light from a virtual camera (the eye) through each pixel in an image plane. On intersection with an object, the amount of light which is reflected back to the camera (eye), also known as radiance, is calculated and additional rays are cast from that point. Photon mapping uses ray tracing for the initial bounce in a scene, but all subsequent bounces query the photon map for its radiance. such that here you can see the red primary ray will **not** query the photon map for the radiance of the view sample it hits, instead calculating its own reflectance. However the photon map **is** queried to calculate radiance when another ray spawns a secondary ray (which is the blue ray here that becomes incident with the view sample). When querying the photon map, radiance is estimated at a point on a surface by gathering the nearest photons. This can result in gathering photons which should not contribute to the radiance of the point, and becomes a prevalent issue when photons with large radii are used. Querying the photon map is thus too costly for real-time



A long standing issue of photon splatting is light leakage This occurs when light from a photon leaks through geometry because of its size. View samples may accumulate radiance from a photon splatted to a surface which is hidden by geometry.

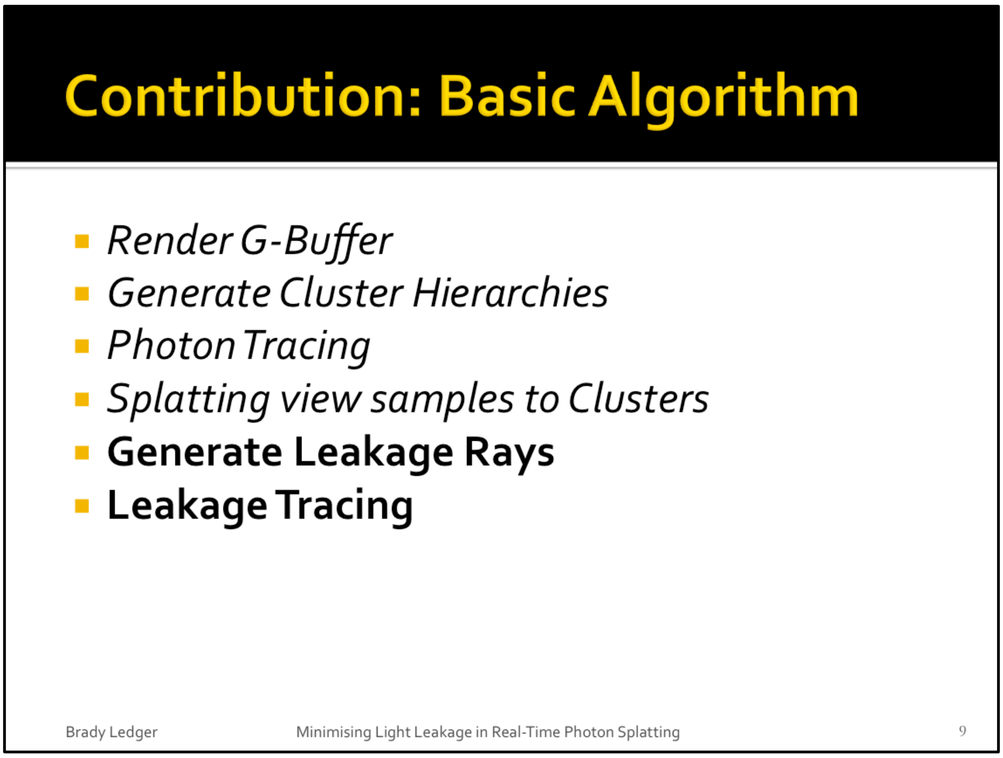
This image here shows an incoming photon that travels along the red arrow vector and becomes incident with the geometry at this red point. Here, the region of influence is created based on the photon's radius (this larger pink circle). The surface positions in the scene which are determined to have contribution- fall within the radius of the photon's position. These enclosed surface positions are considered to reflect the light as their surface normals oppose the direction of the ray, this is indicated by the yellow coverage on the geometry. The surface within the radius that is not covered is not considered to accumulate radiance, due to the surface normal facing away from the direction of the ray. When the scene is rendered from the position of the viewer, light from the photon that hits behind the pillar leaks through into view, which is denoted by the yellow coverage here.

This problem is what my thesis aims to minimise.



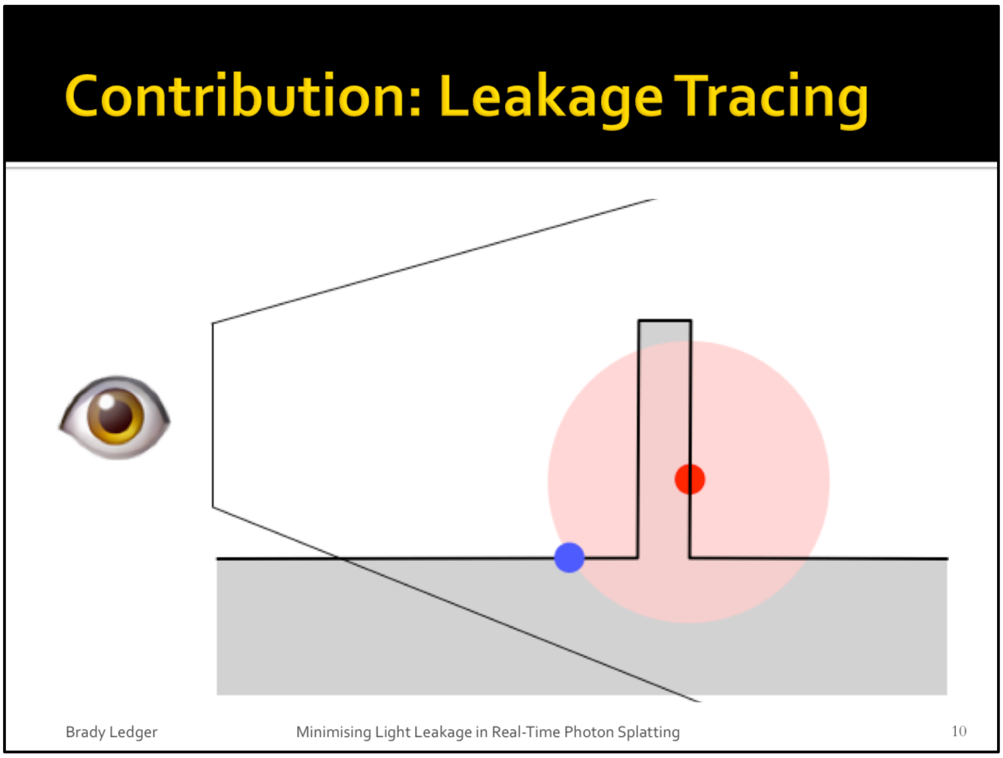
The work I have produced is based on a recent photon splatting algorithm by Moreau et al.. They chose to splat view samples with their splatted photon indices into a cluster hierarchy, as opposed to storing the photons themselves.

The algorithm they produced was very efficient, and gives us an opportunity to find some milliseconds to work with in order to add a leakage removal pass.



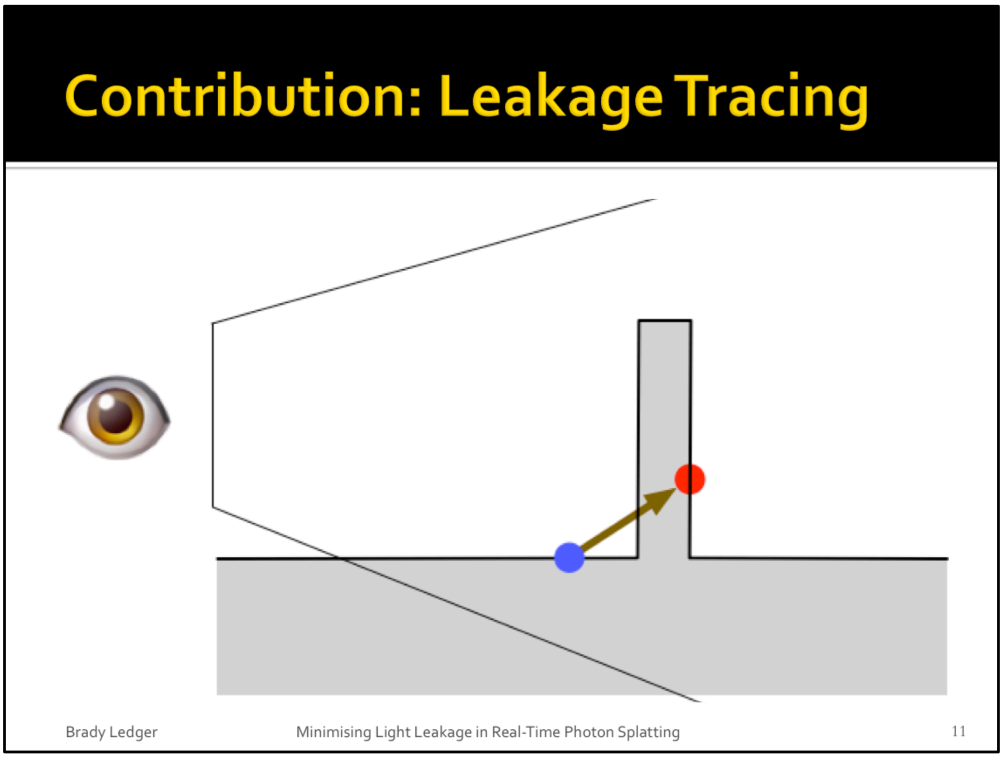
Many of the underlying passes in the photon splatting algorithm remain unchanged from the Moreau et al. implementation. We are primarily interested in the photons that have been considered to contribute to a view sample. Alterations and new work follows the splatting stage.

- Initially we Render the G-Buffer. The view sample positions and normals are rendered to a GBuffer texture using the standard pipeline.
- Then, Based on the view sample positions from the previous step, generate the cluster hierarchy.
- Paths are traced from the light source, and at each bounce (from the second bounce onward) a photon is stored to a list.
- The photons are then traversed through the hierarchy where their ID's are stored following intersection or envelopment of a node.
- Our algorithm comes in here and For each view sample, we traverse the photon lists of the containing nodes and generate a ray from the pixel's location in world space in the direction of the photon's world space location.
- At the leakage tracing stage, paths are traced from the pixel in the direction of each contributing photon.

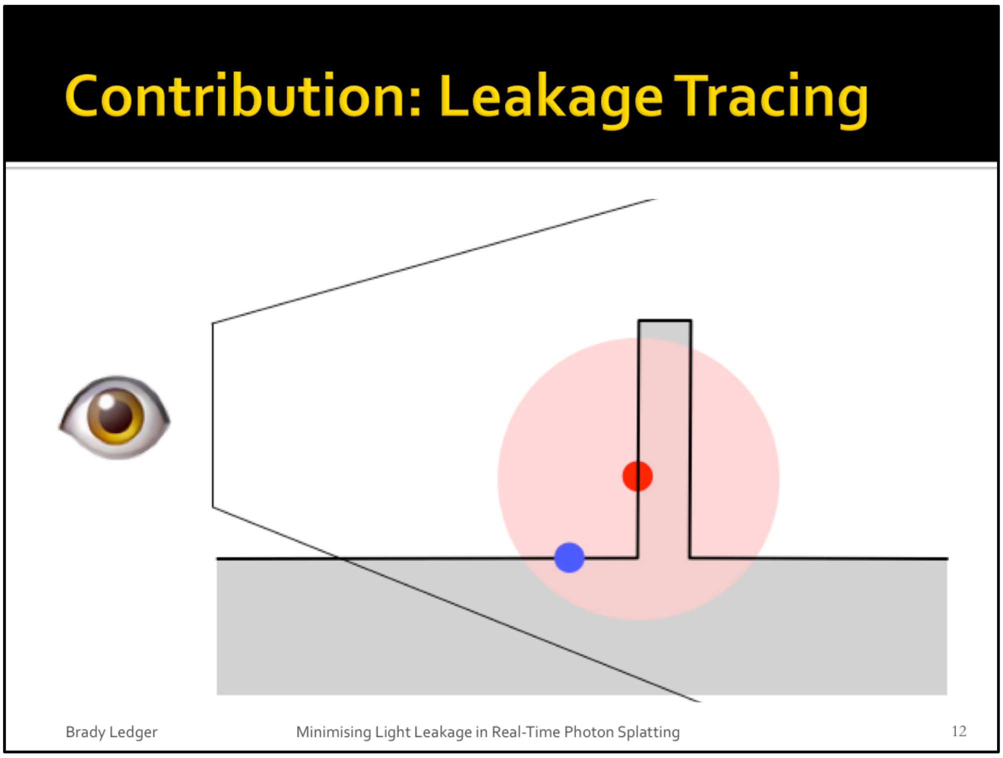


Assume the blue dot is our pixel it lies within the radius of the photon located at the red dot, and thus would accumulate radiance.

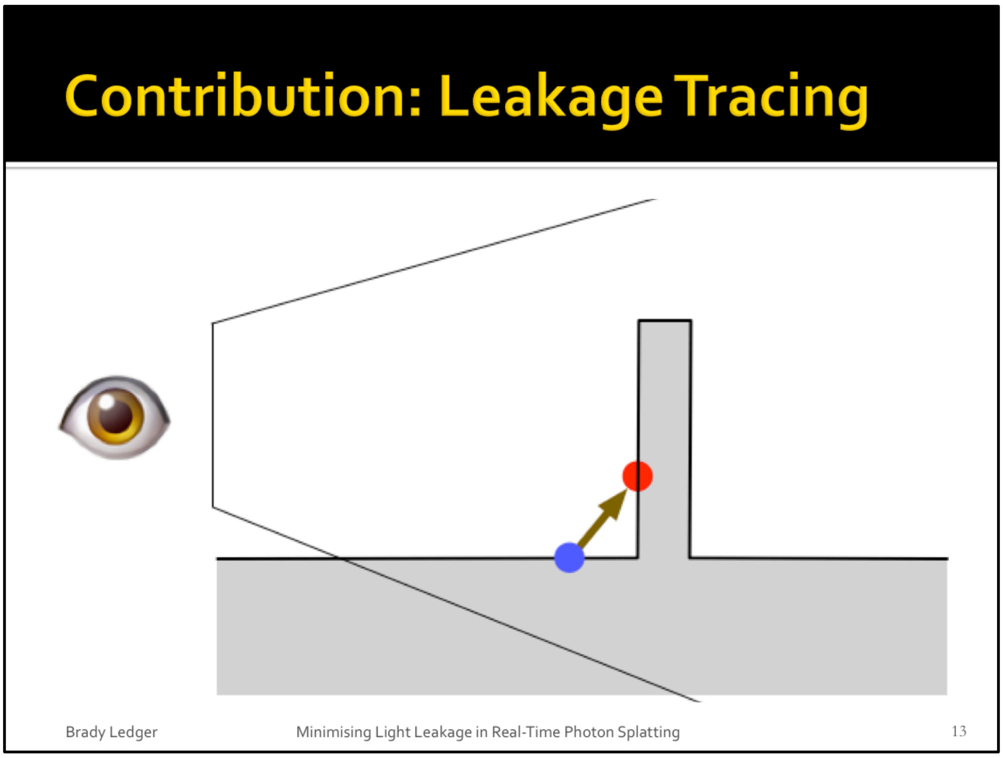
We trace a ray from the blue dot pixel in the direction of the red dot photon.



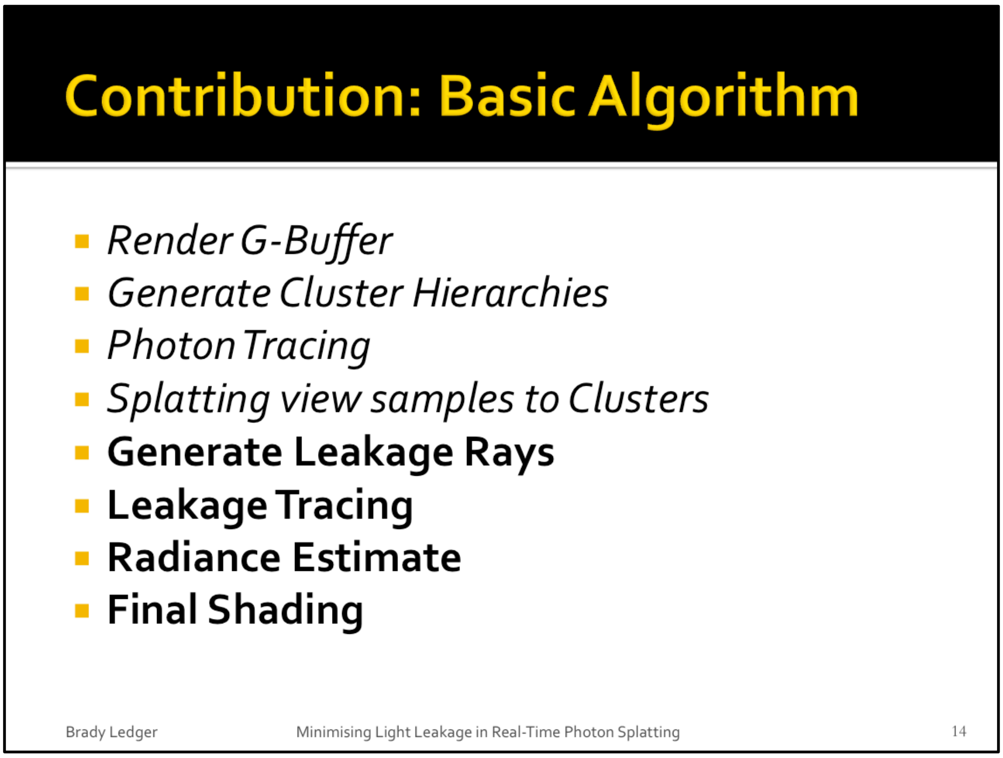
OptiX, our ray tracer outputs data on the trace. This ray has intersected geometry and therefore has a “hit” value.
We can thus loop over all of the outputs for the ray trace operation to remove photon contributions to pixels whose associated rays hit geometry.



In another example, our blue pixel falls within the radius of the photon located at the red dot and is deemed to accumulate radiance.
We again trace a ray from the blue dot pixel in the direction of the red dot photon.

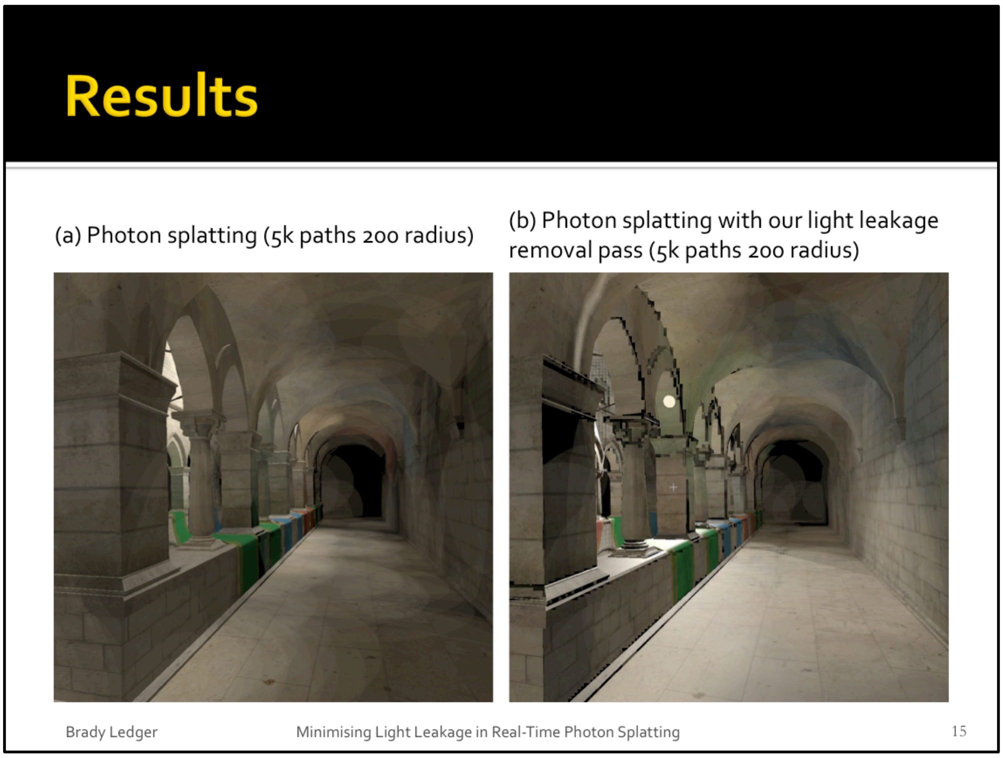


This time the ray that we have traced did not encounter geometry when shot. It's output thus has a negative hit value and it is safe to accumulate the energy from the photon at the red dot's position into the radiance estimate for the pixel situated at the position of the blue dot.



So in estimating the radiance, as was described from the previous diagrams,

- Rays from the trace operation that become incident with geometry are determined to have leaked. Thus the associated photon for the ray is culled from contributing towards the radiance of its associated pixel. All photons judged to be legitimate are accumulated into the radiance estimates for its corresponding pixel.
- In final shading, Direct lighting is calculated in a full-screen pass. The indirect lighting we've accumulated from the previous pass, is added towards achieving the final pixel colour.

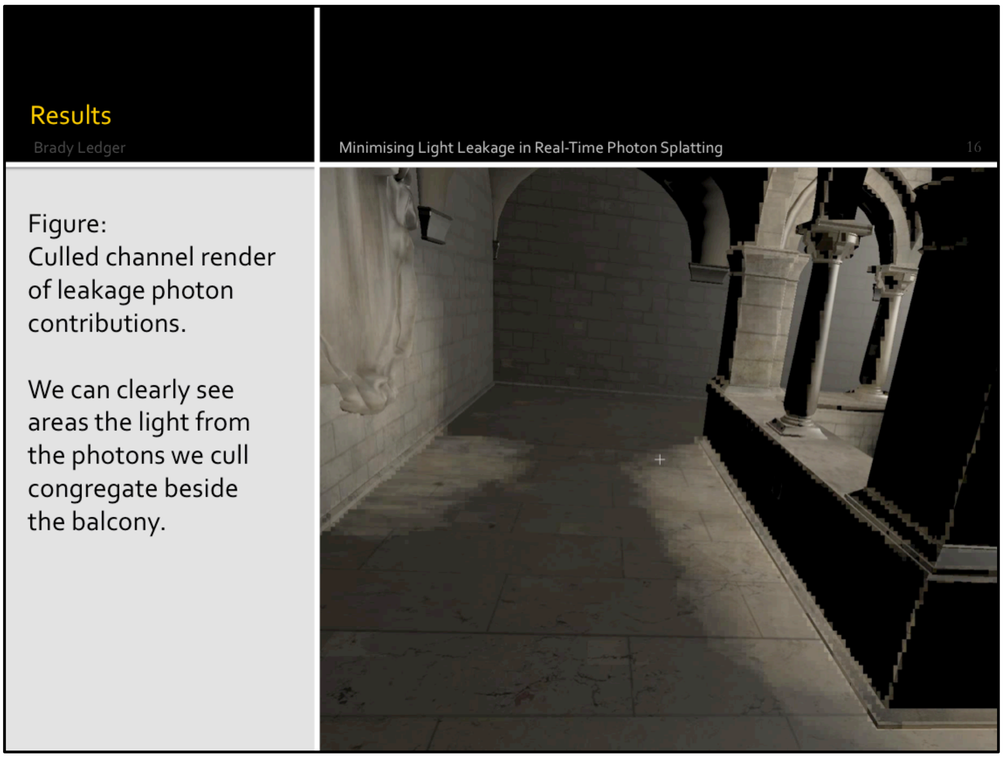


Here you can see the results of our algorithm.

The right-hand image shows the result of performing photon splatting with light leakage minimisation onto a scene in Sponza. We present this beside this left-hand image, which is of the original kernel running the same settings. It is shown that the leakage photons present within the original kernel's output image are not present in the leakage traced image, and such highlighting occuring around the base of the Sponza balcony is minimised. There remains some unnatural light, but the lighting gradient on the floor is much more believable having minimised the leakage contributions.

The artifacts covering the edges in the image our splatting algorithm produces are a result of us having to apply a cap on the amount of photons collected per pixel. We do this because edge view samples have contributions from an inordinate amount of photons. It is possible to fix this by placing the excess photons for edges in a buffer to add to the final radiance estimation.

As is also the case with photon mapping, increasing the amount of photons and reducing the sizes of their radii can improve lighting accuracy and final image quality, but at the cost of further computation time.



The image here is produced from rendering the culled contributions of photons with direct lighting.

The lighter areas in the scene are from photons causing leakage, most notably the area central to the image that lies beside the balcony.
These balcony areas are the most susceptible parts of the Sponza scene to produce light leakage, and as such we can clearly associate the light leakage with these highlighted areas of culled photons.

Performance

- Overhead
  - Ray tracing pass
    - 10k paths 255 radius ≈ 78ms
    - 75k paths 150 radius ≈ 78ms
  - Extra buffers, kernel calls
  - Removing trivial accumulation in the cluster hierarchy

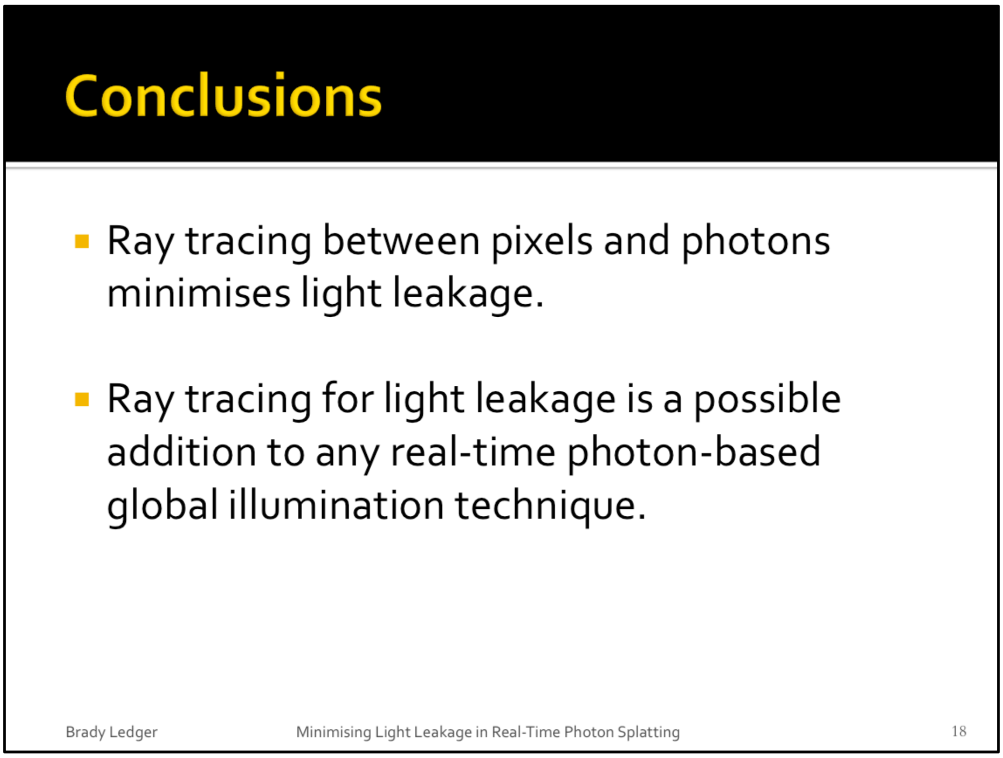
Brady Ledger Minimising Light Leakage in Real-Time Photon Splatting 17

We must take into consideration that we are adding to an existing algorithm, results will inevitably be slower. Thus the overhead for tracing the leakage photons increases the draw time for a frame. Measuring times between executions show that despite running with a greater amount of photons aiming for providing better lighting accuracy, timings are approximately the same when compared to the fewer but larger photon execution. So 10k paths with a radius size of 255 performs the same in the leakage tracing as 75k paths with a radius size of 150. This is because the buffers collecting photons for leakage ray tracing have a maximum limit, and each pixel has a cap on how many photons could possibly be traced. The overhead created by the leakage ray trace operation clearly dictates the timing for our photon splatting algorithm.

The timings here push the boundaries of the GPU memory for accuracy purposes. 25 photons per pixel were used which is significantly more than required to get a reasonable coverage for photons splatted onto a planar surface, and such incur a 78ms overhead to the algorithm for photon collection, tracing and radiance accumulation. A smaller cap would substantially reduce the overhead for this trace, and presents the possibility to be performed in real-time.

Extra buffer allocations, clearing and kernel also add to the overhead our algorithm produces.

In order to enable the leakage trace we remove any possibility for trivially accepting photons' radiance in the view cluster hierarchy, which has previously been shown to accumulate energy for a cluster of view samples should they be enveloped by a photon. Removing this technique is necessary as this method ignores the possibility of light leakage entirely.



From our findings we can conclude that ray tracing between pixels and photons can be effectively used in minimising the light leakage caused by large photons.

We can also conclude from our findings that using ray tracing to trace light leakage contributions from photons is a feasible addition to real-time photon splatting, and any other variation of real-time photon based global illumination.

Finally, I want to thank you for coming here today to listen to my thesis presentation. I also wish to thank Michael Doggett, my supervisor, and Pierre Moreau who has been of great help in my time learning OptiX. I hope you found the work I've been doing somewhat interesting, and I would be happy to take questions.