Auxiliary Features-Guided Super Resolution for Monte Carlo Rendering

Abhishek Jaiswal (A20380004)
Jackie McAninch (A20436746)
Zainab Hasnain (A20516879)

Deep Learning

Computer Graphics

Monte Carlo

Auxiliary Features-Guided Super Resolution for Monte Carlo Rendering

Neural Network

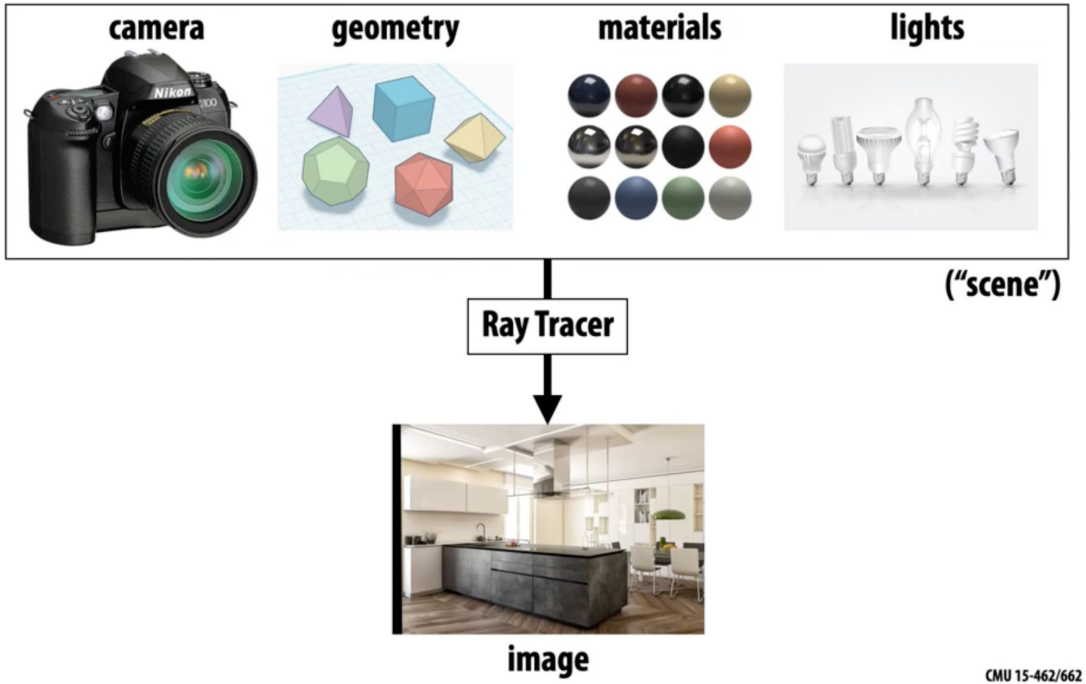
Computer Vision

Ray Tracing

Rendering Equation

What is Rendering?

- Rendering refers to the process of creating an image from a 3D scene
- A scene consists of geometry, materials, lighting, and other visual elements



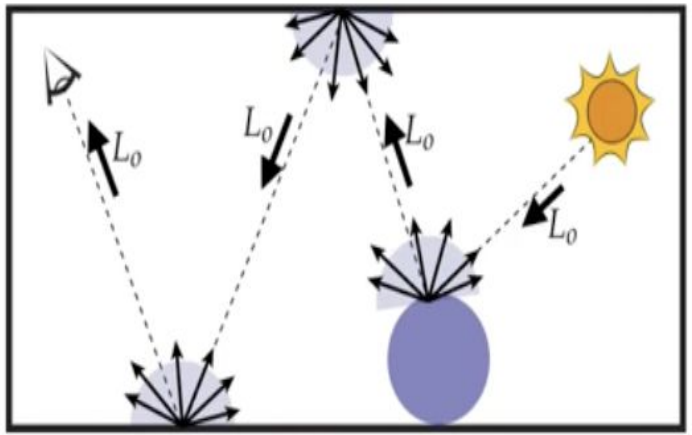
Rendering Techniques

- This paper deals with improving Monte Carlo rendering
- Other primitive techniques of rendering:
  - **Rasterized** images are not smooth, as they consider the local geometry for rendering
  - **Ray tracing** renders high quality images by considering “light paths”
- MC rendering is a method of ray tracing which minimizes computation

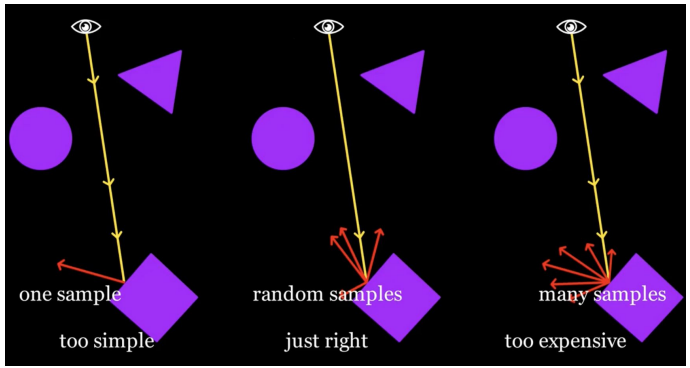
Rendering Equation

$$L_{\mathrm{o}}(\mathbf{x}, \omega_{\mathrm{o}}, \lambda, t) = L_{\mathrm{e}}(\mathbf{x}, \omega_{\mathrm{o}}, \lambda, t) + \int_{\Omega} f_{\mathrm{r}}(\mathbf{x}, \omega_{\mathrm{i}}, \omega_{\mathrm{o}}, \lambda, t) L_{\mathrm{i}}(\mathbf{x}, \omega_{\mathrm{i}}, \lambda, t)(\omega_{\mathrm{i}} \cdot \mathbf{n}) \, \mathrm{d}\, \omega_{\mathrm{i}}$$

- Different from stock pricing: ray tracing deals with a contained system
- Single path/ray is shown in the figure on the right
- However, physical light system is very complex
- We can simplify it by finding the total light emitted by each pixel (in RGB scale)
- The rendering equation below cannot be solved analytically
- We can APPROXIMATE it using Monte Carlo methods.



Monte Carlo Rendering



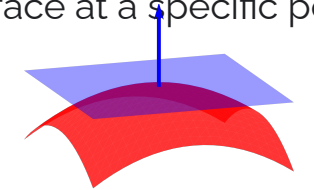
- MC rendering requires a lot of samples per pixel
- Therefore, it is computationally expensive
- With insufficient samples, the rendering results suffer from noise
- Some proposed solutions:
  - Importance sampling
  - Denoising using deep learning
  - **Our Method: Auxiliary features-guided Super Resolution**

Auxiliary Features

We use *albedo* and *normal* auxiliary features. These are **fast to compute** and contain **high frequency information.**

1. Albedo is the fraction of light that a surface reflects. If it is all reflected, the albedo is equal to 1. If 30% is reflected, the albedo is 0.3.

2. Normal refers to a vector that is perpendicular (or "normal") to a surface at a specific point.



Super Resolution

- It is an ill-posed problem
- It computes a high-resolution render from a low resolution image
- It uses deep neural networks to speed up rendering
- Our method uses two inputs: Auxiliary features and low-resolution render to produce a high-resolution image
- This is done by leveraging neighbouring frames to improve quality
- We split the image into patches and feed those tokens to transformer networks

Training Details

Hyperparameters

- Implementation in PyTorch
- Used Adam optimizer
- Learning rate: 0.0001
- 400 epochs
- Mini batch size of 16
- Takes 1 week to train the model

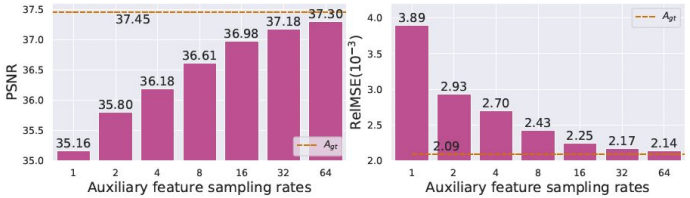


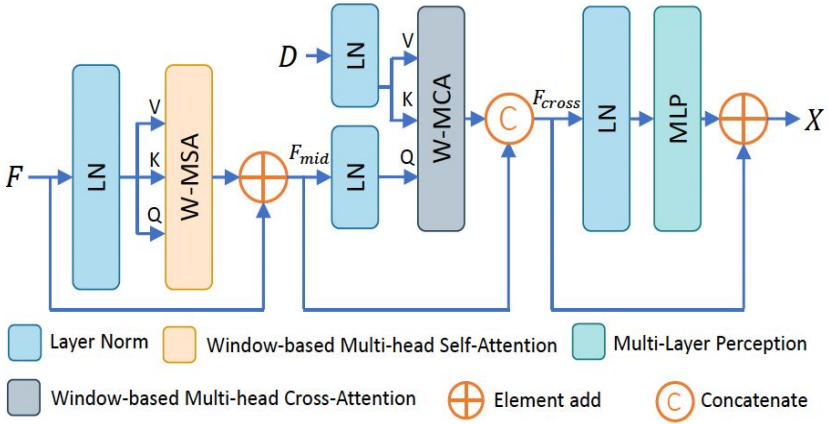
Figure 5: *The effects of the sample rates used to generate fast-to-compute auxiliary features on the performance of our method. A_{gt} indicates the ground truth auxiliary features (4000spp).*

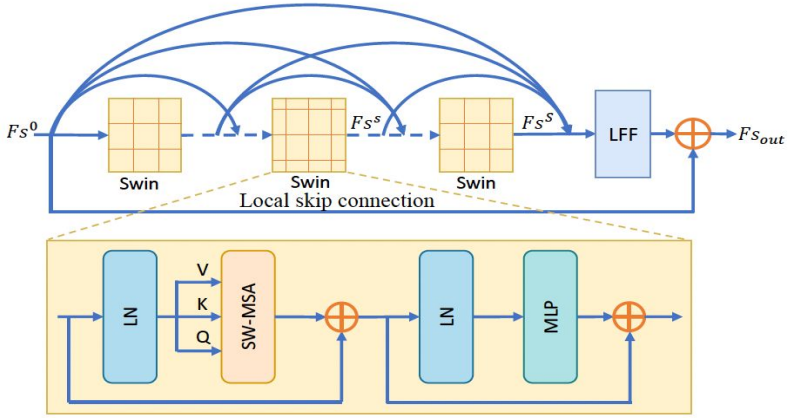
Dataset

- Used BCR dataset
- BCR: blender cycles ray tracing
- ~ 87% train, ~ 6% test, ~ 7% validation

	Images	Scenes
Total	2449	1463
Train	2126	1283
Test	130	104
Validation	193	76

Network Architecture

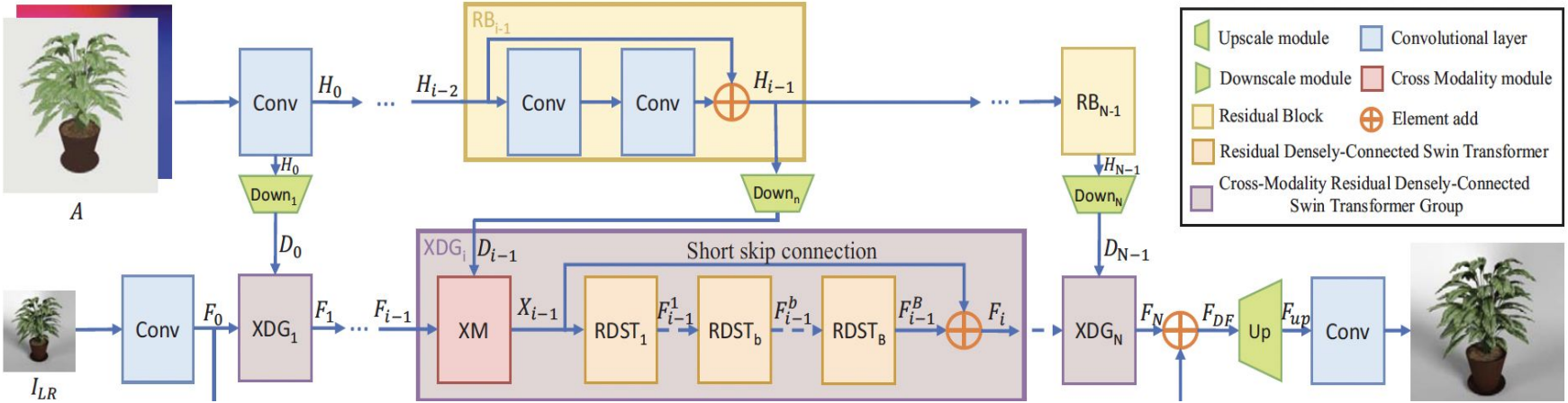




- F is the low-resolution rendering branch
- D is the auxiliary feature branch which contains high-resolution information

- Swin layers allow for learning of more descriptive features by shifting windows
- Skip connection helps fuse the features

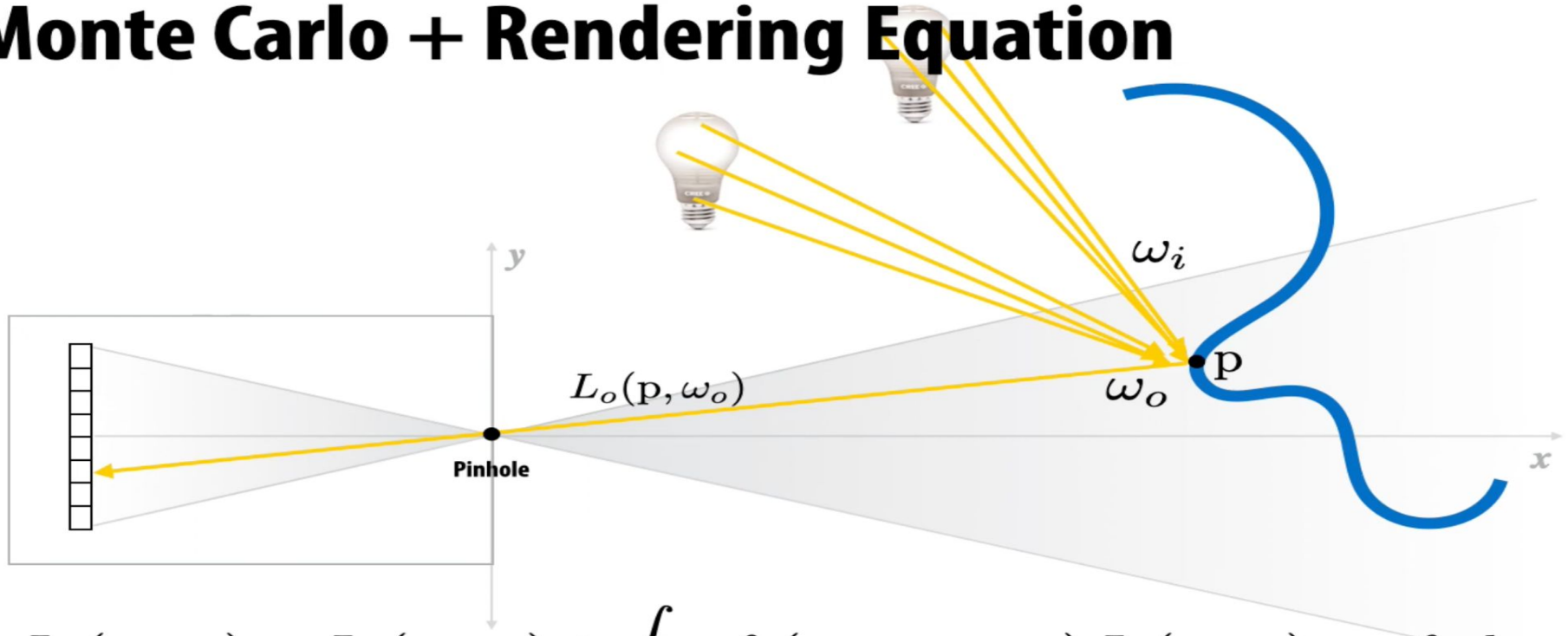
Network Architecture (continue..)



- Downscale auxiliary features
- Upscale low resolution features

How does ‘Monte Carlo Rendering’ work?

Monte Carlo + Rendering Equation

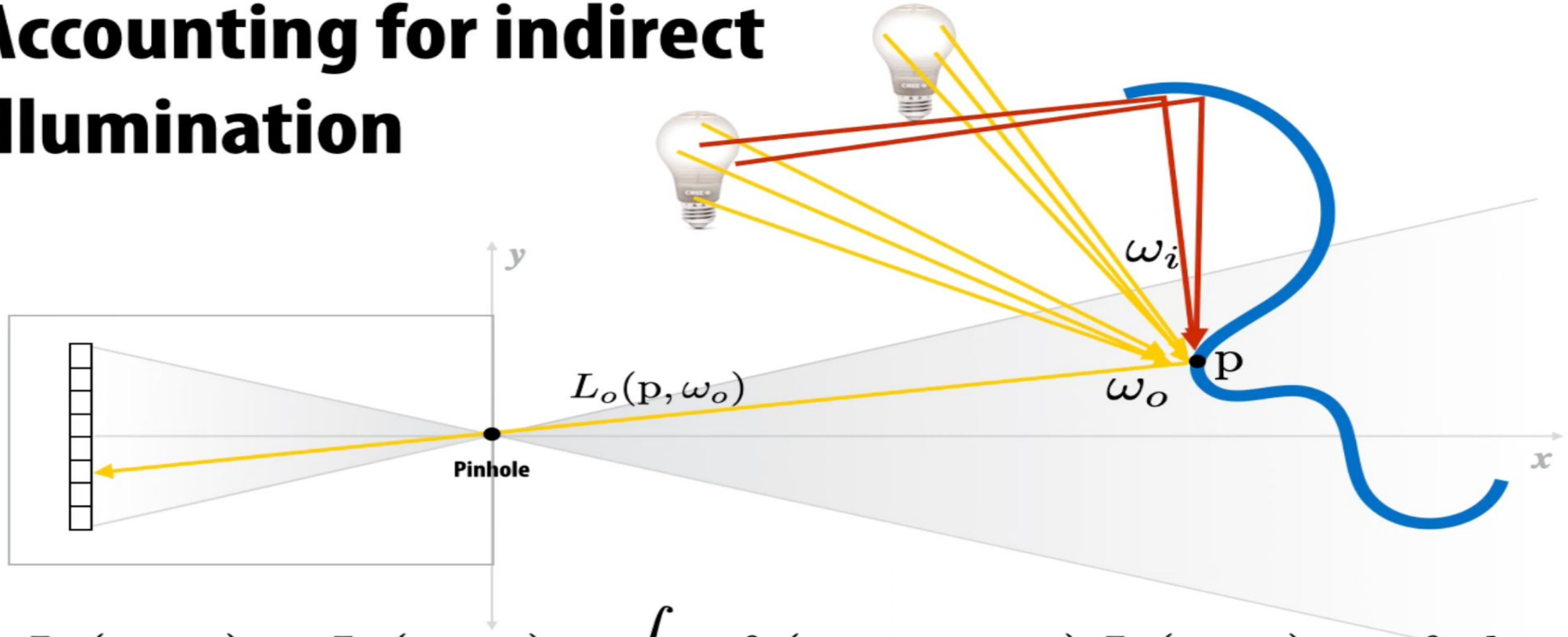


$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_o)\, L_i(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

Need to know incident radiance.

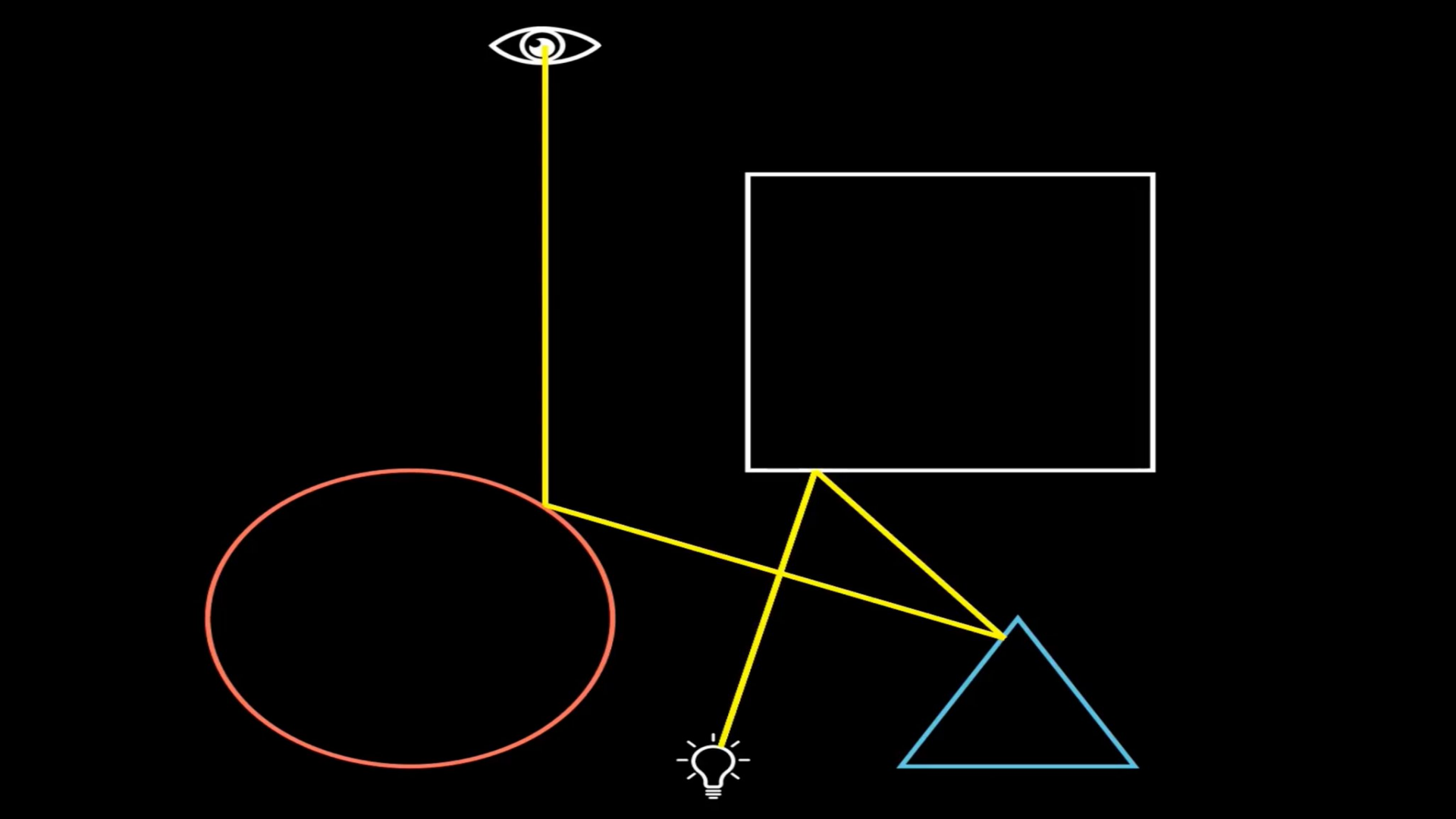
So far, have only computed incoming radiance from scene light sources.

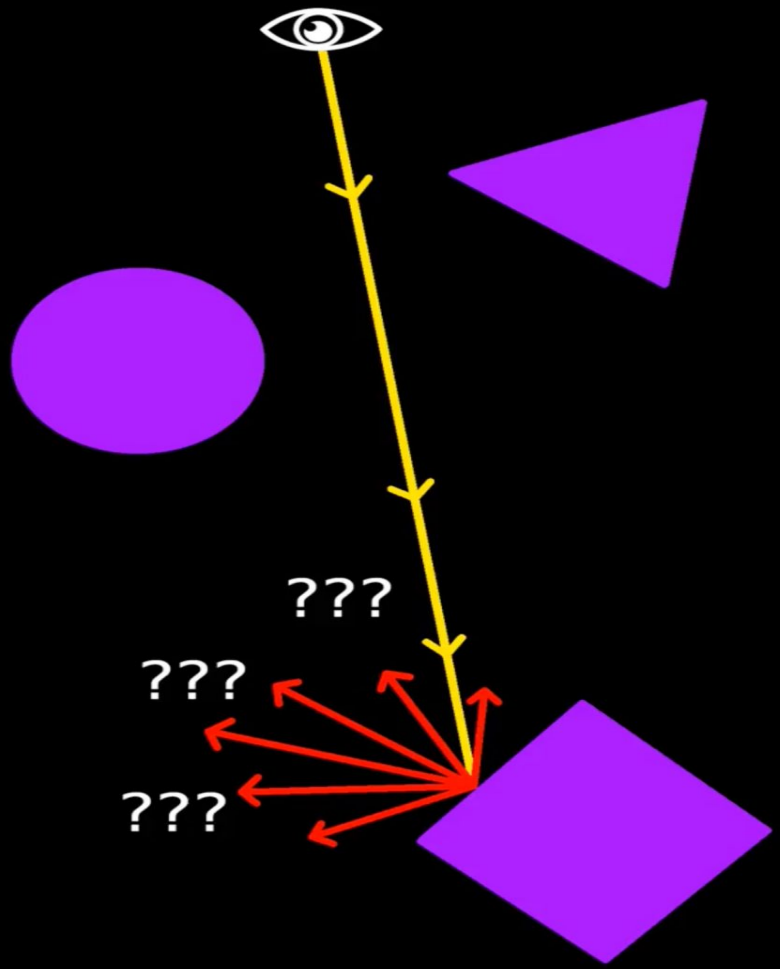
Accounting for indirect illumination



$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_o)\, L_i(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

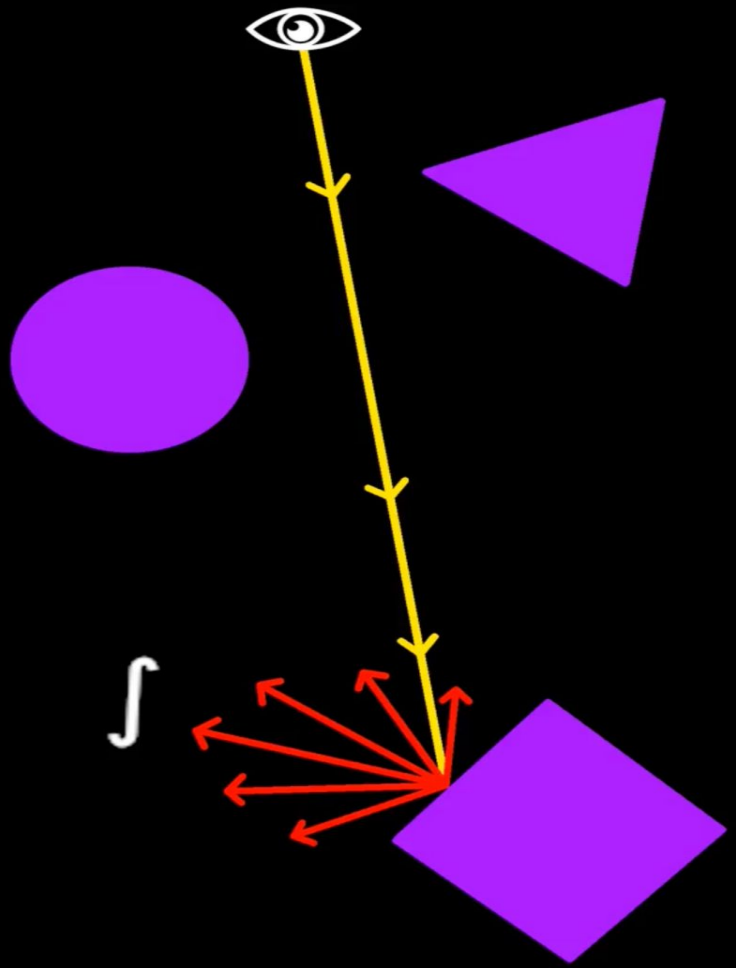
Incoming light energy from direction ω_i may be due to light reflected off another surface in the scene (not an emitter)

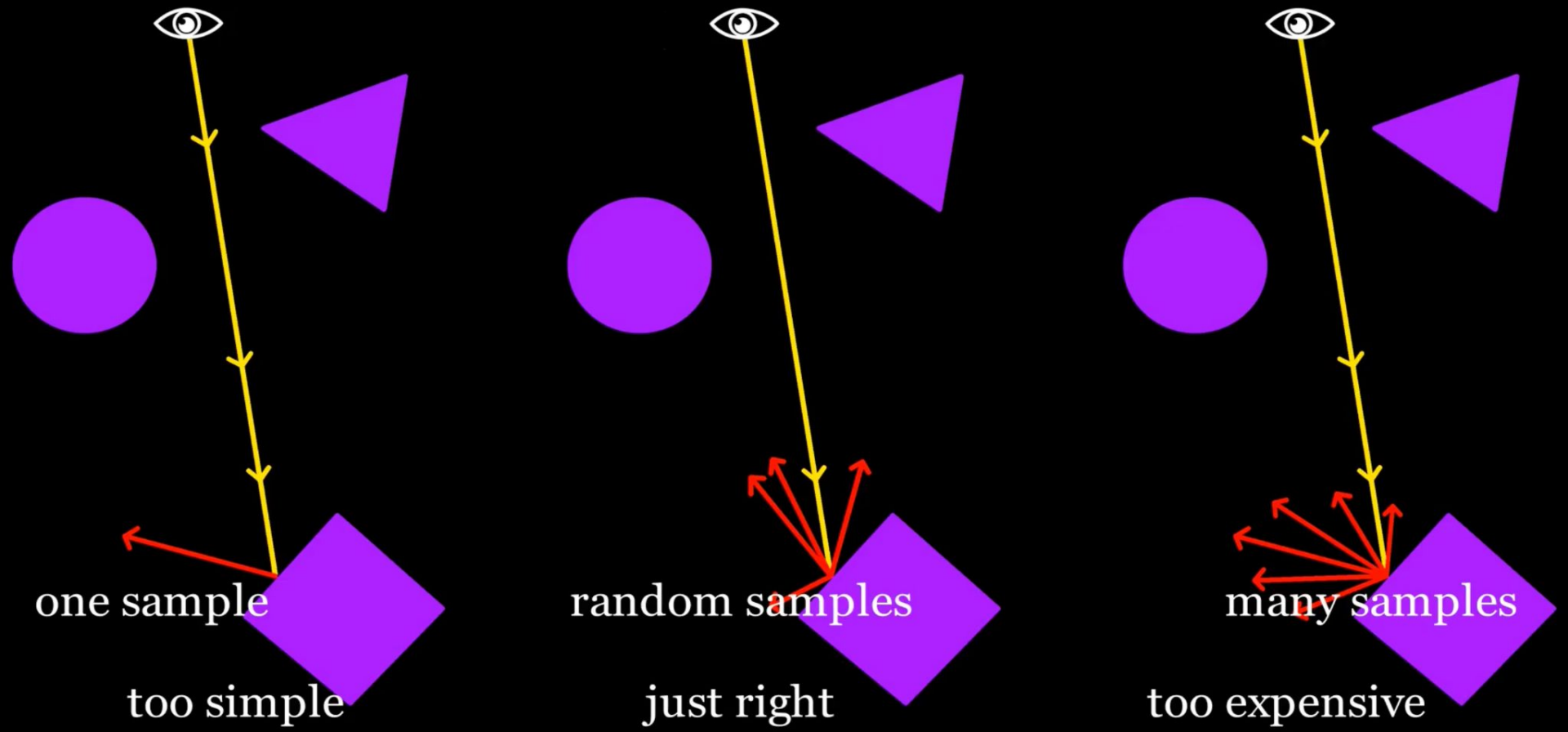


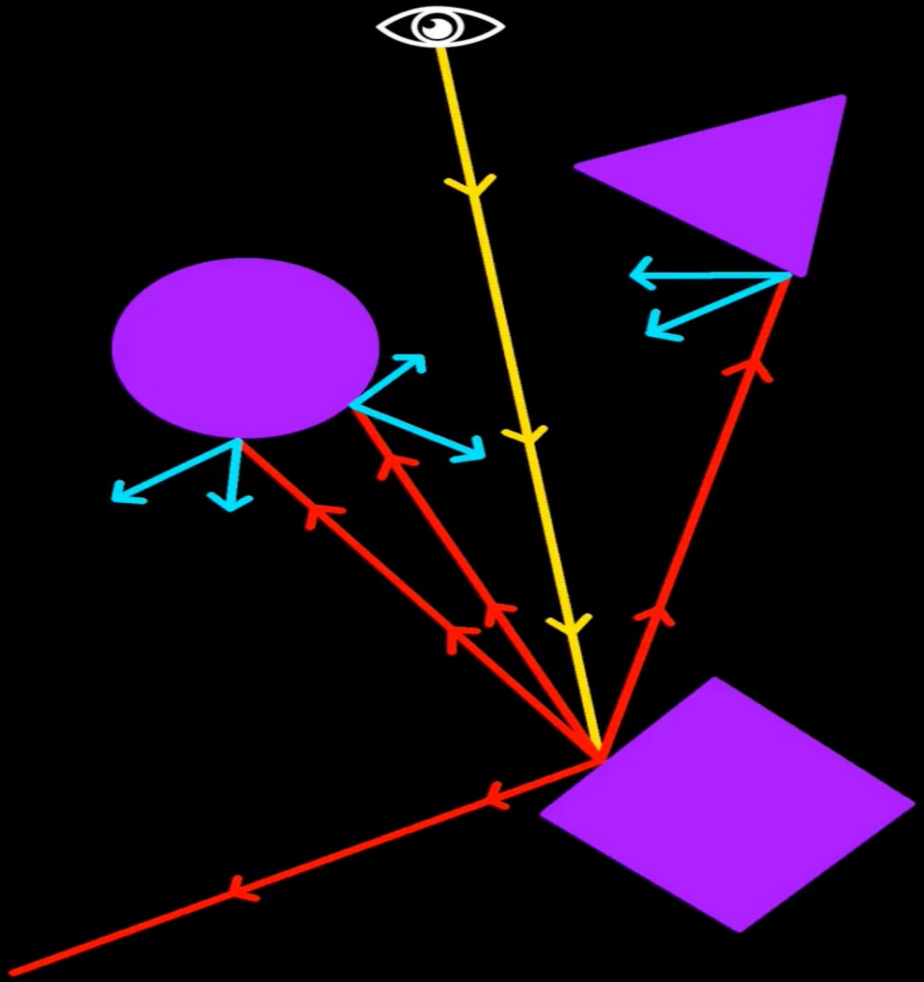


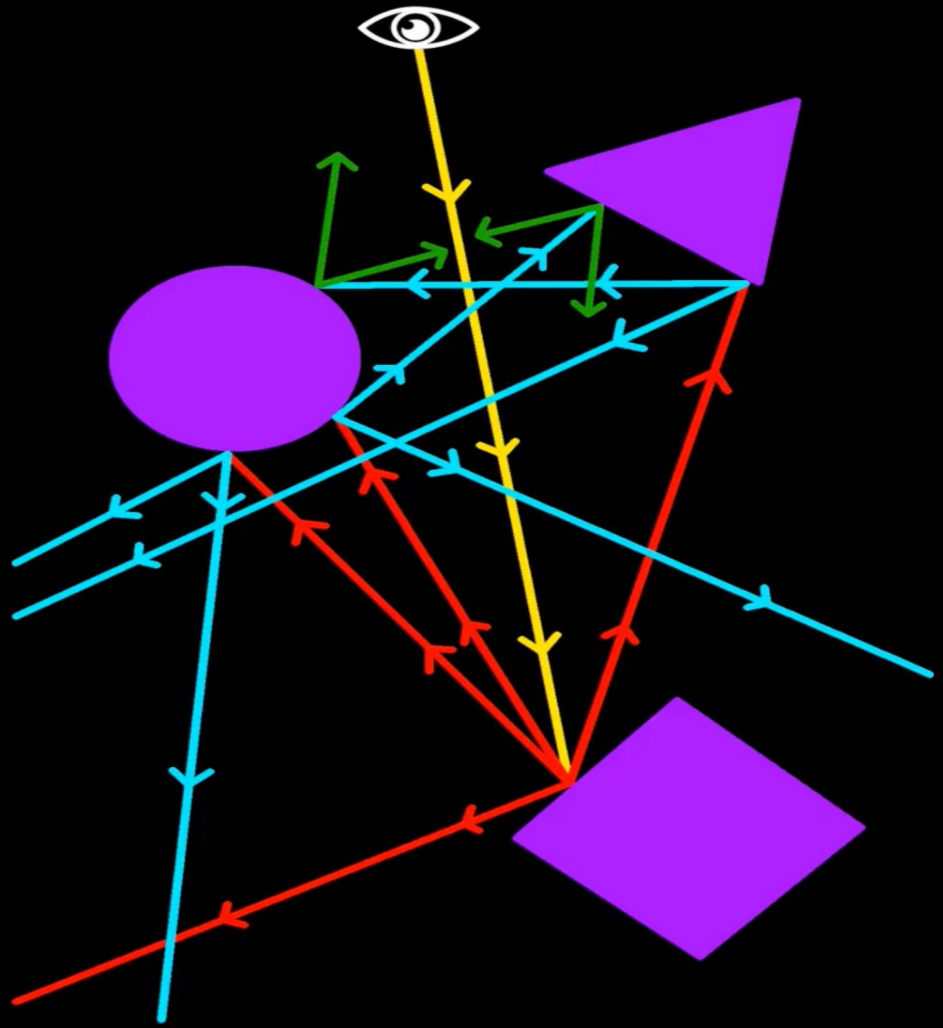
$$I = \int_{\Omega} f(\overline{\mathbf{x}})\, d\overline{\mathbf{x}}$$

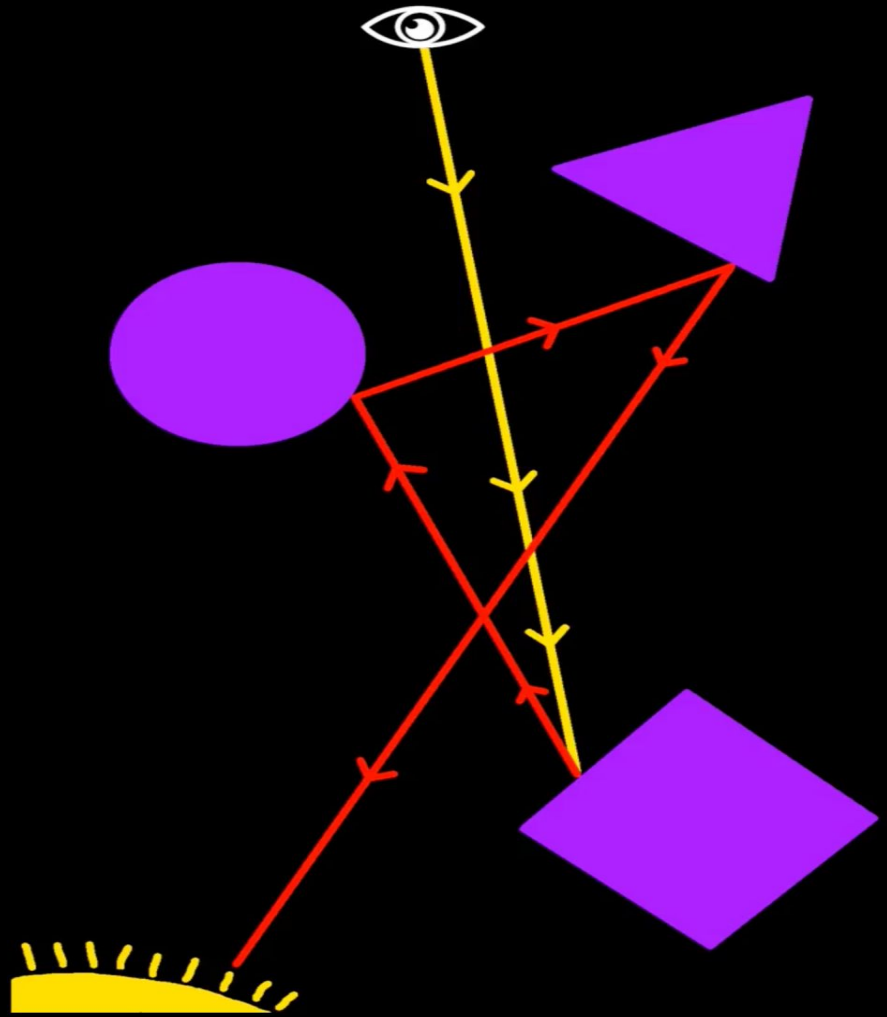
$$I \approx Q_N \equiv V \frac{1}{N} \sum_{i=1}^{N} f(\overline{\mathbf{x}}_i) = V \langle f \rangle.$$




one sample
too simple
random samples
just right
many samples
too expensive







Comparison

Super Resolution

- Generally uses ‘Rasterization’
- Focuses mainly on increasing number of pixel in the resolution.
- Could be biased

Denoising Methods

- Low ‘Sampling Rate’
- ‘Curse of Dimensionality’
- Could be biased

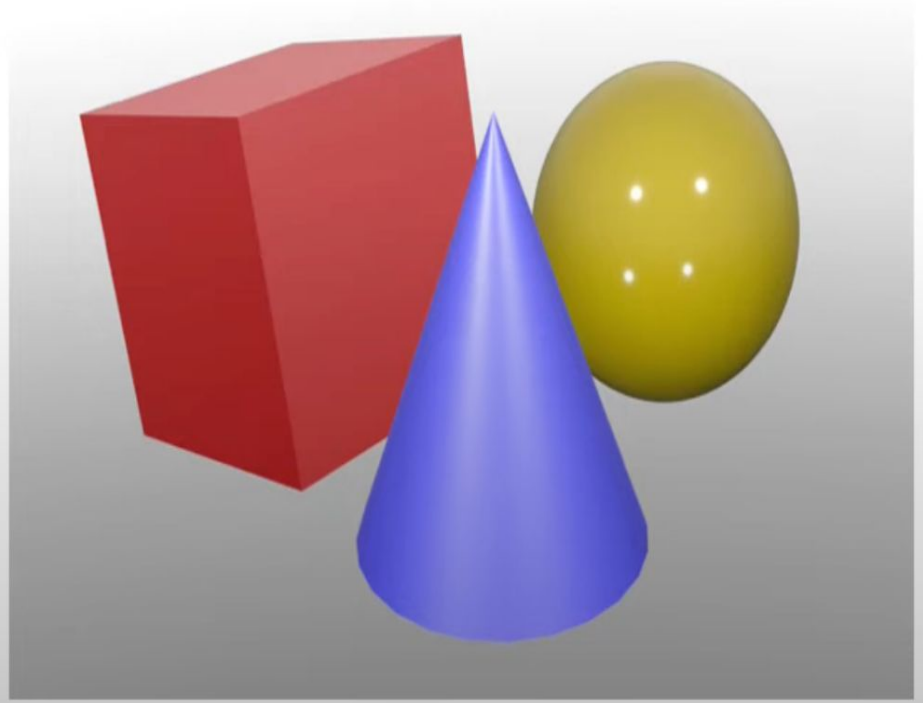
Deep Learning Models

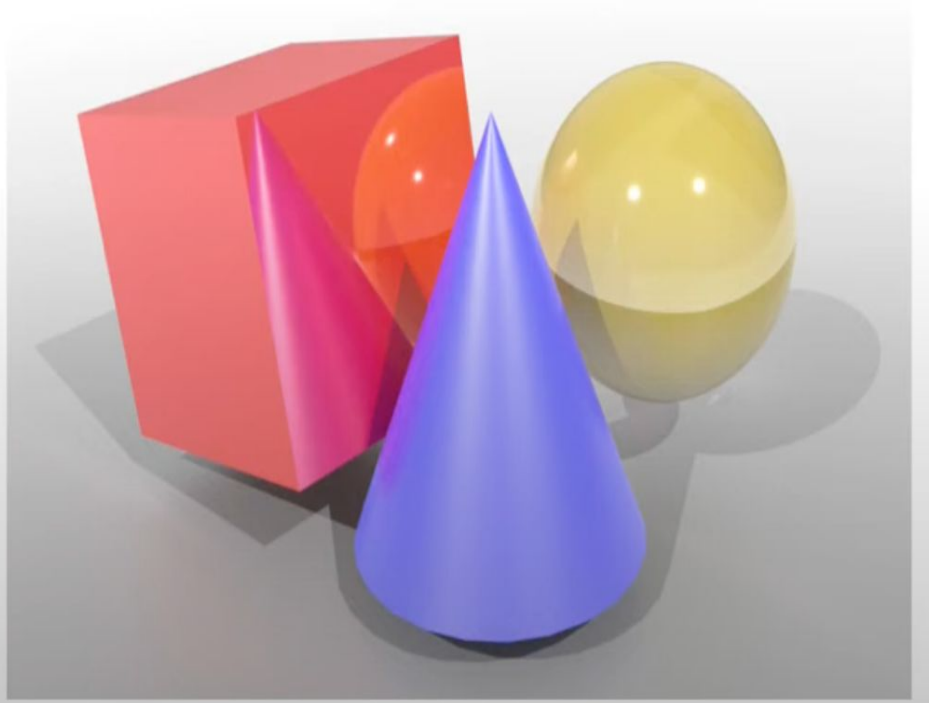
- Requires historic data

Why Monte Carlo Method?

- Reduces the ‘Rendering Time’
- Higher ‘Sampling Rate’
- Benefit of- ‘Ray Tracing’
- Reduces the ‘Curse of Dimensionality’
- Unbiased
- Better ‘Smoothing’
- Variance is always linear With increase in number of experiments, variance is going to be 0 (means avg. value is approaching the expected value).

Comparison





Demonstration

Direct illumination

One-bounce global illumination

Two-bounce global illumination

Four-bounce global illumination

Impact of Monte-Carlo on the Field

- 3D rendering is extremely important in modern times with the push for the “metaverse”
- Many various applications
  - Video games
  - Animation
  - Medical research
  - Flight/driving simulation
- **Increasing speed and quality of renders can open doors for even more applications**

Improvements on Monte Carlo Tracing

- Paper is vague, but appears to be unbiased Monte Carlo estimation
- Result is a low-res image where each pixel is the MC result
- Sampling rate is between 1 and 250 spp
- We can make improvements using methods covered in class

Results

- The best performance of this model was seen with a 4:1 ratio of MC spp to auxiliary spp
- This MC/ML system outperforms others in…
  - Image quality
  - Information required
- Does not beat them in…
  - Runtime speed
  - Memory required

Method	2spp		4spp		8spp	
	PSNR	RelMSE	PSNR	RelMSE	PSNR	RelMSE
Input	18.12	0.2953	21.51	0.1400	24.75	0.0646
KPCN	25.87	0.0390	27.31	0.0299	28.11	0.0276
KPCN-ft	31.03	0.0078	33.69	0.0043	35.83	0.0026
Bitterli	26.67	0.0293	27.22	0.0252	27.45	0.0226
Gharbi	30.73	0.0068	31.61	0.0057	32.29	0.0050
MSSPL×2	33.27	0.0044	35.15	0.0027	36.74	0.0019
MSSPL×4	33.94	0.0039	35.21	0.0028	36.31	0.0022
MSSPL×8	31.37	0.0075	32.35	0.0057	33.14	0.0049
AdvMC-ft	30.33	-	32.30	-	33.69	-
MCSA-ft	32.68	0.0049	34.81	0.0031	36.61	0.0021
Ours×1	(1 - 1)		(2 - 2)		(4 - 4)	
	31.04	0.0078	34.67	0.0030	36.62	0.0020
Ours×2	(4 - 1)		(8 - 2)		(16 - 4)	
	34.12	**0.0035**	**35.49**	**0.0026**	**37.09**	**0.0018**
Ours×4	(16 - 1)		(32 - 2)		(64 - 4)	
	34.08	0.0046	35.06	0.0034	35.77	0.0029
Ours×8	(64 - 1)		(128 - 2)		(250 - 4)	
	31.10	0.0095	31.36	0.0093	31.62	0.0093

Table 3: *Comparison on the BCR dataset [HLM*21]. (4 - 1) indicates that our method takes a 4-spp low resolution rendering and 1 spp fast-to-compute auxiliary feature as input. MSSPL also takes the 1-spp high-resolution rendering as input, including the diffusion and specular layers. Our method takes less shading information compared to MSSPL [HLM*21].*

Results

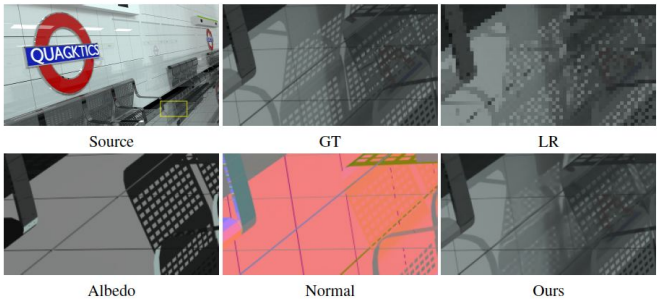
- The best performance of this model was seen with a 4:1 ratio of MC spp to auxiliary spp
- This MC/ML system outperforms others in…
  - Image quality
  - Information required
- Does not beat them in…
  - Runtime speed
  - Memory required

	Scale	EDSR	RCAN	SwinIR	MSSPL	Ours
Runtime(ms)	×4	503.96	280.51	1149.25	125.24	1009.08
Peak memory(MB)	×4	2493.9	672.1	806.0	739.70	941.3
Peak memory(MB)	×8	2375.6	621.3	659.0	1010.1	803.8
Peak memory(MB)	×16	2359.7	615.4	608.4	1008.0	783.4

Table 2: *Comparison of runtime cost and peak memory with super resolution methods to produce a* 1024×1024 *image on an Nvidia Titan XP.*

Limitations and Future Work

- Compared to CNN based models, our method is very slow.
- Compared to Transformer based method, our method uses less peak memory and is considerably faster.
- The fusion for high-reflection part is challenging as albedo and normal in those areas fail by nature.
- The performance of our method can be further improved by using transformer hardware accelerators.
- Exploring other auxiliary features, such as whitted ray traced layer, may potentially improve our results.



Critiques

- Runtime and memory statistics for preliminaries
  - Albedo and normal computation
  - Monte Carlo generation of low-res input
- Runtime for x8 and x16 should be included
- Runtime is still **not feasible**
  - 1 full second for just the machine learning segment
- A strategy should be supplied for overcoming the new bottleneck on the ML model
- More focus on improvements to the Monte Carlo aspect

Improvements on Monte Carlo Tracing

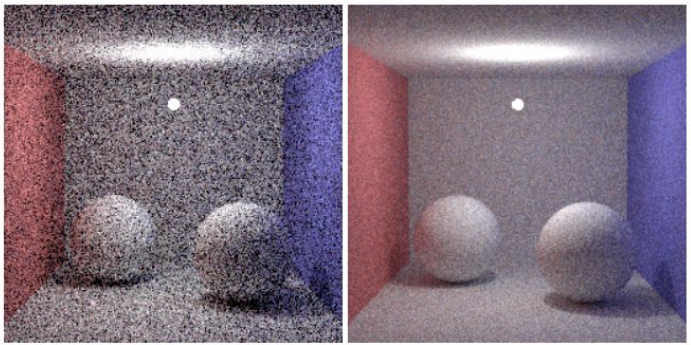
Importance sampling

- Reflective surfaces require many more rays to make accurate estimations of color
- Shape of objects affect how much light bounces off them as well
- By using info from the given objects to render, can create an importance sampling

Improvements on Monte Carlo Tracing

Importance sampling

- Left image: unbiased; 2,500 spp; runtime 10 min
- Right image: importance-sampled; 50 spp; runtime 10 sec



Improvements on Monte Carlo Tracing

Control Variates

- RGB color values
  - Generally positively correlated for well-lit, or very dark, natural scenes
  - Compute one and use as a control variate… diminish work
  - Downside: more colorful means more noisy
- Neighboring pixel values
  - Physical 3D objects have minimal noise… so colors transition smoothly
  - Downside: highly contrasted images would generate more noise
  - Upside: could increase video rendering efficiency

Improvements on Monte Carlo Tracing

Additional Monte Carlo simulation

- Images and objects often have large areas of similar color
- Construct an importance sampling of pixels
- Alleviate some extraneous MC work by identifying which pixels can be copied

References

- Q. Hou, F. Liu, "Auxiliary Features-Guided Super Resolution for Monte Carlo Rendering", Arxiv Oct 20th 2023, pp. 1-12, https://arxiv.org/pdf/2310.13235.pdf
- “Monte Carlo Ray Tracing in 5 minutes,” https://www.youtube.com/watch?v=9NrODC6pbLg
- “Monte Carlo methods for improved rendering,” https://smerity.com/montelight-cpp/
- “Monte Carlo Rendering (CMU lecture)”, https://www.youtube.com/watch?v=FUZJNlRqrAc
- “Rendering Lecture (TU Wien)”, https://www.youtube.com/watch?v=_56eYqYYO6I