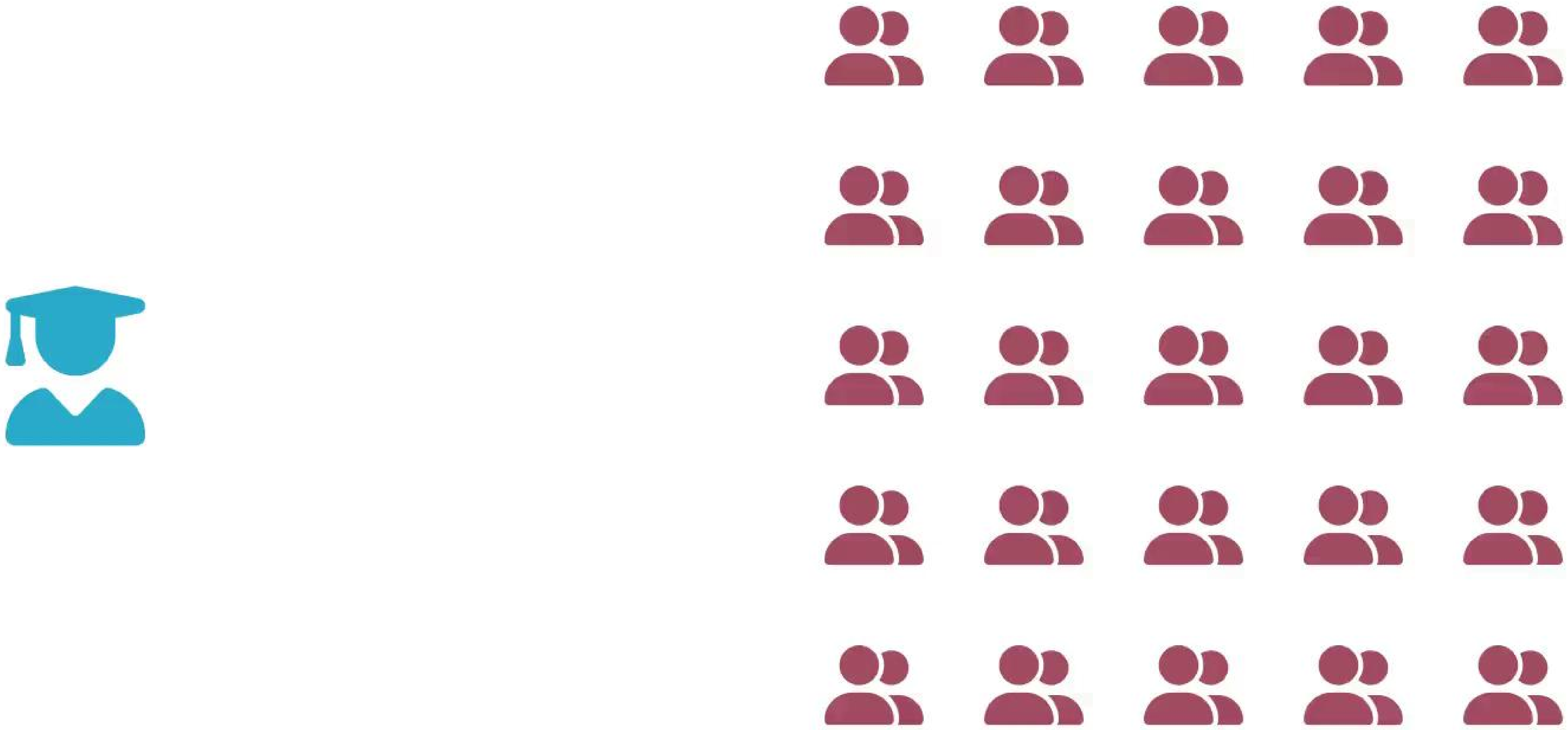
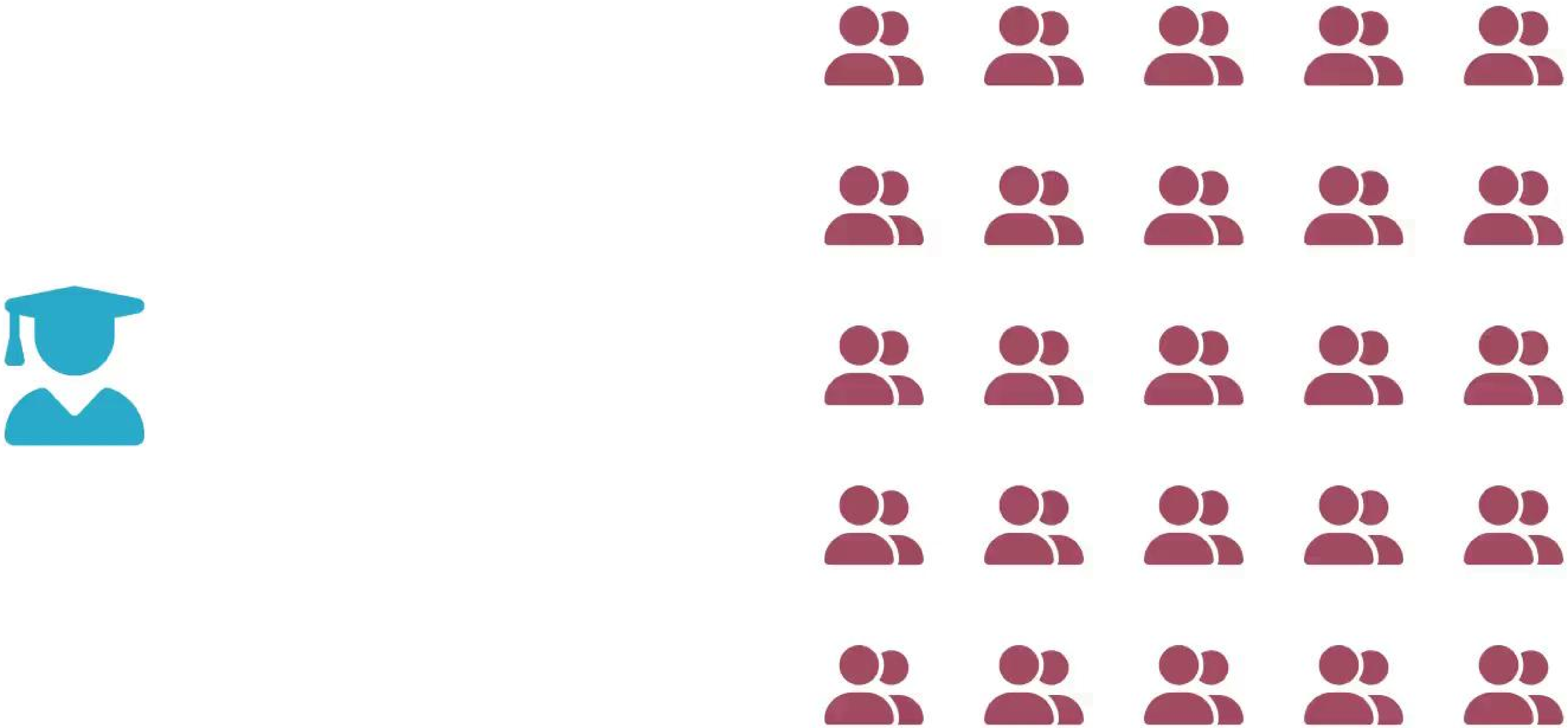
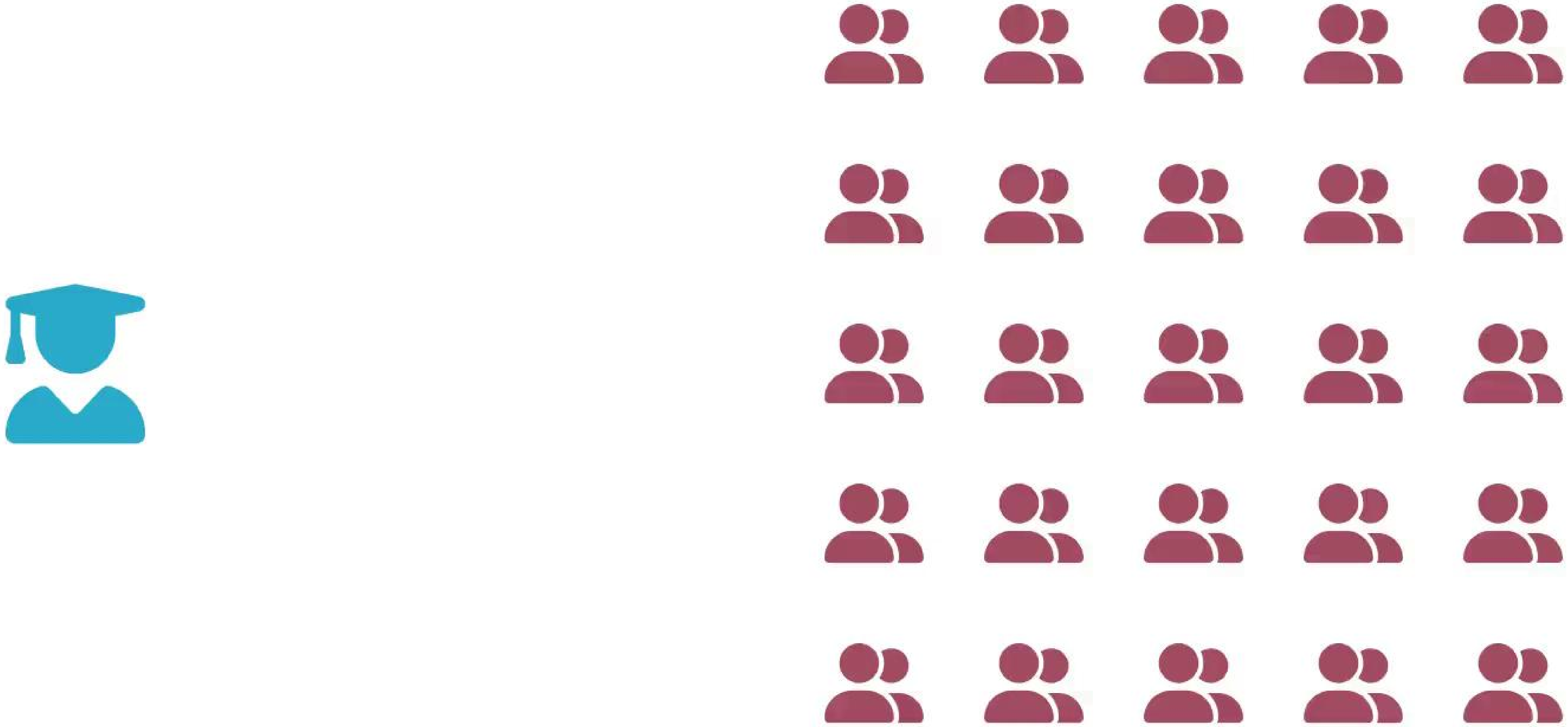
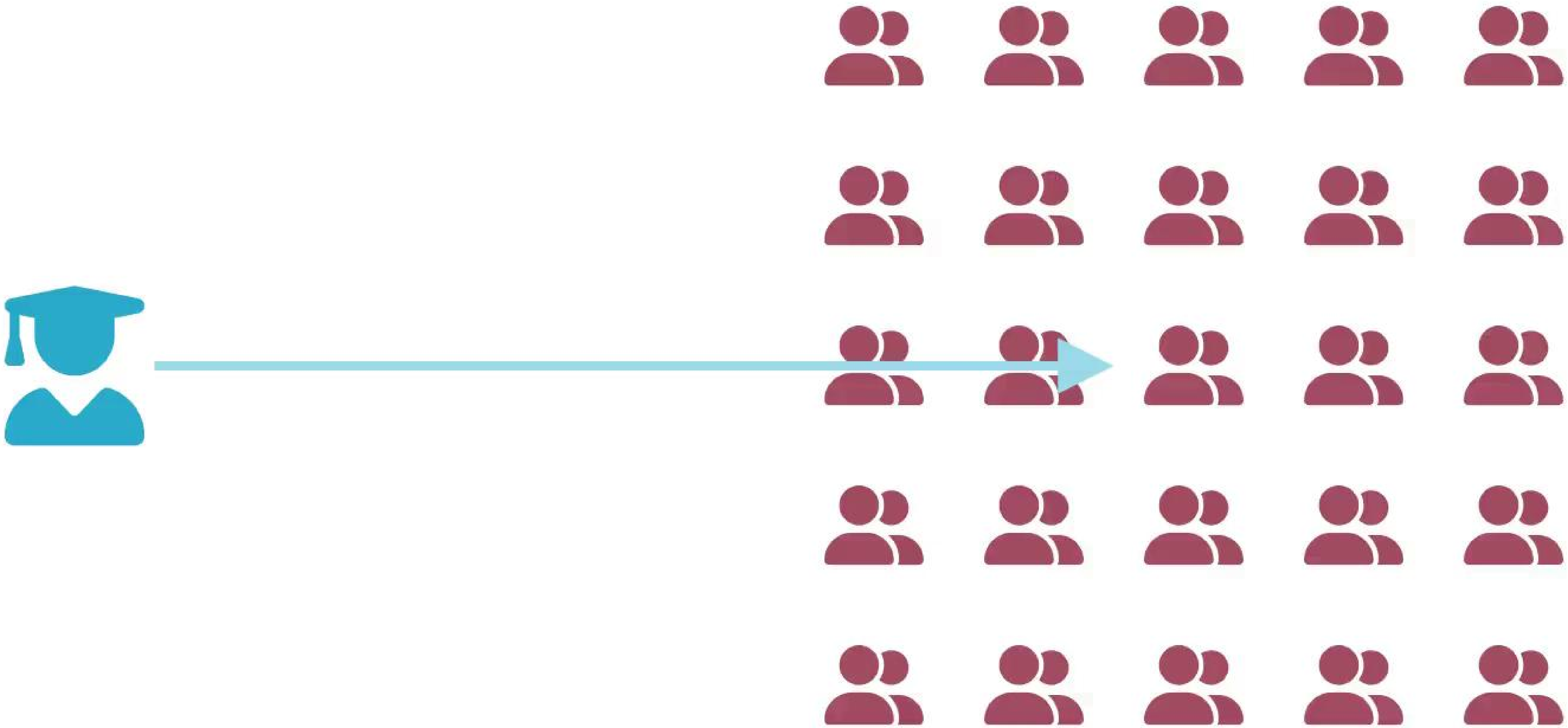
Differentiable Ray Tracing for Telecommunications

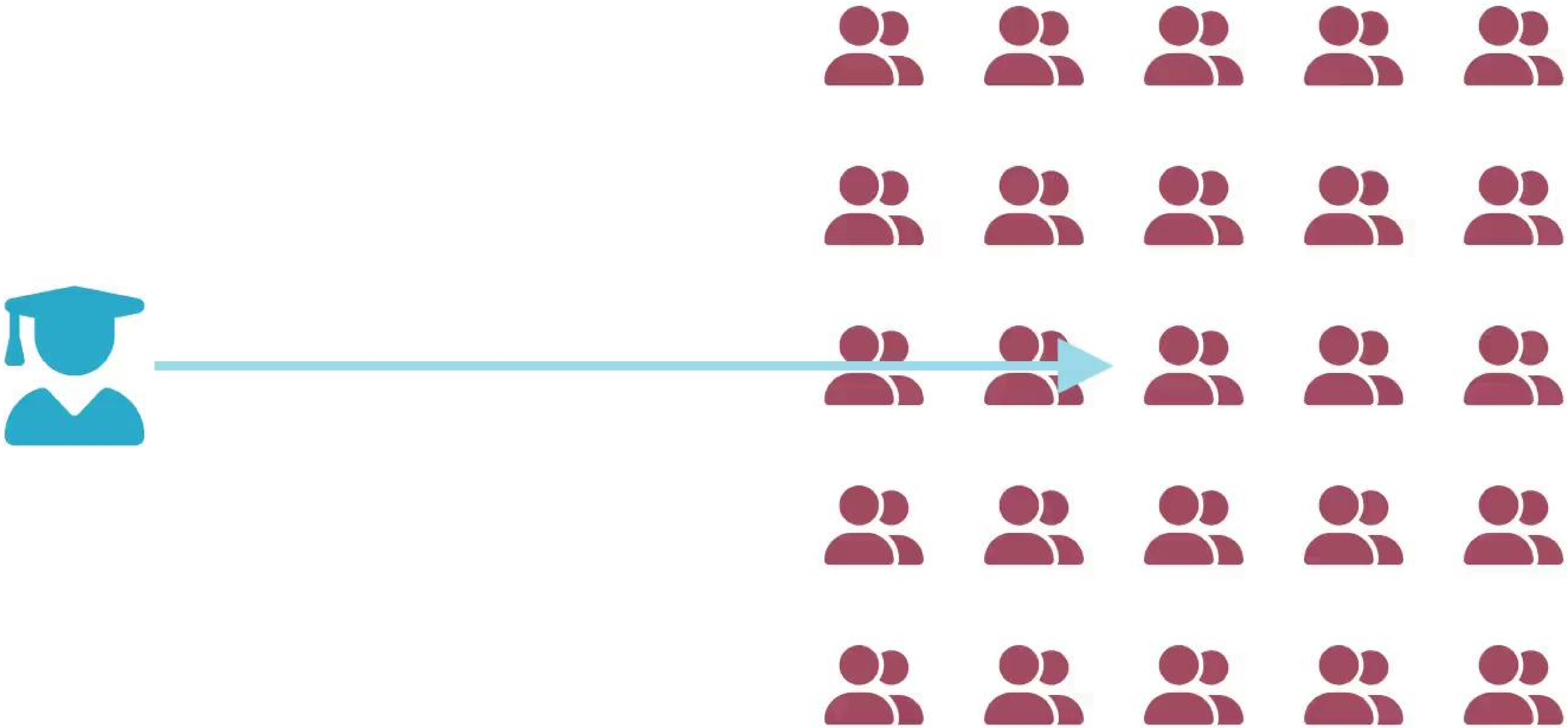
Jérome Eertmans - December 7th 2023

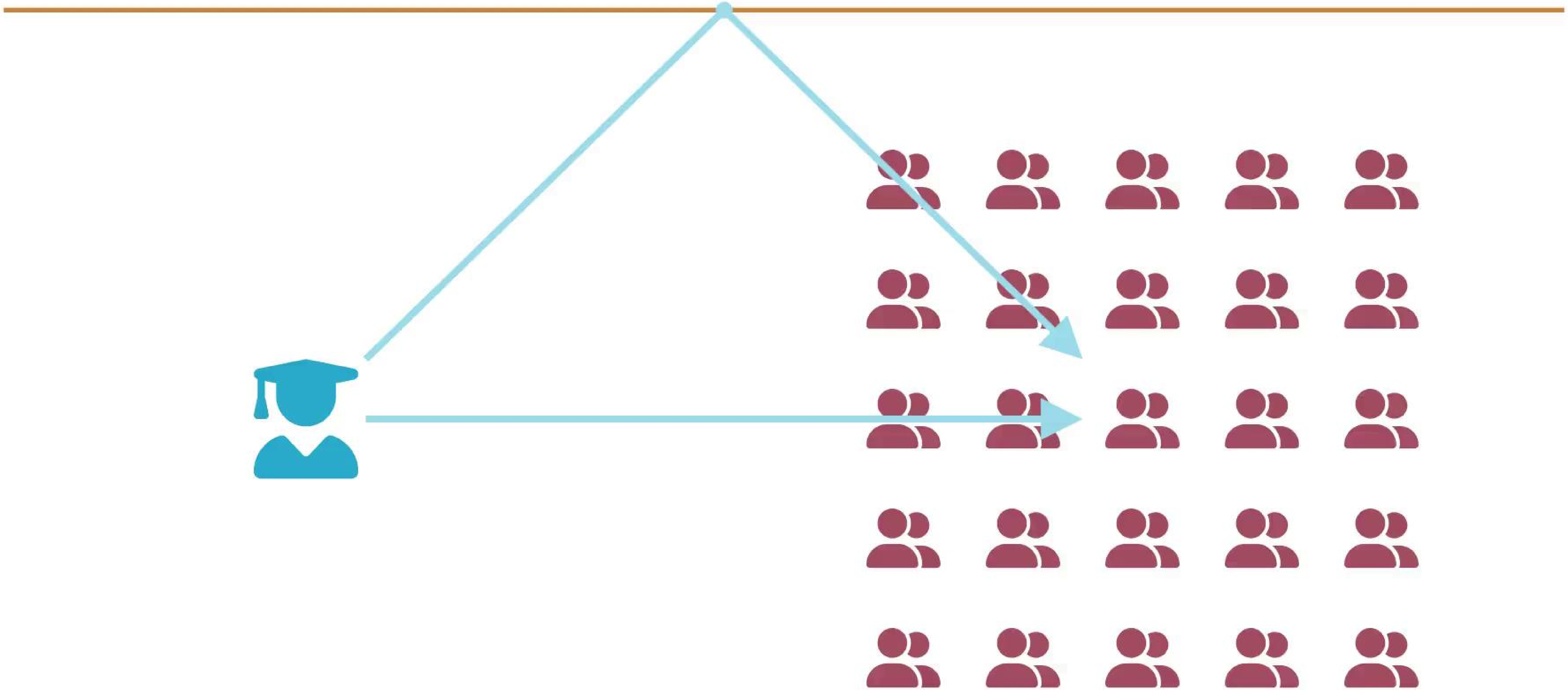


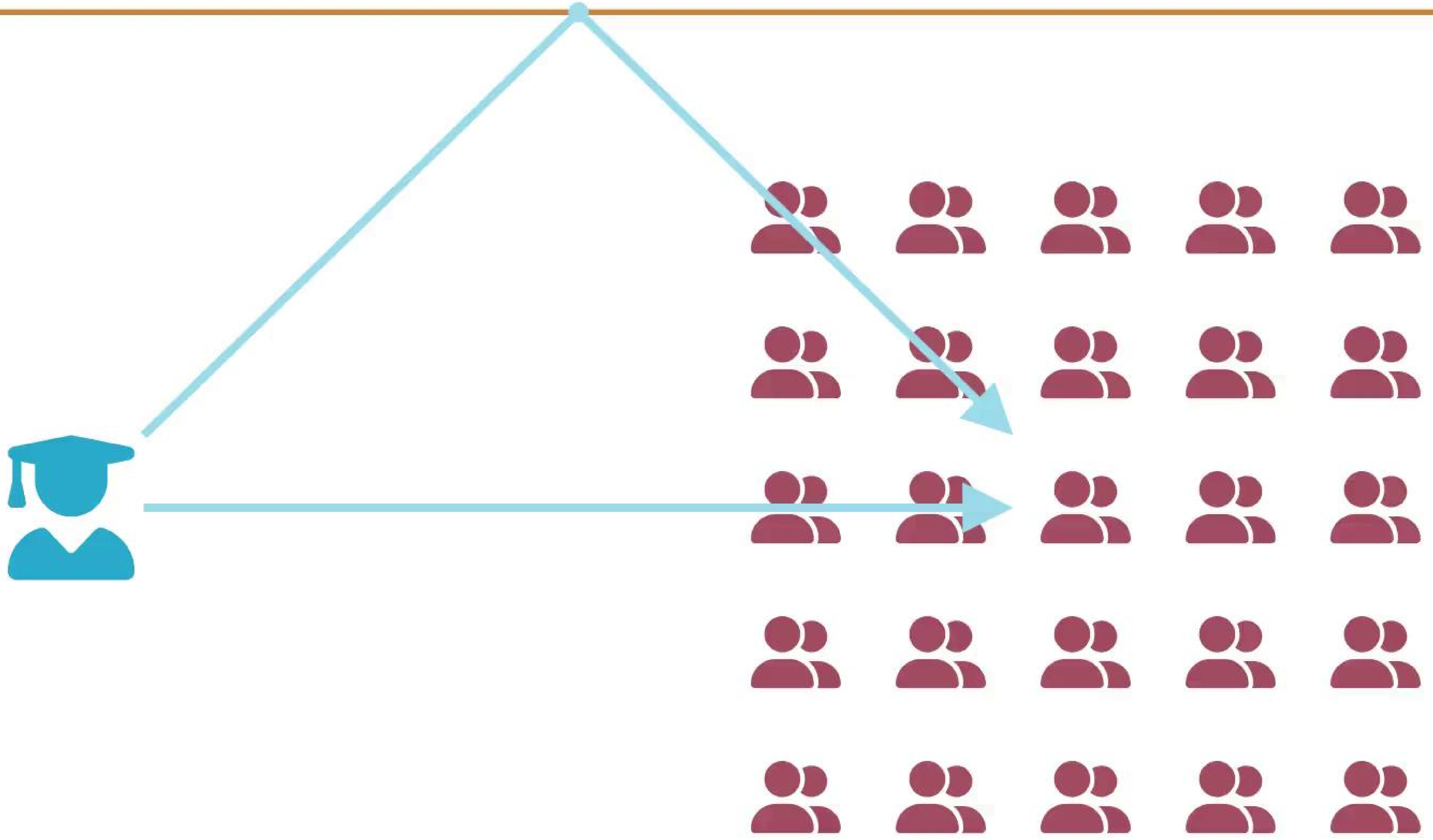


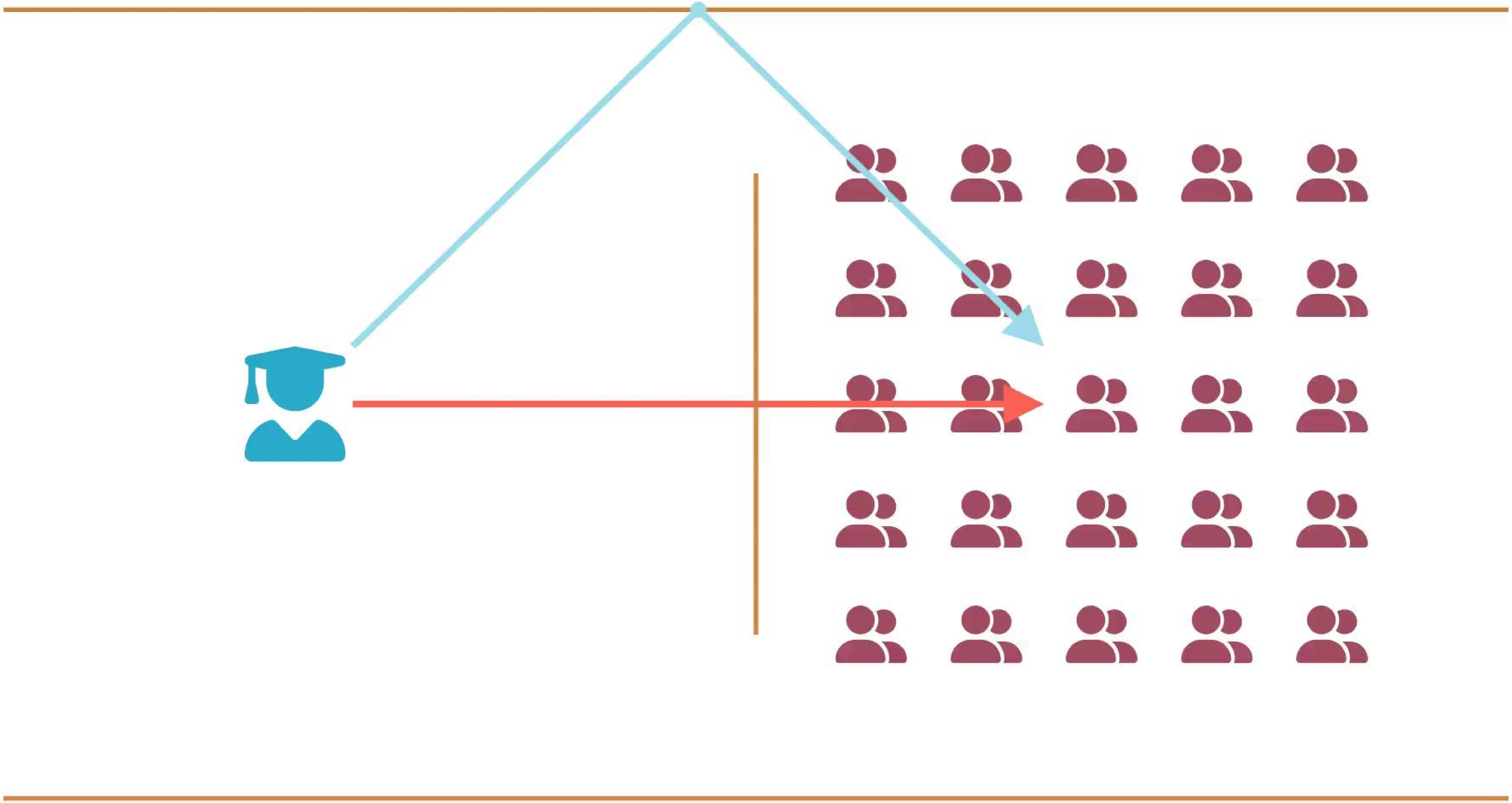


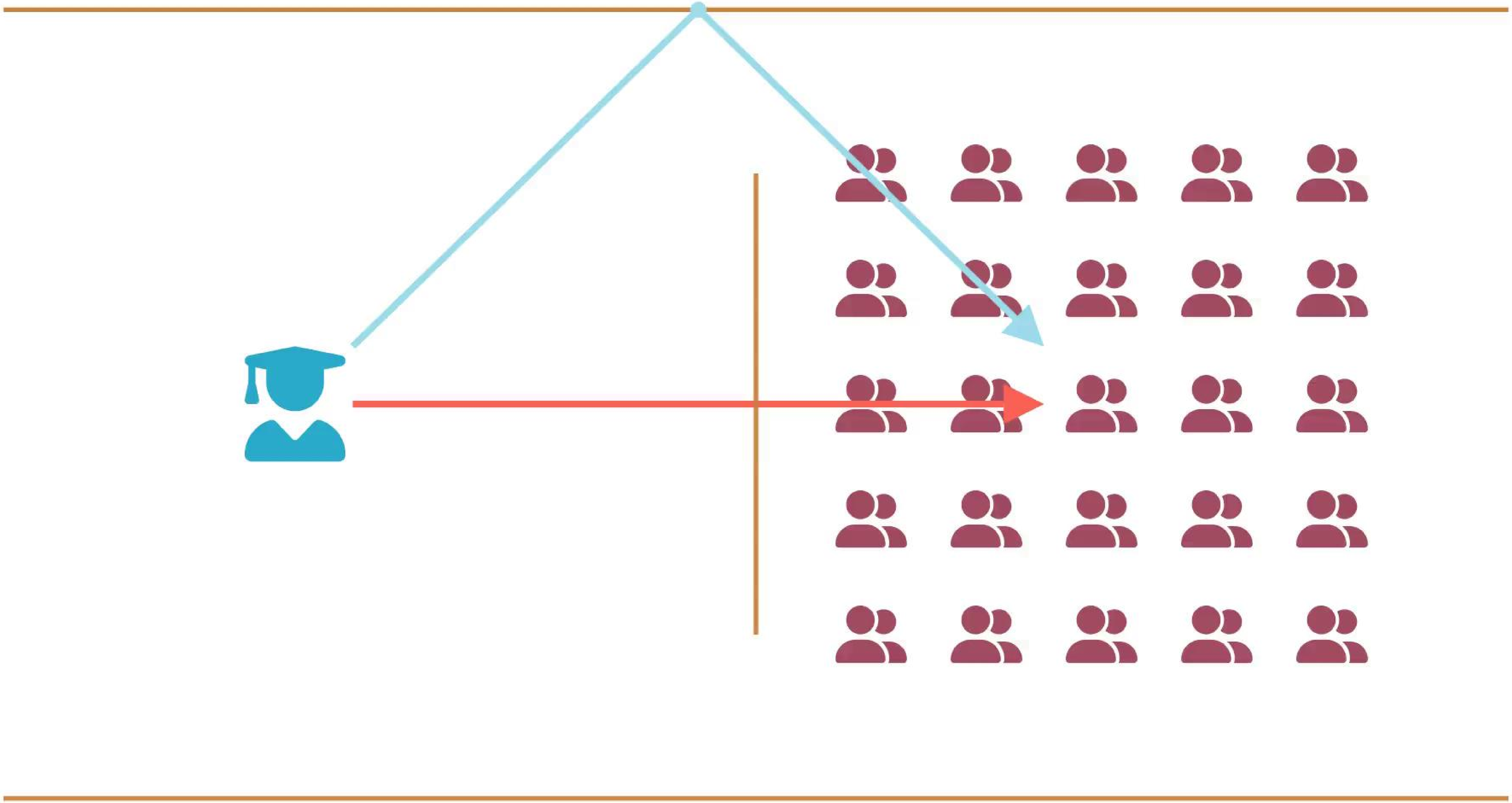


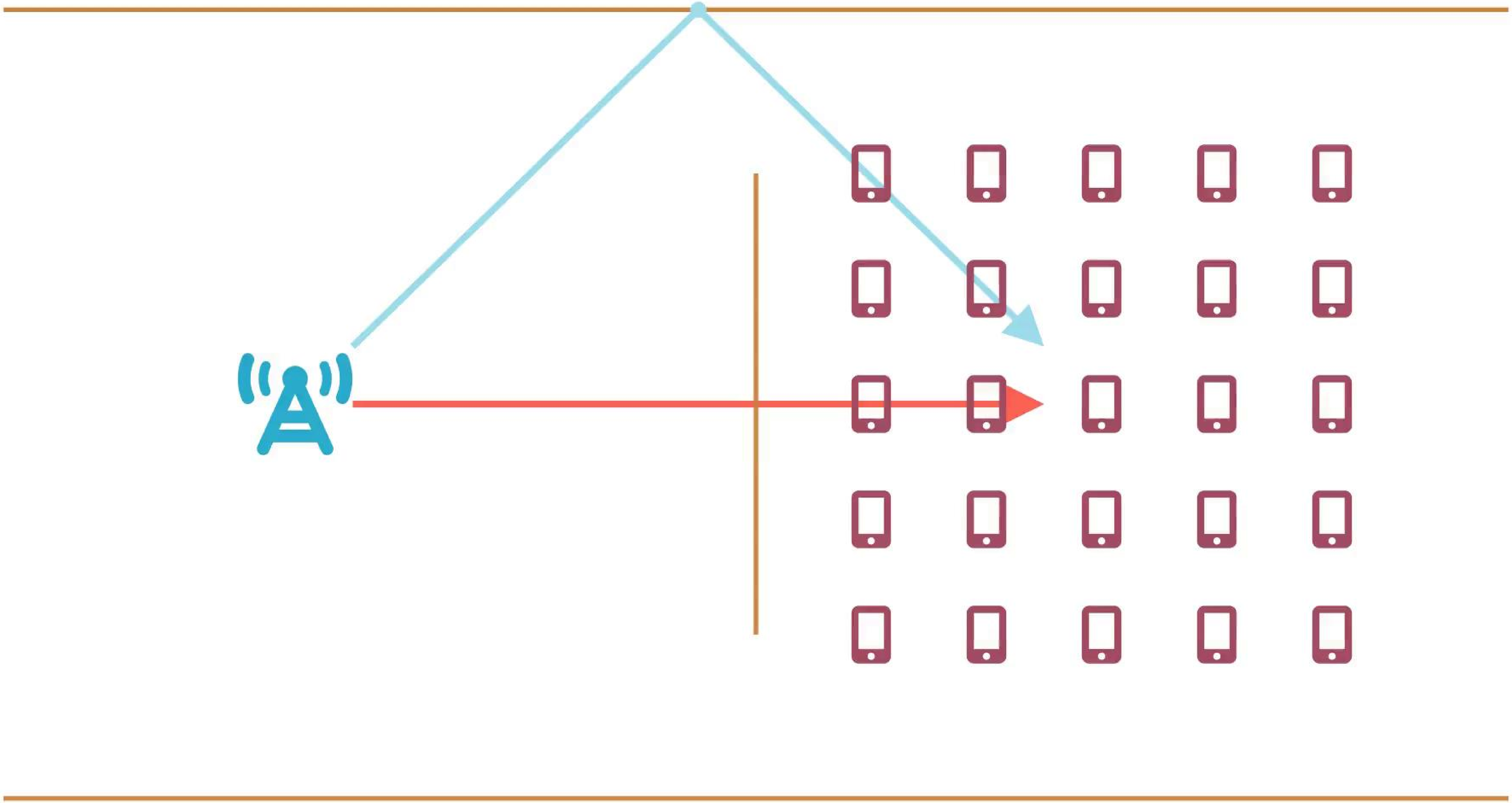


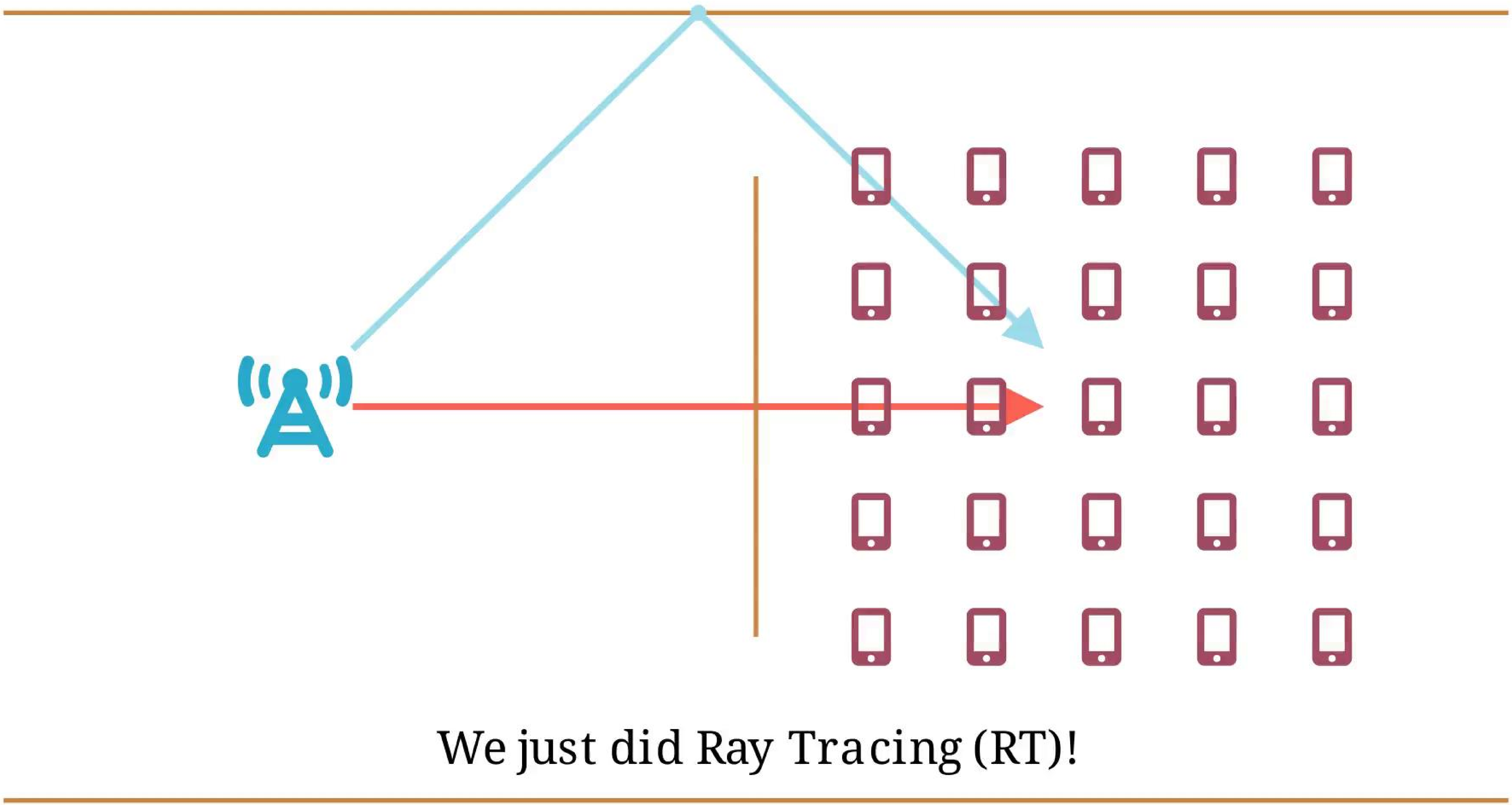












We just did Ray Tracing (RT)!

Contents

1. Ray Tracing and EM Fundamentals;
2. Motivations for Differentiable Ray Tracing;
3. How to trace paths;
4. Differentiable Ray Tracing;
5. Status of Work;
6. and Conclusion.

RT and EM Fundamentals

- Core idea;
- Architecture and Challenges;
- Applications;
- Alternative methods.

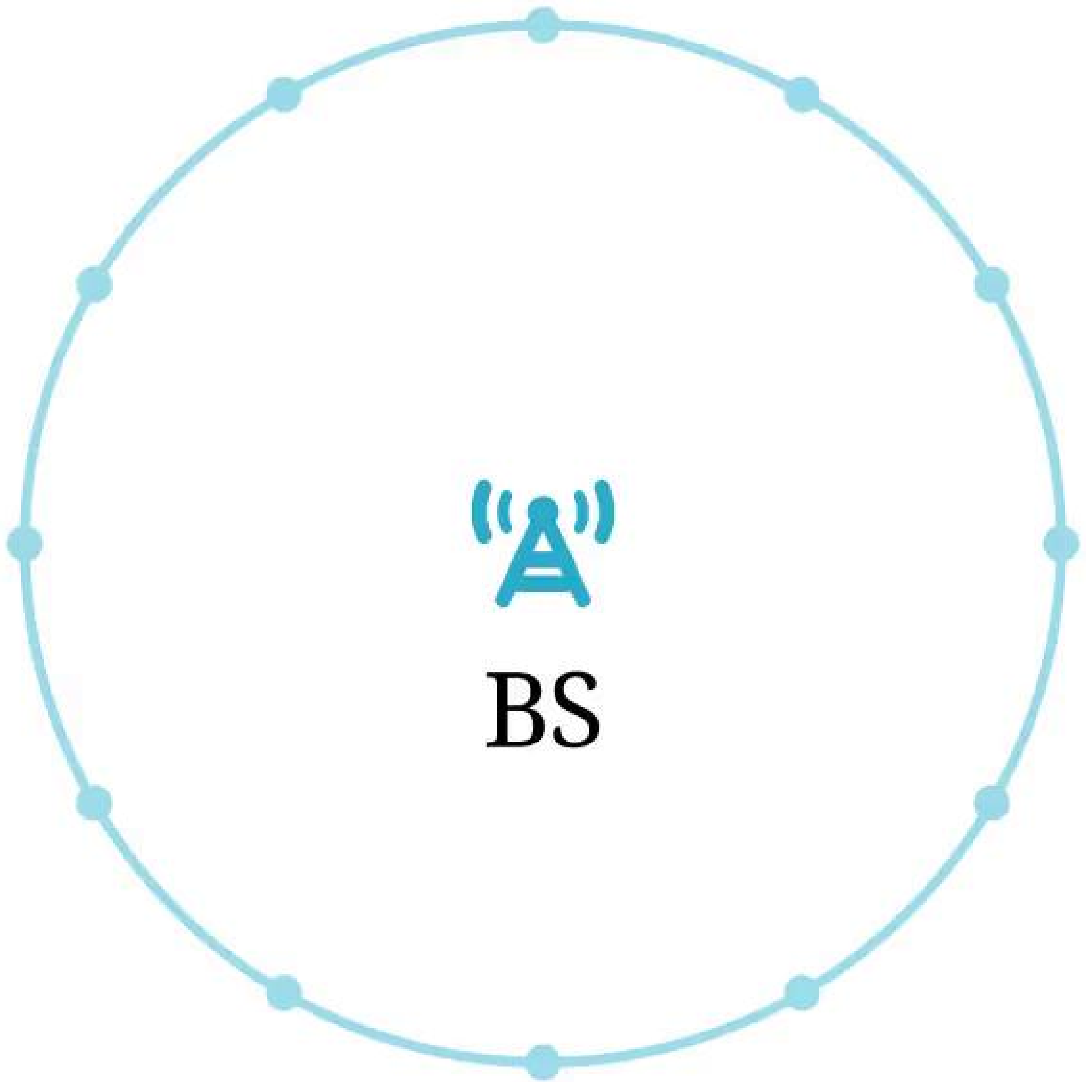
RT and EM Fundamentals

BS

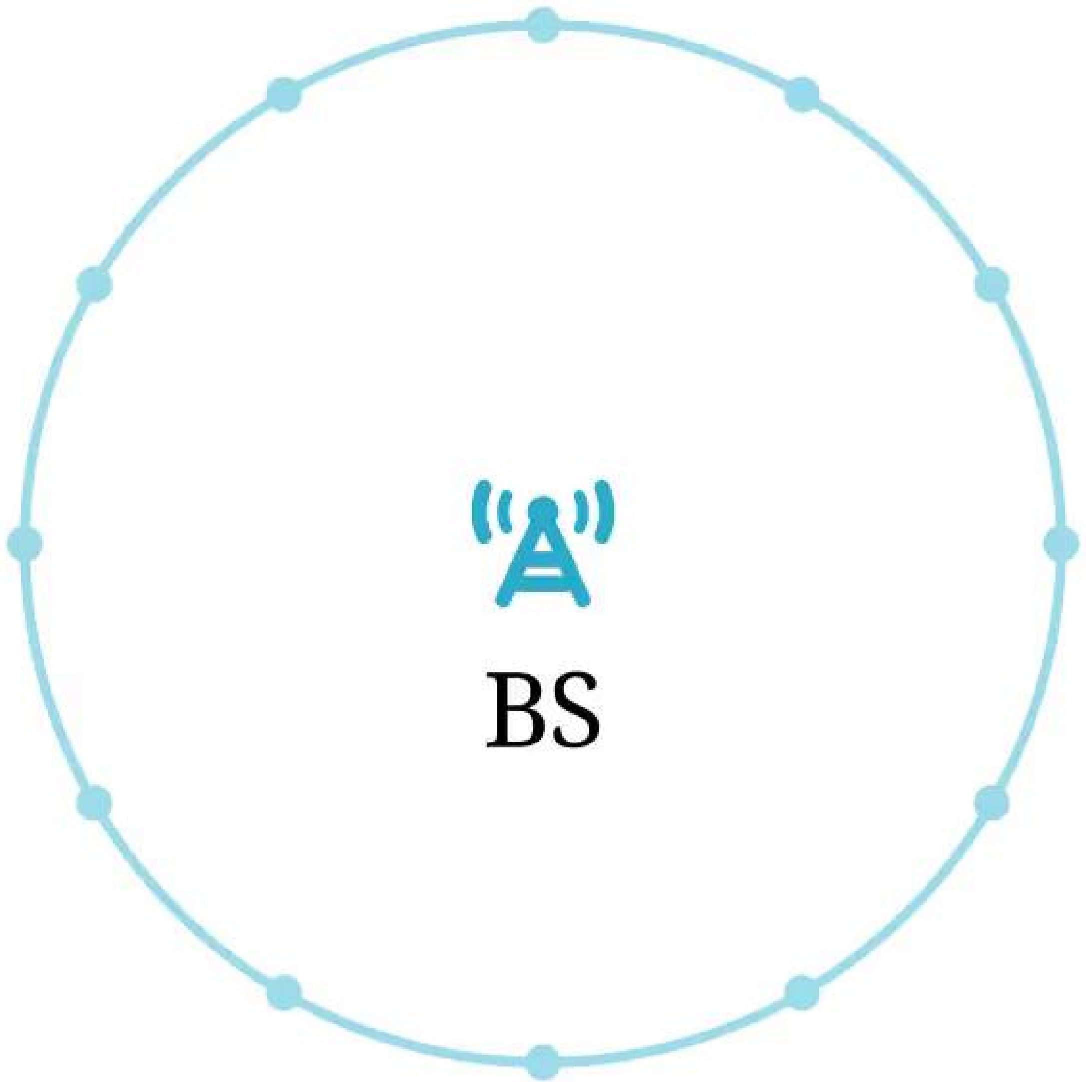
RT and EM Fundamentals

BS

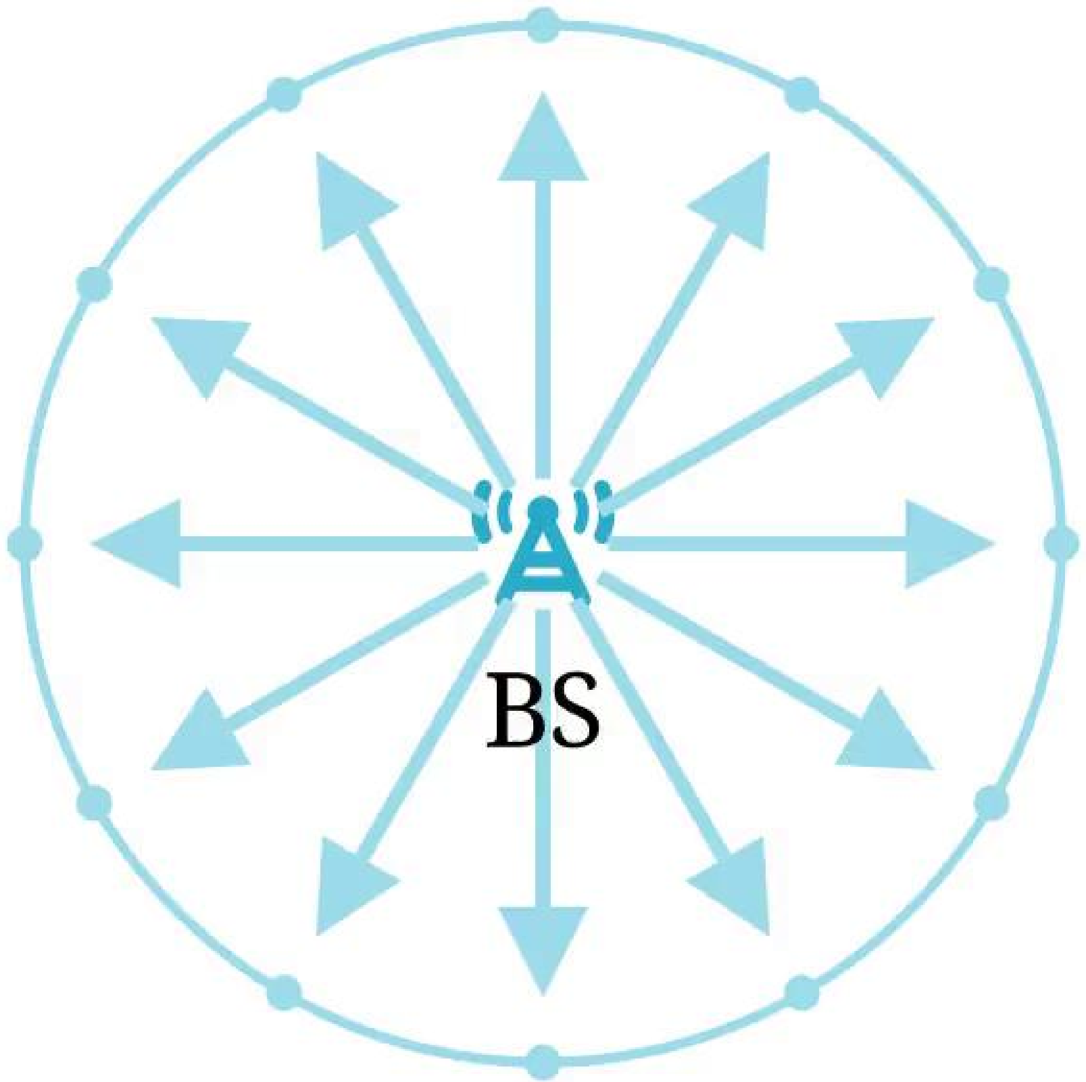
RT and EM Fundamentals

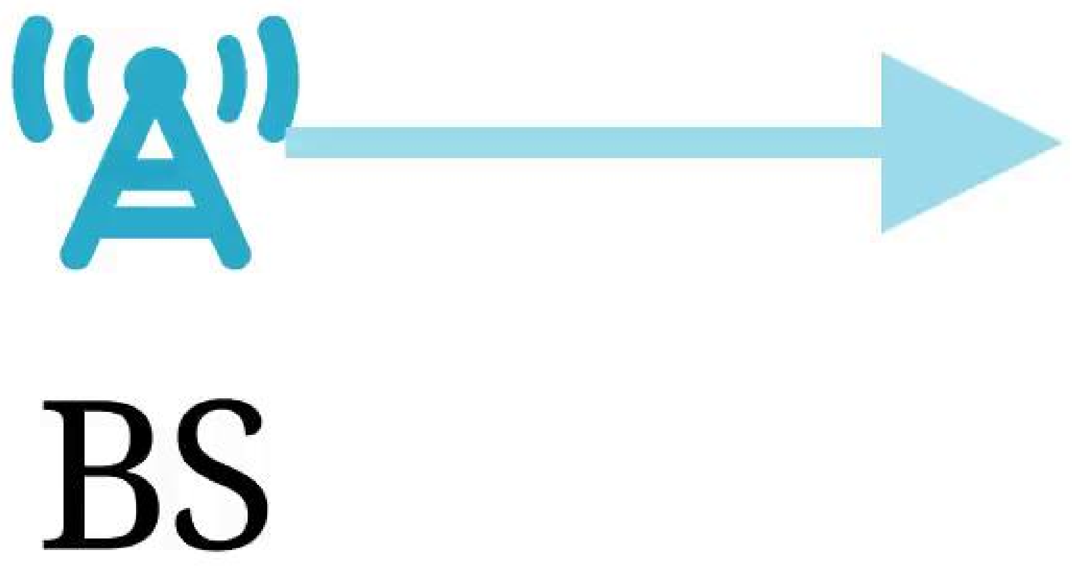


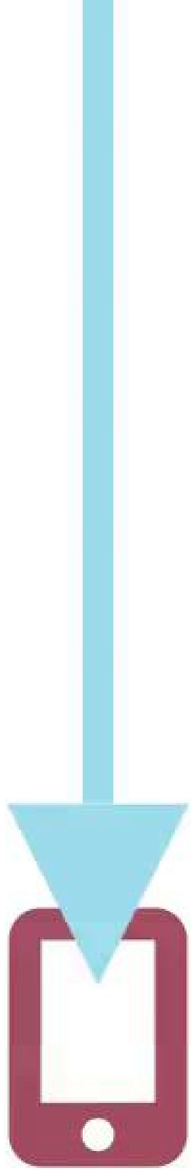
RT and EM Fundamentals



RT and EM Fundamentals



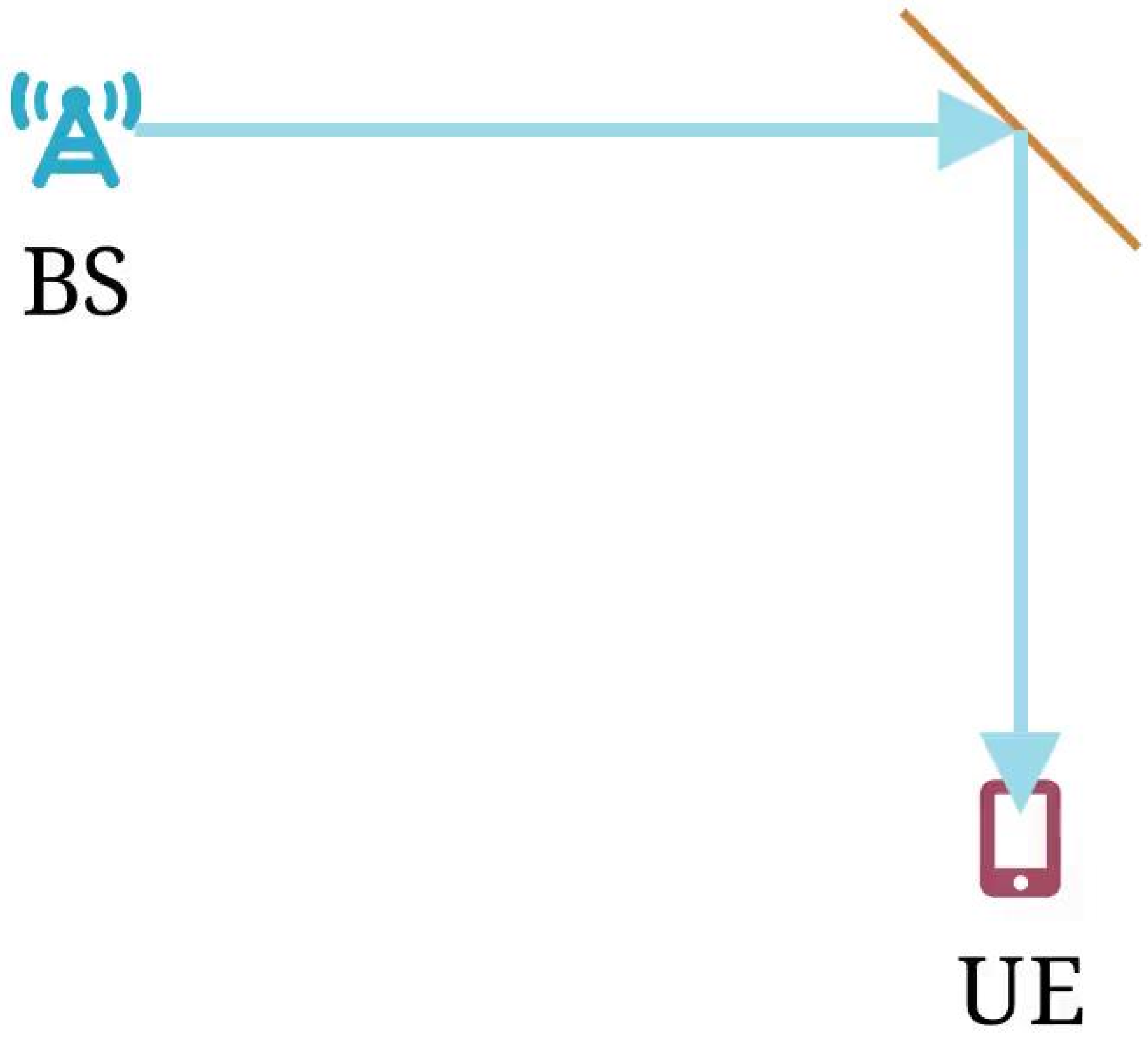

BS



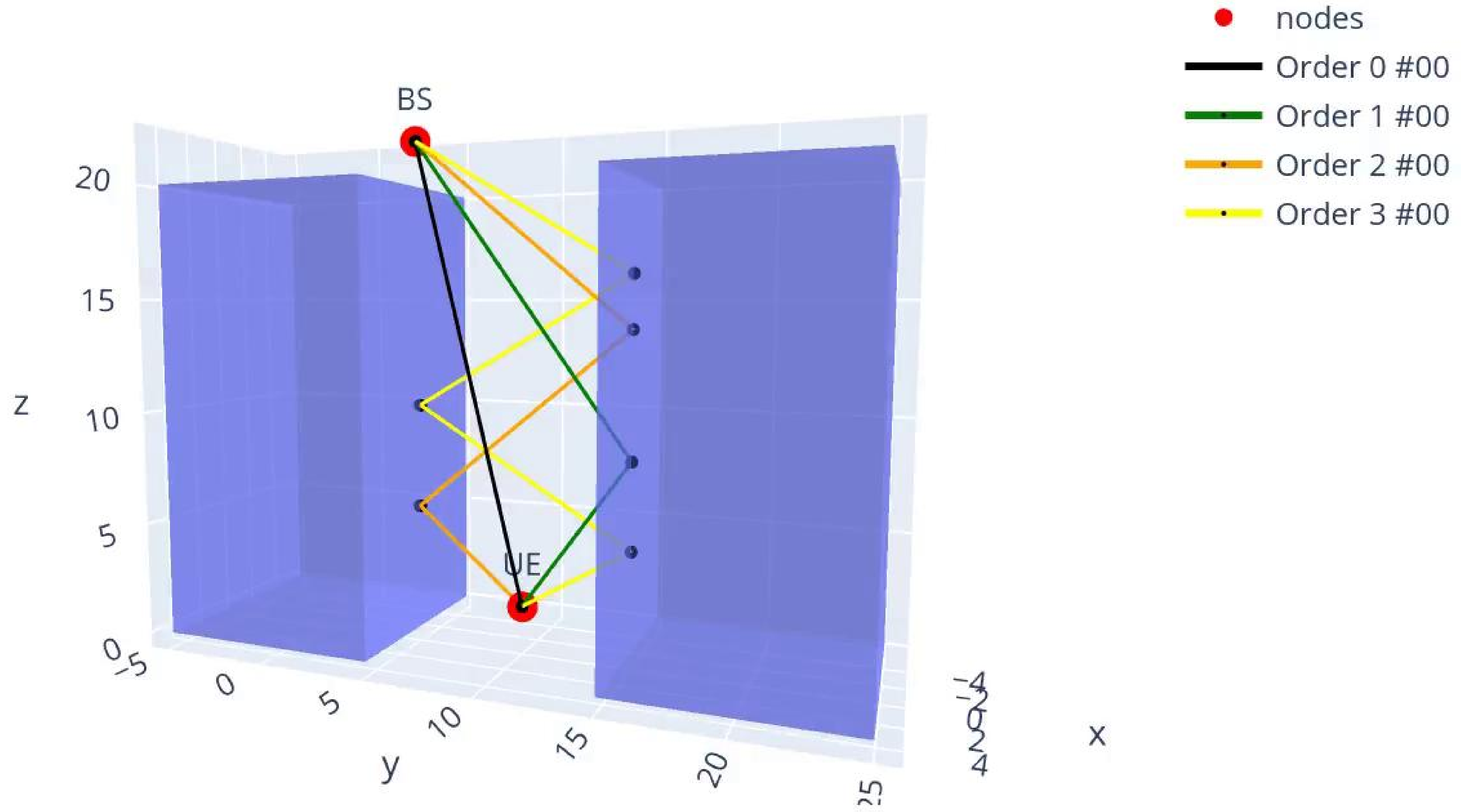
UE

3

RT and EM Fundamentals



RT and EM Fundamentals



RT and EM Fundamentals

Electrical and Magnetic fields

$\vec{E}$ $(\mathrm{V\,m^{-1}})$ & $\vec{B}$ (T)

RT and EM Fundamentals

Electrical and Magnetic fields

$$\vec{E}\ (\mathrm{V\,m^{-1}})\ \&\ \vec{B}\ (\mathrm{T})$$

$$\vec{E}(x, y, z) = \sum_{\mathcal{P} \in \mathcal{S}} \bar{C}(\mathcal{P}) \cdot \vec{E}(\mathcal{P}_1),$$

RT and EM Fundamentals

Electrical and Magnetic fields

$$\vec{E}\ (\mathrm{V\,m^{-1}})\ \&\ \vec{B}\ (\mathrm{T})$$

$$\vec{E}(x,y,z) = \sum_{\mathcal{P}\in\mathcal{S}} \bar{C}(\mathcal{P}) \cdot \vec{E}(\mathcal{P}_1),$$

$$\text{where } \bar{C}(\mathcal{P}) = \prod_{i\in\mathcal{I}} \bar{D}_i \cdot \alpha_i \cdot e^{-j\phi_i}.$$

RT and EM Fundamentals

Input scene

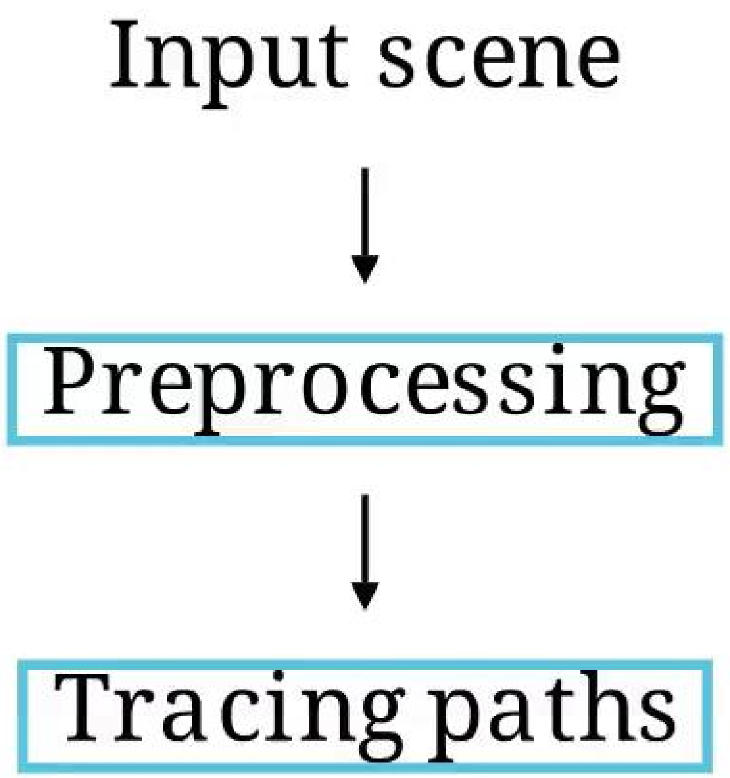
RT and EM Fundamentals

Input scene

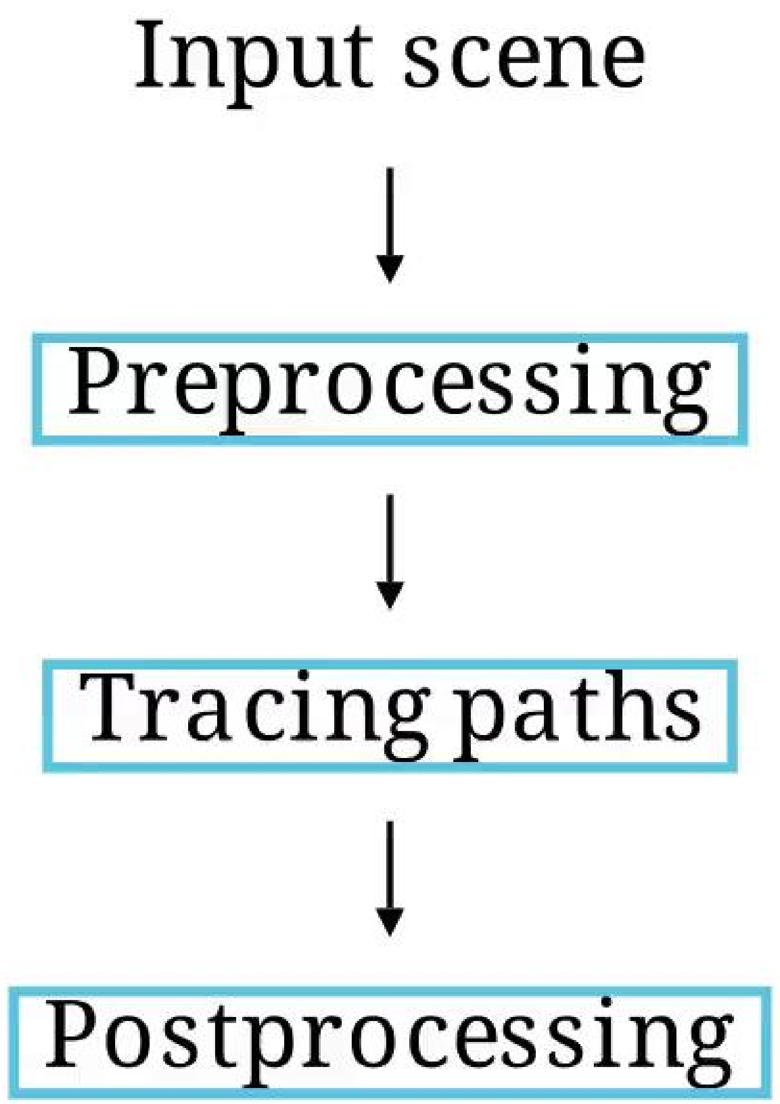
↓

Preprocessing

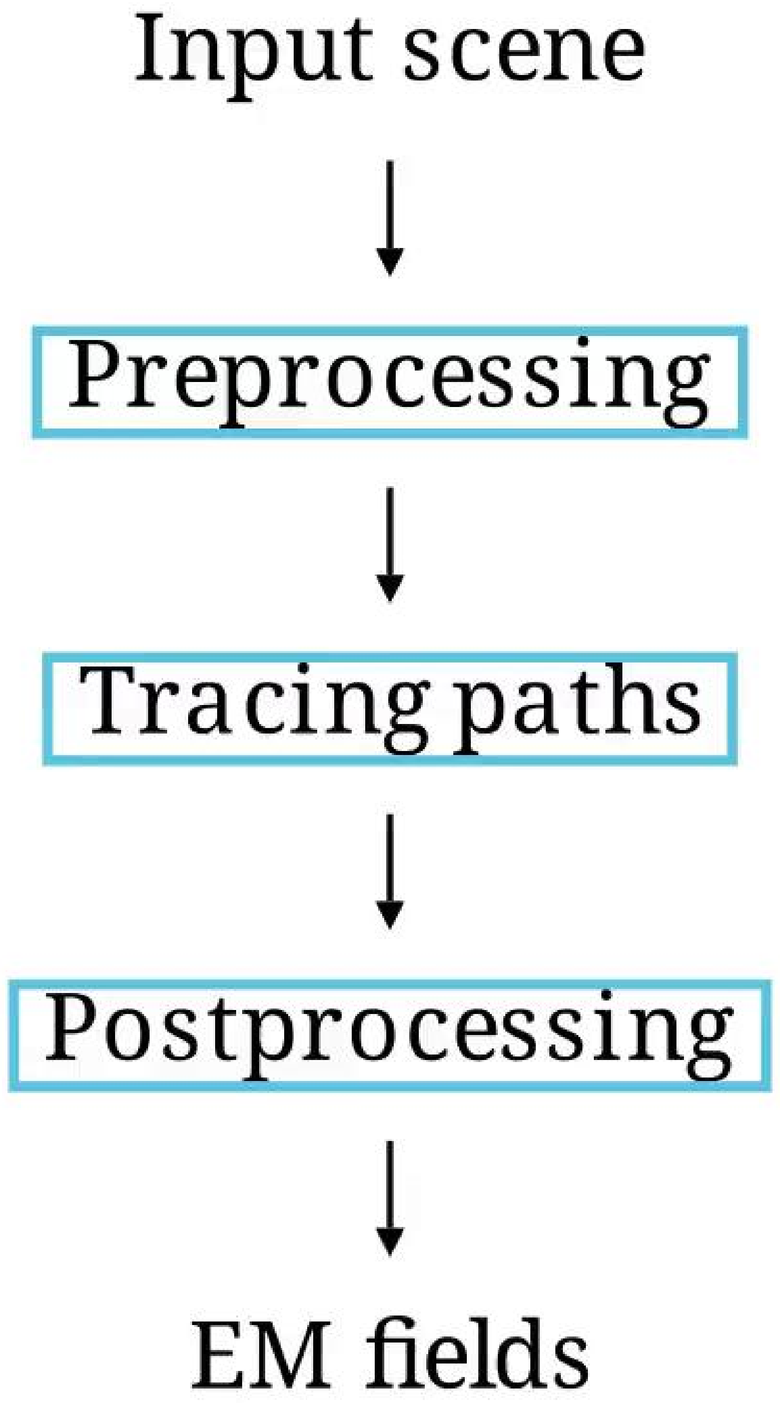
RT and EM Fundamentals



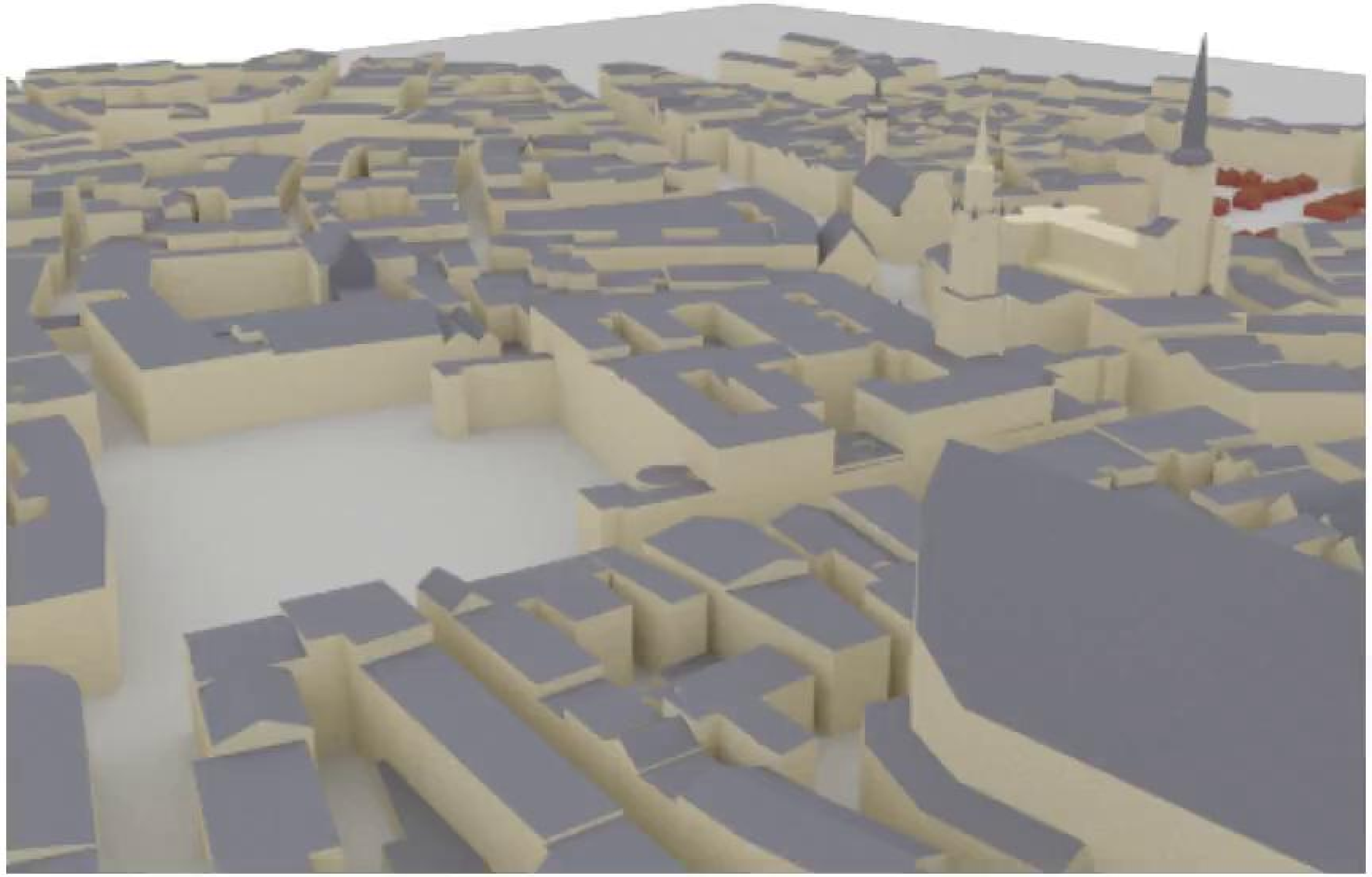
RT and EM Fundamentals



RT and EM Fundamentals

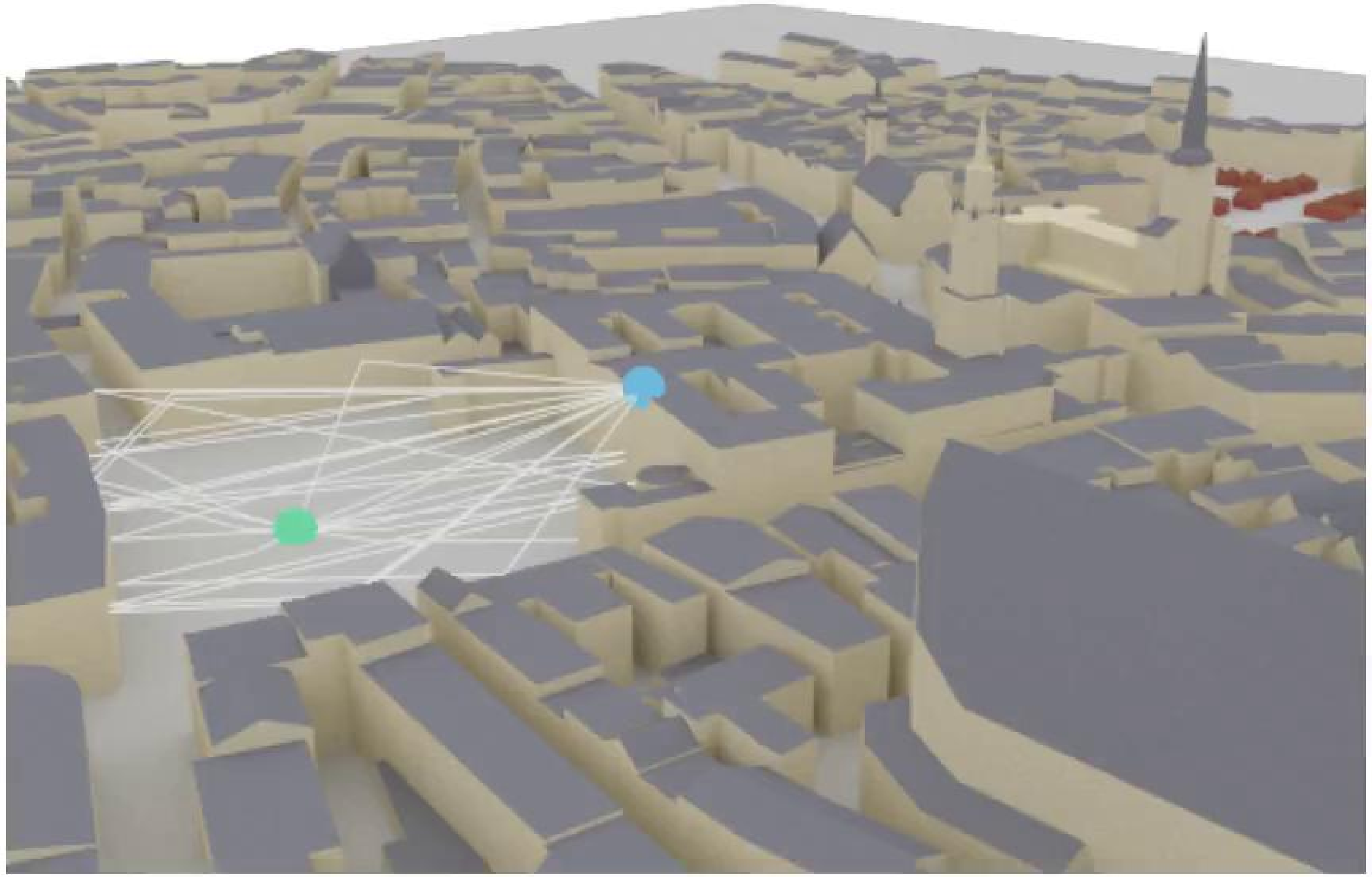


RT and EM Fundamentals



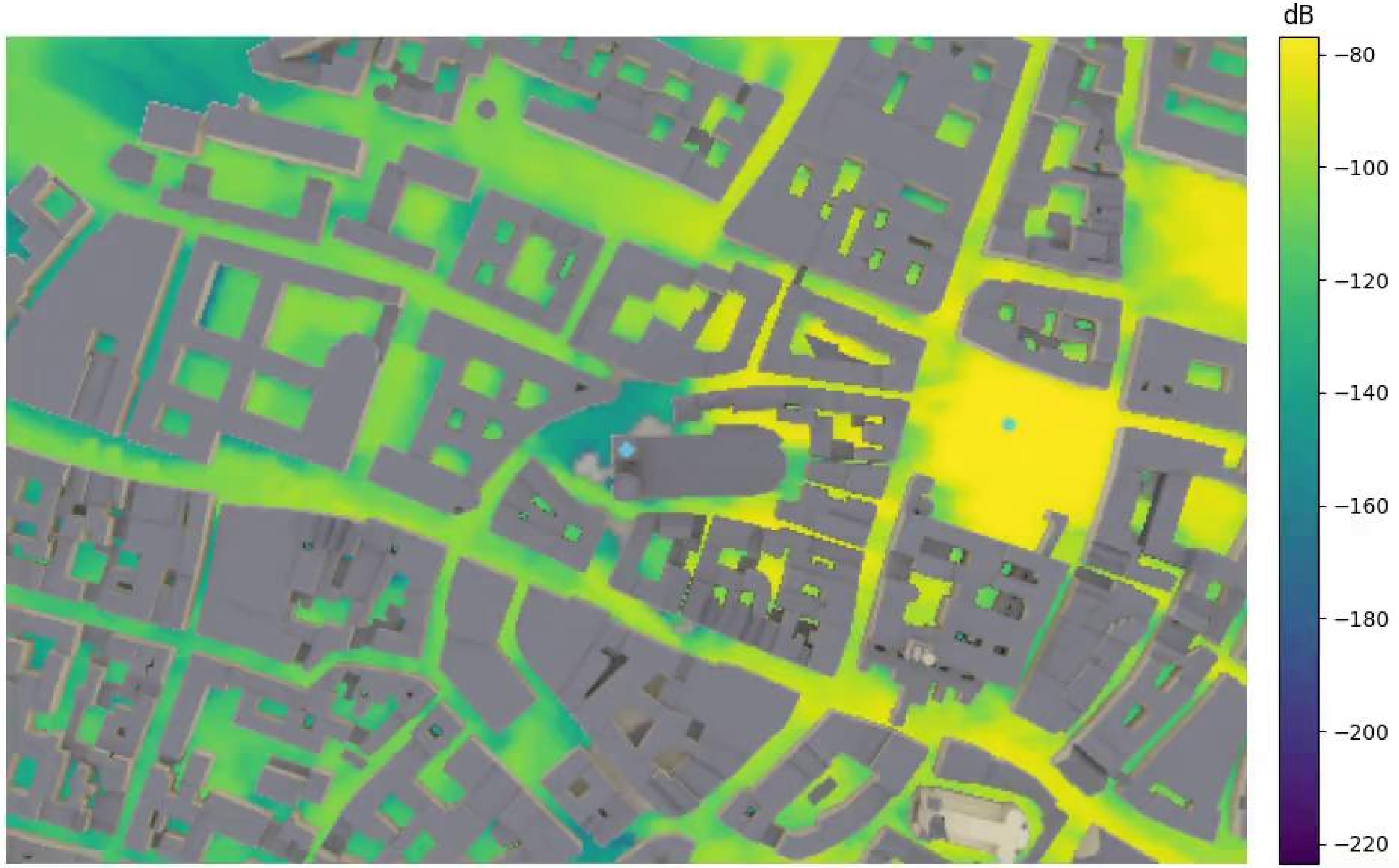
Credits: Sionna authors, Nvidia.

RT and EM Fundamentals



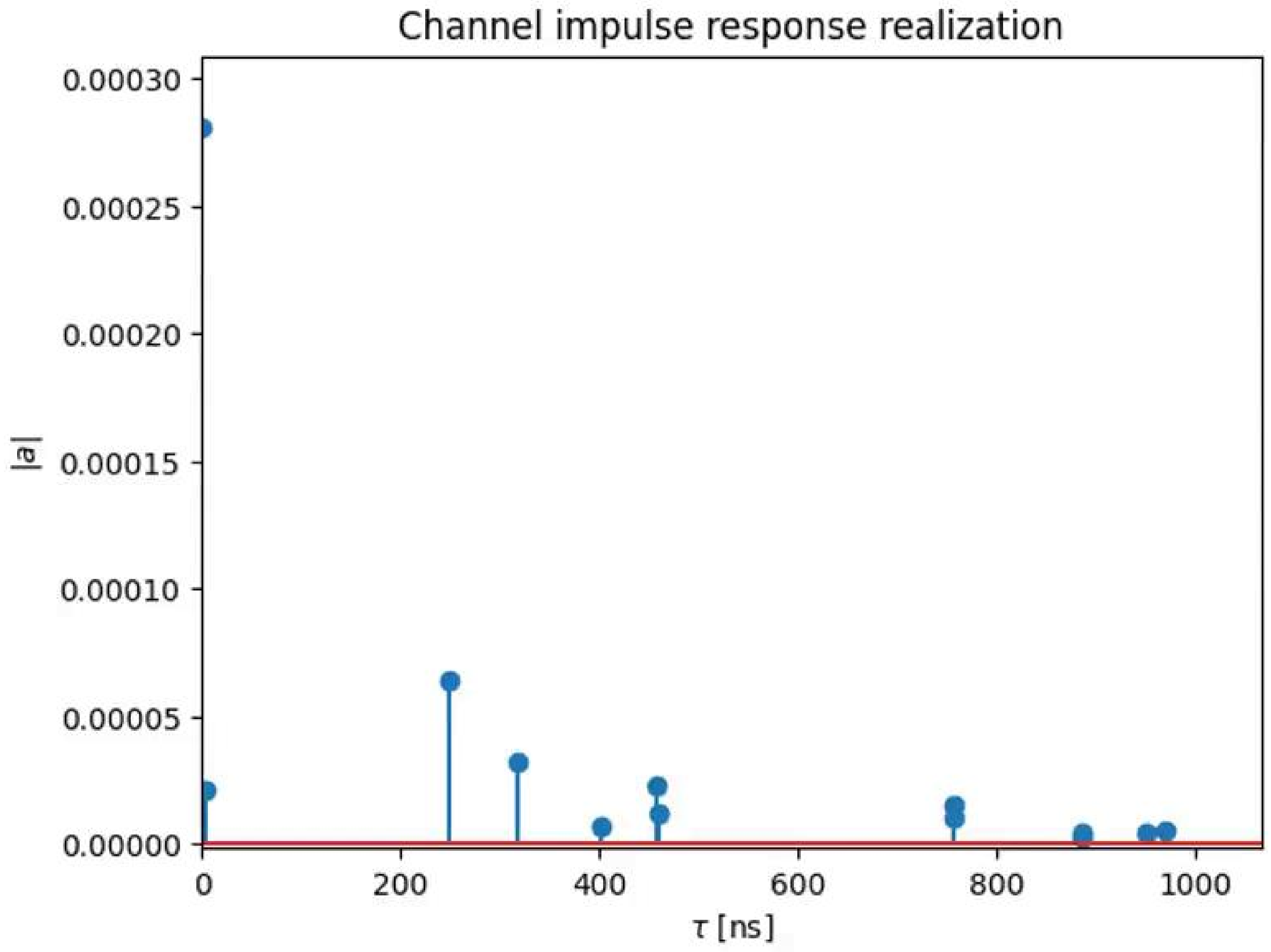
Credits: Sionna authors, Nvidia.

RT and EM Fundamentals



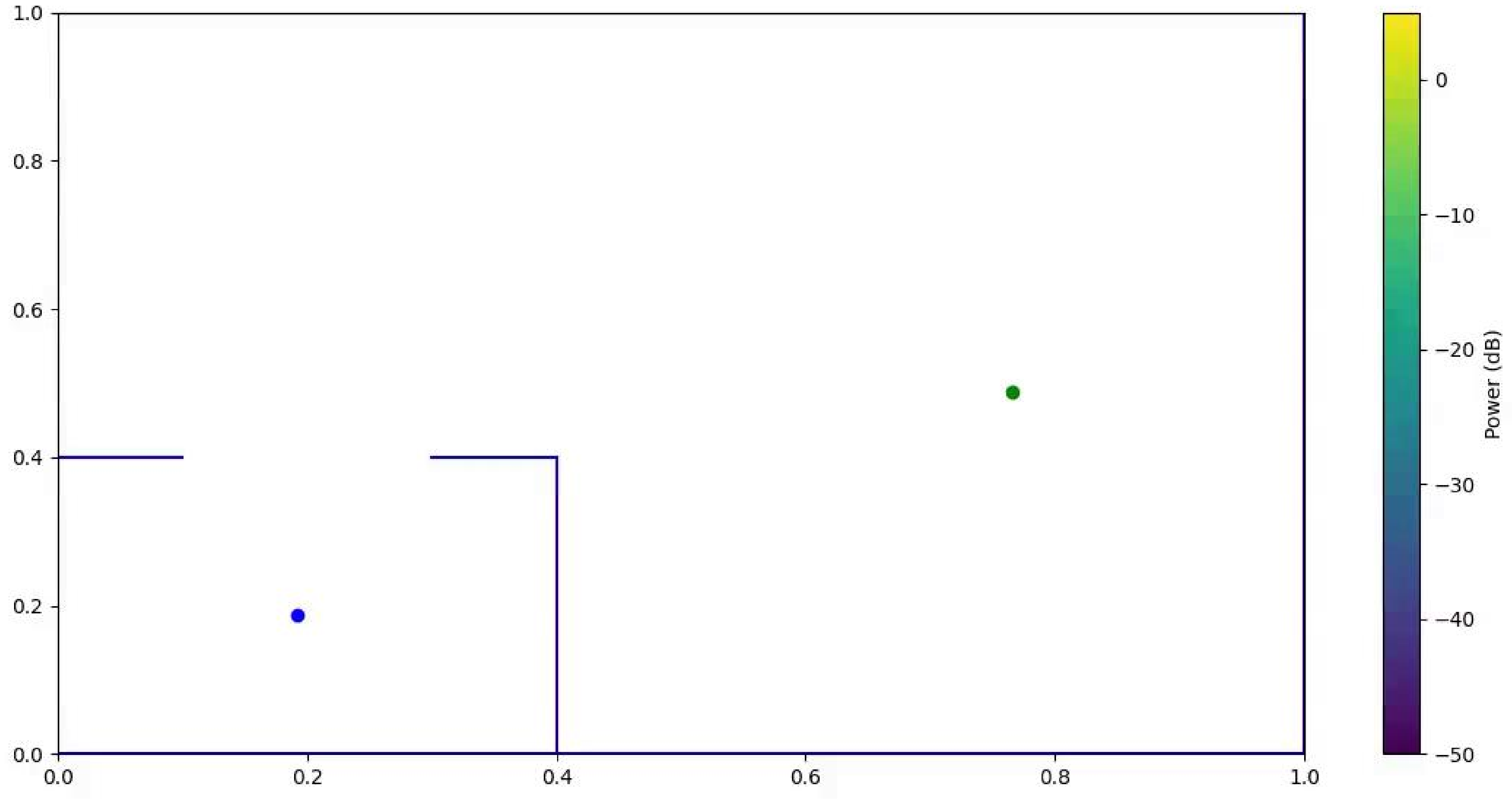
Credits: Sionna authors, Nvidia.

RT and EM Fundamentals



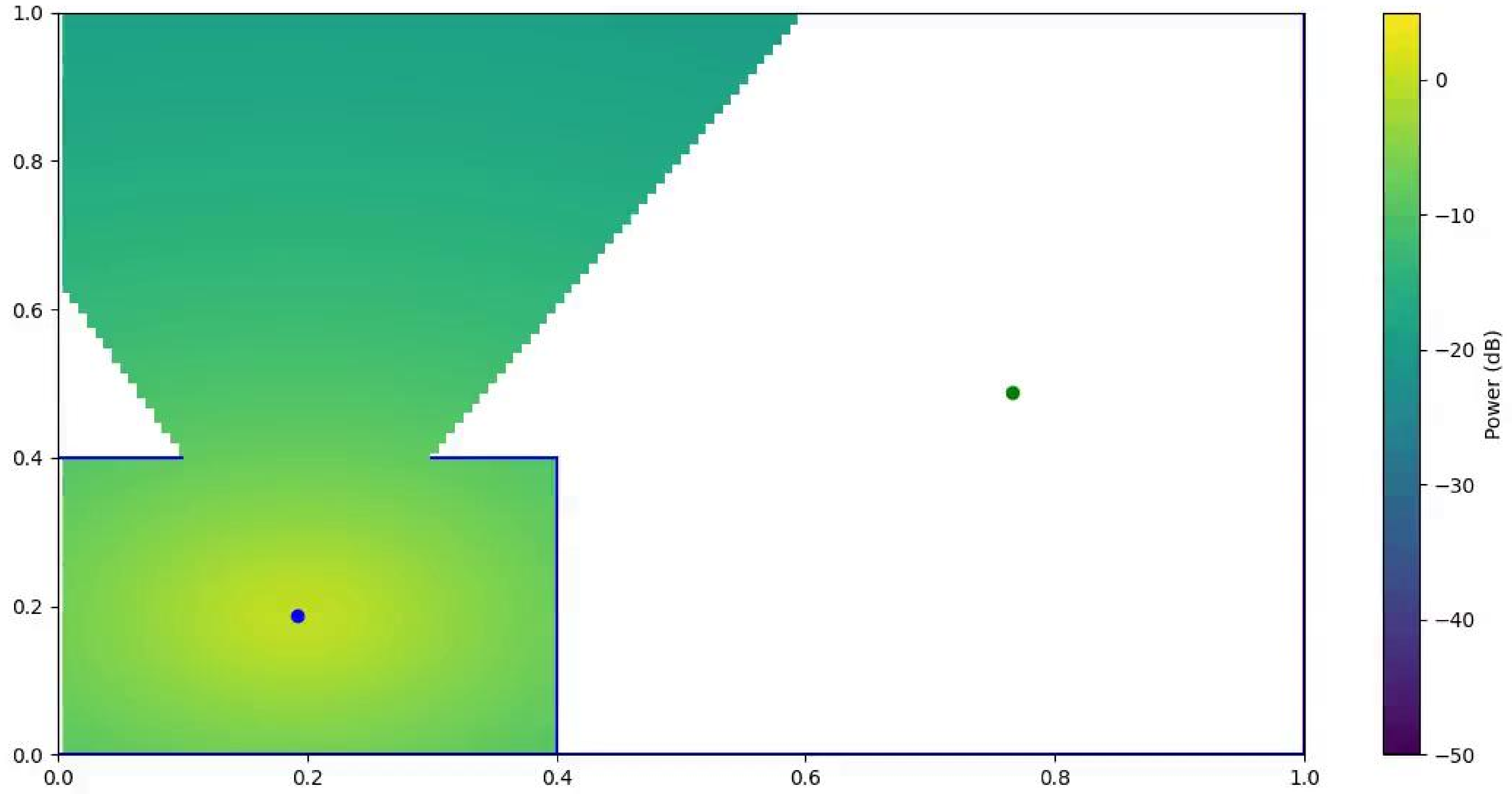
Credits: Sionna authors, Nvidia.

RT and EM Fundamentals



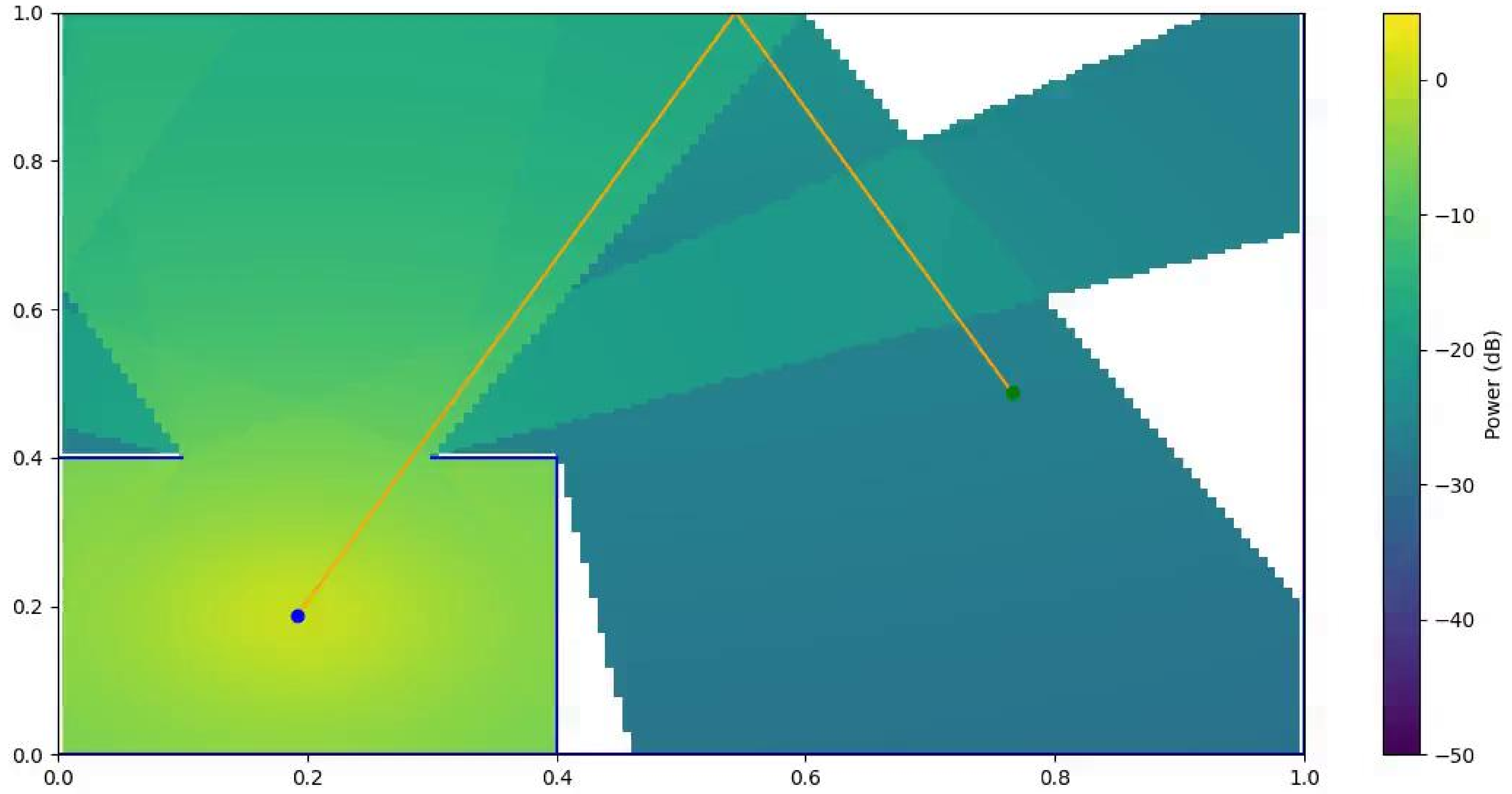
Challenge: number of paths.

RT and EM Fundamentals



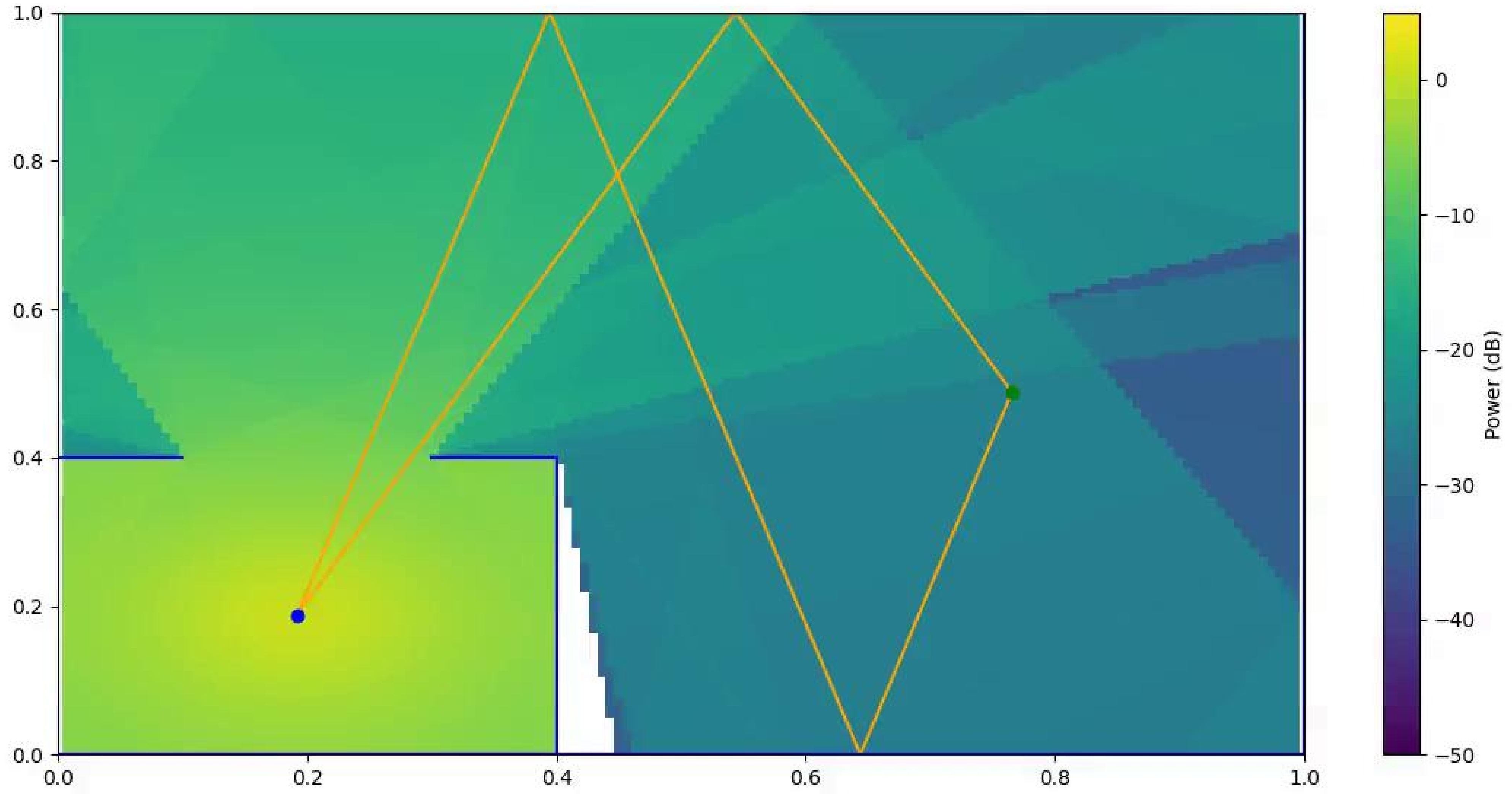
Challenge: number of paths.

RT and EM Fundamentals



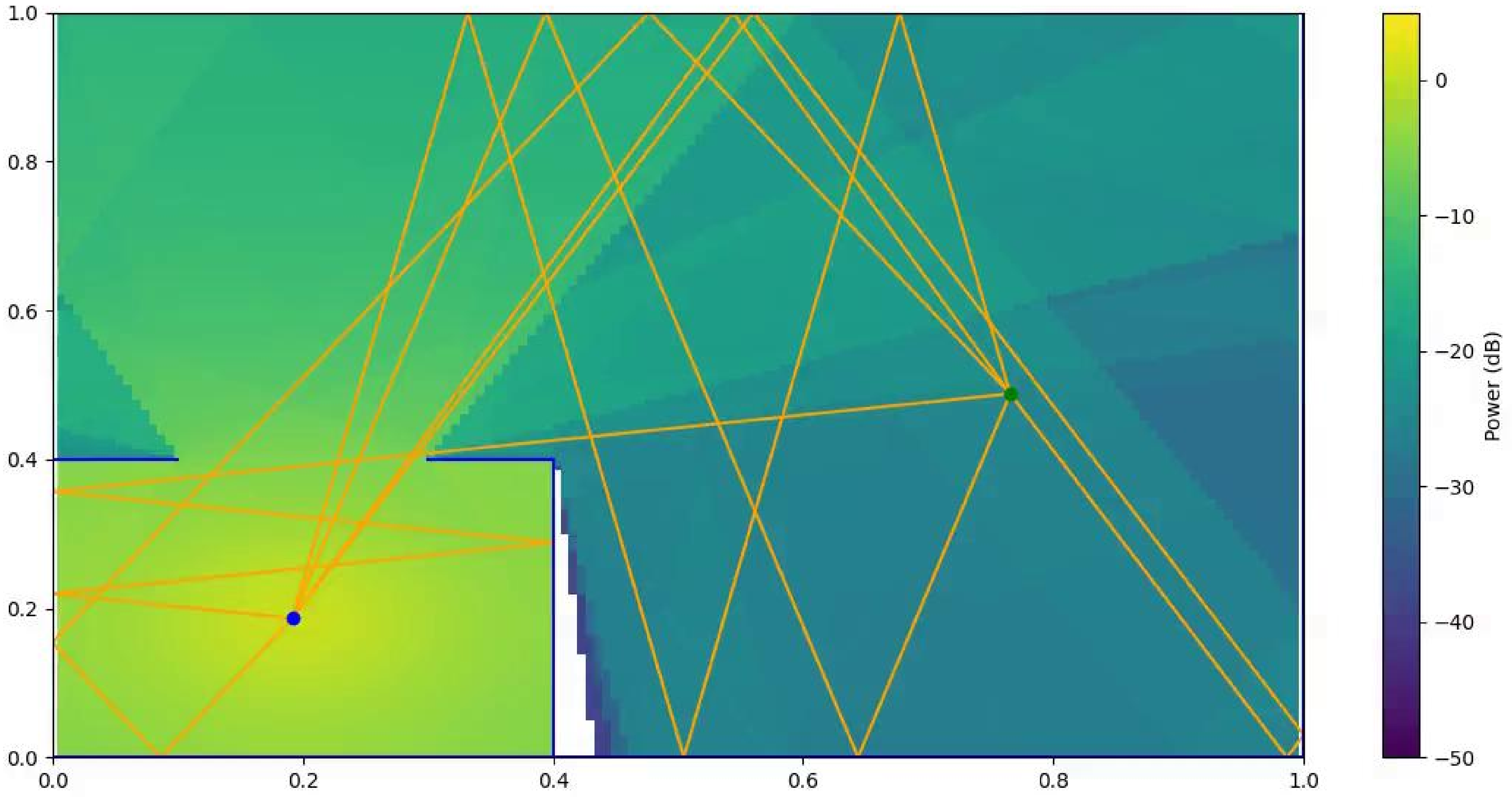
Challenge: number of paths.

RT and EM Fundamentals



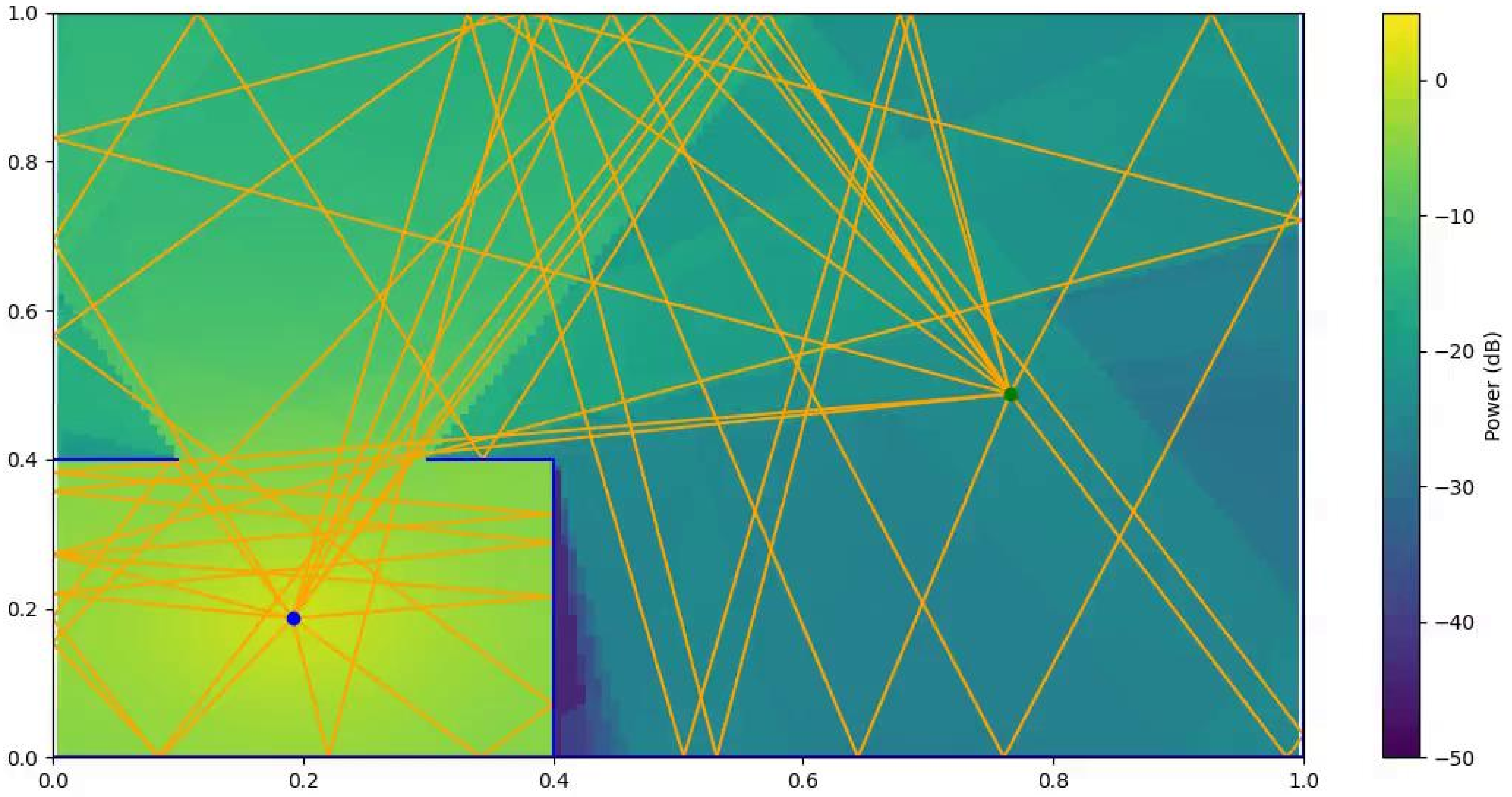
Challenge: number of paths.

RT and EM Fundamentals



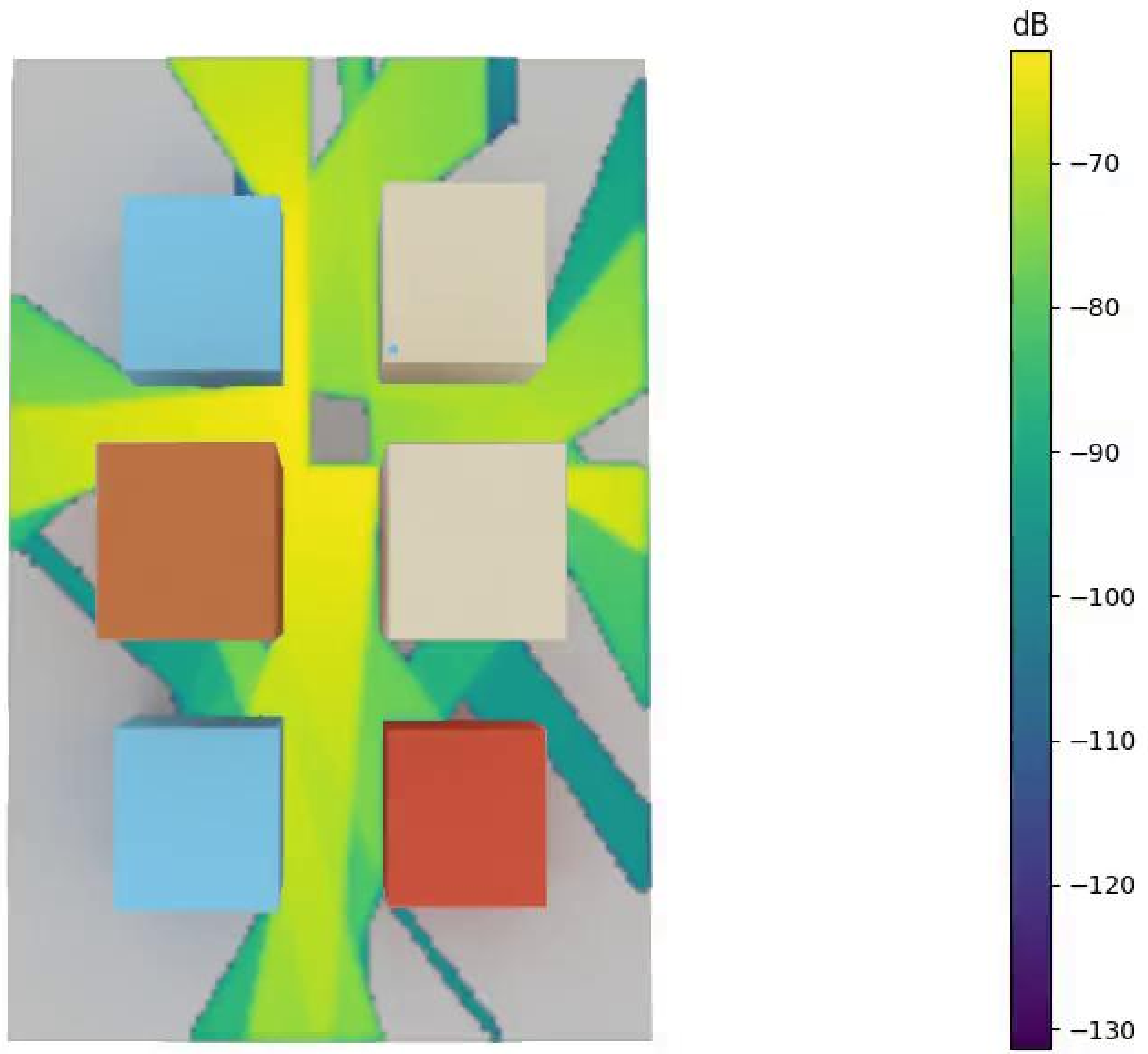
Challenge: number of paths.

RT and EM Fundamentals



Challenge: number of paths.

RT and EM Fundamentals

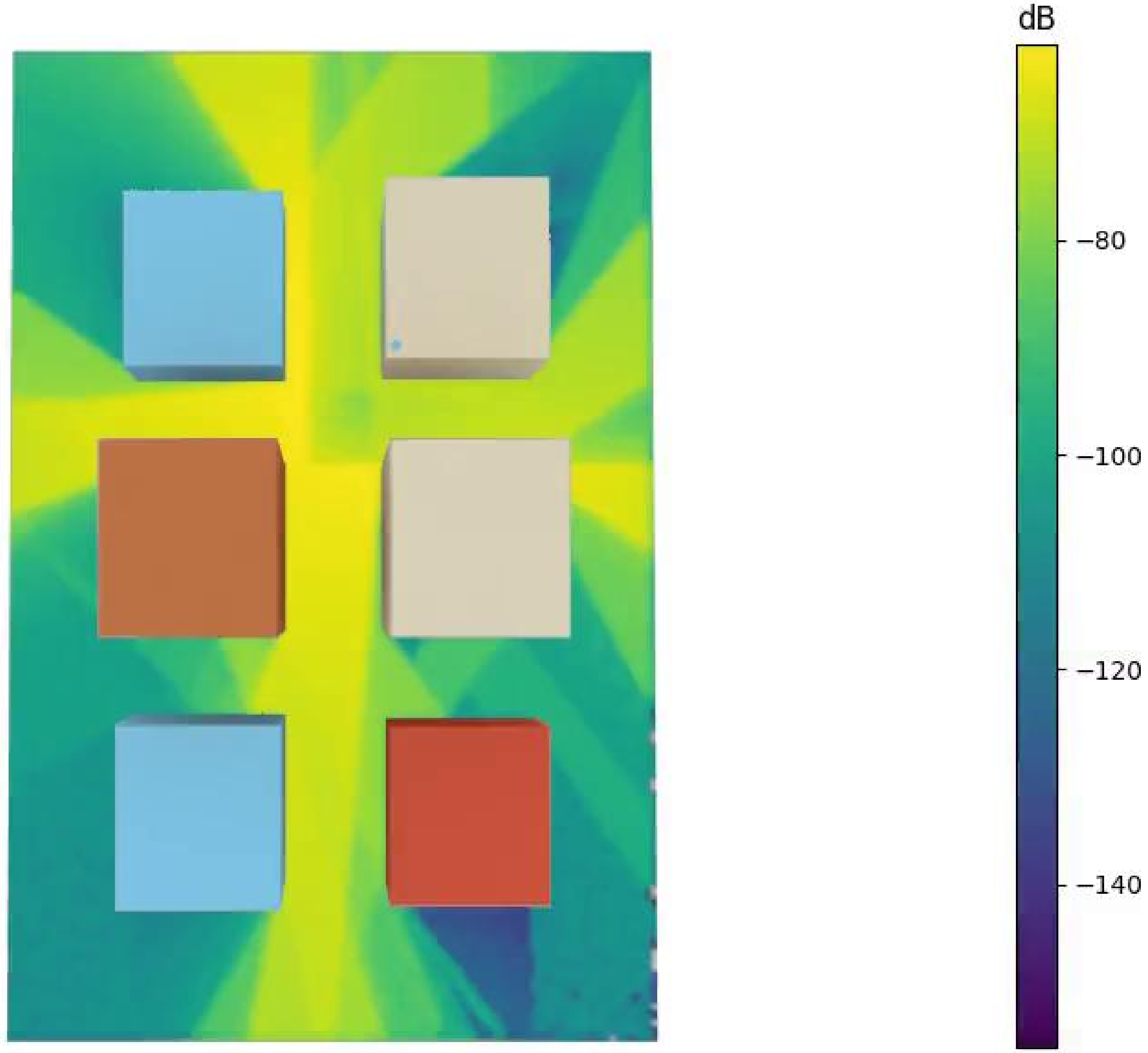


LOS + reflection

Challenge: coverage vs order and types.

Credits: Sionna authors, Nvidia.

RT and EM Fundamentals

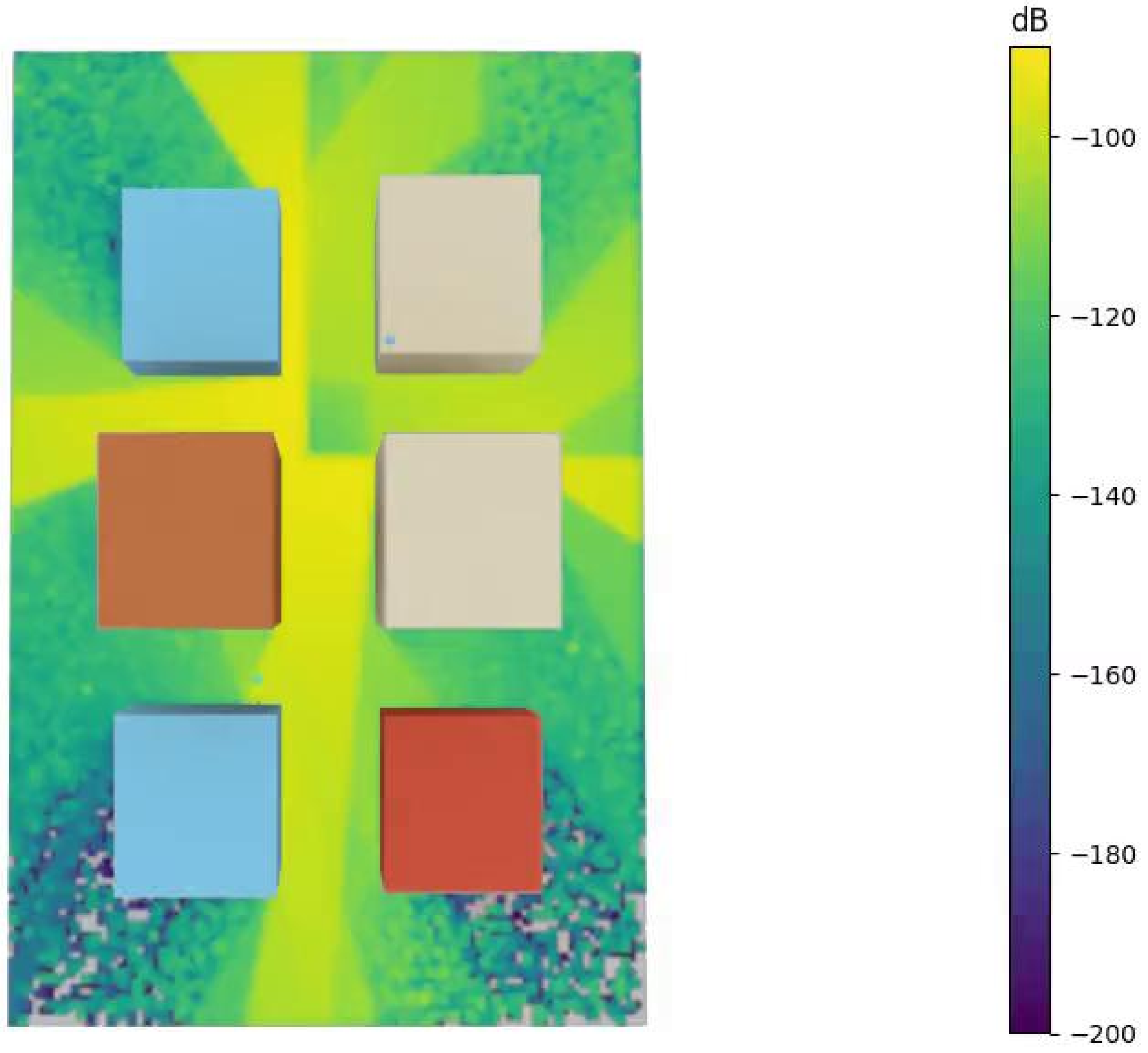


LOS + reflection + diffraction

Challenge: coverage vs order and types.

Credits: Sionna authors, Nvidia.

RT and EM Fundamentals



LOS + reflection + scattering

Challenge: coverage vs order and types.

Credits: Sionna authors, Nvidia.

RT and EM Fundamentals

Main RT applications:

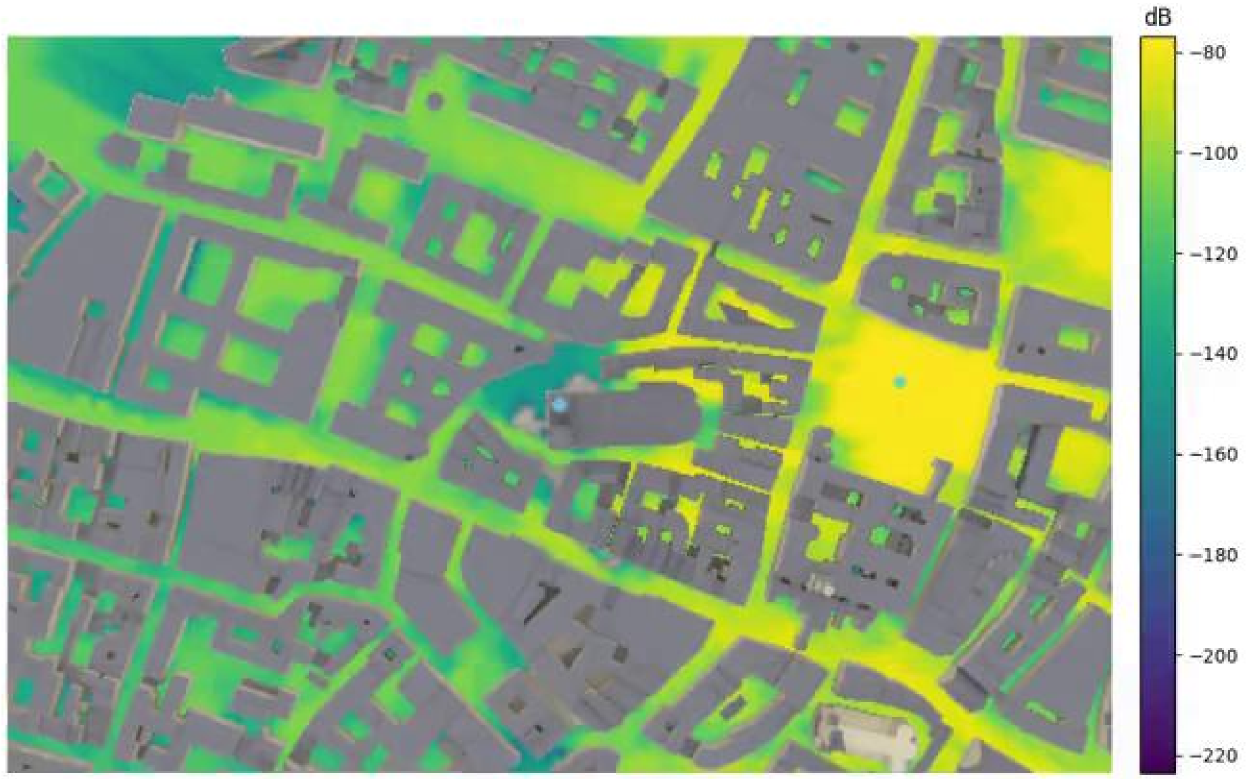
- radio channel modeling;
- sound and light prop. in video games;
- inverse rendering in graphics;
- lenses design and manufacturing.

RT and EM Fundamentals

Most used channel modeling methods:

- RT;
- empirical models;
- stochastic models;
- full-wave models (e.g., finite elements).

Motivations



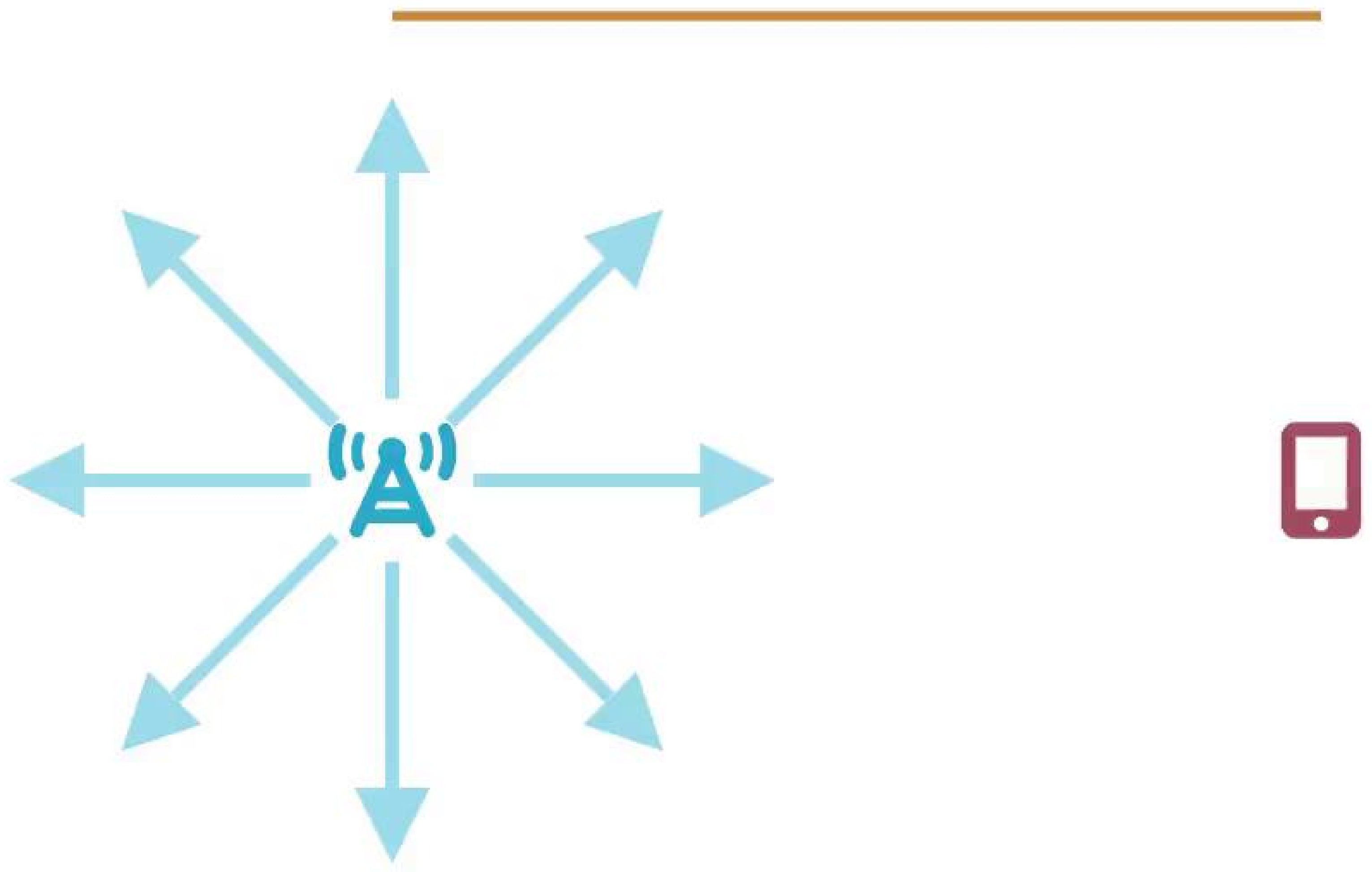
Why Differentiable Ray Tracing?

- RT is inherently static;
- but scenarios are becoming dynamic;
- recomputing the "whole map" is bad;
- Differentiability should be a goal!

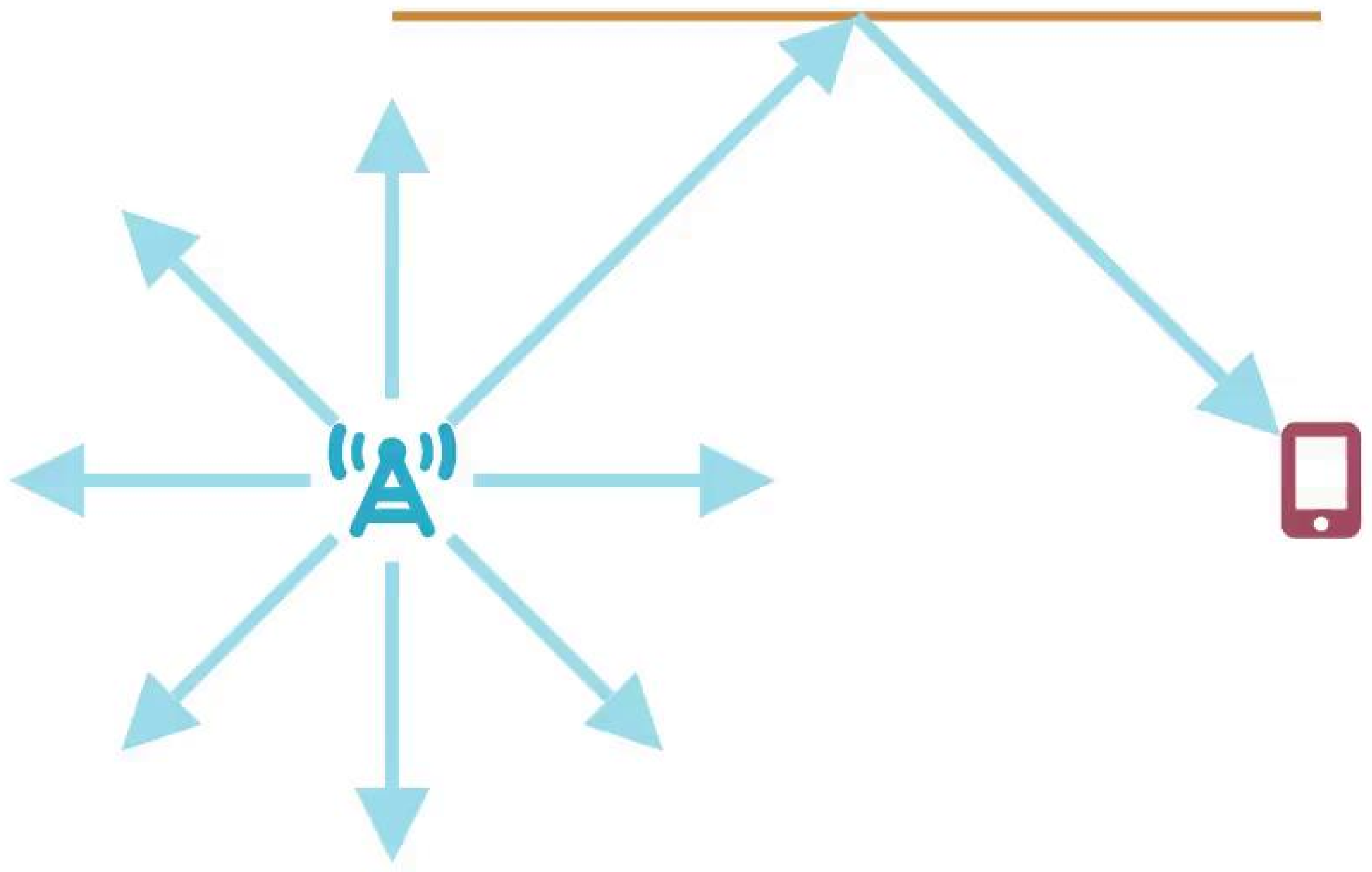
How to trace paths

- Ray Launching vs Ray Tracing;
- Image Method and similar;
- Min-Path-Tracing;
- Arbitrary geometries.

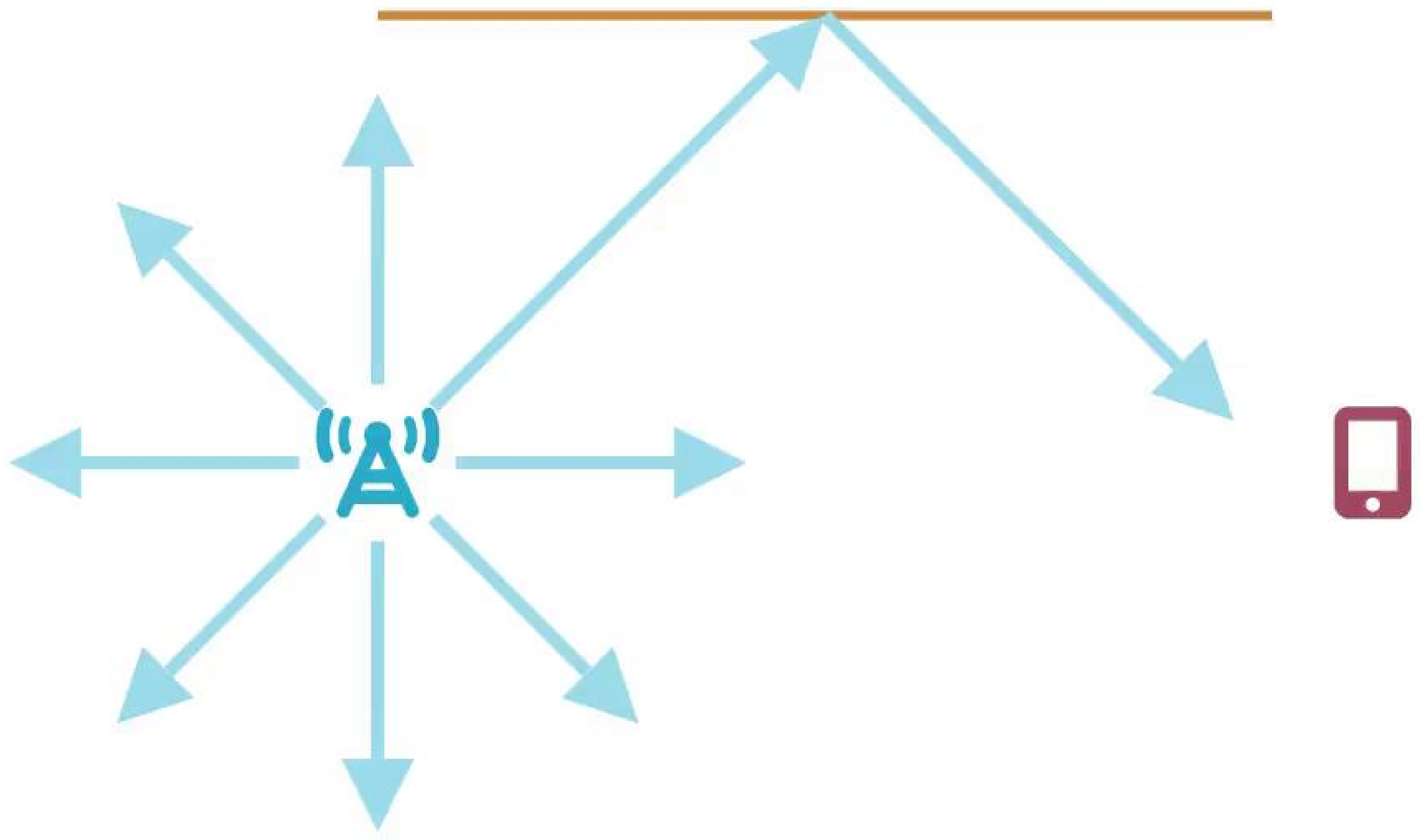
How to trace paths



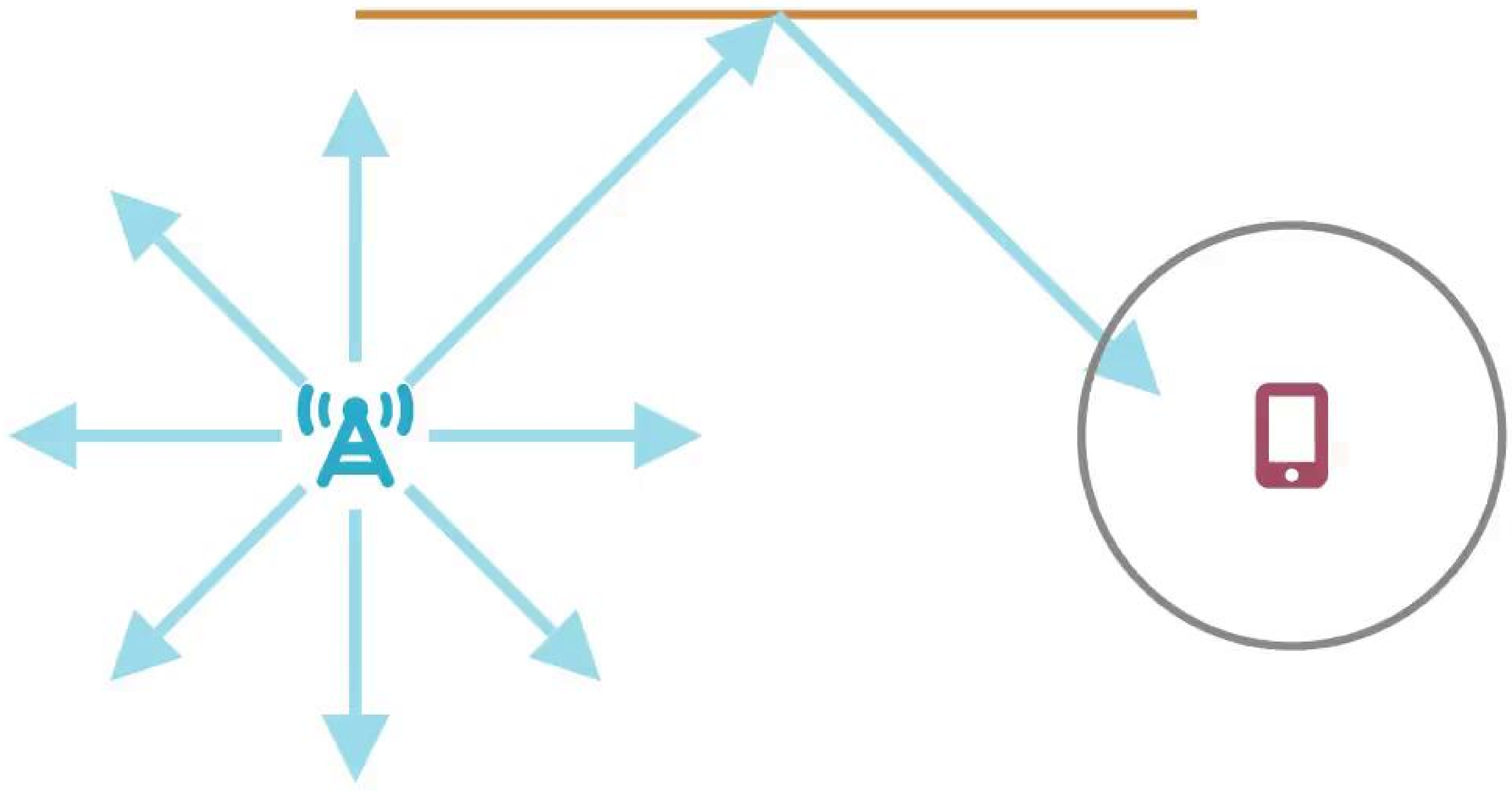
How to trace paths



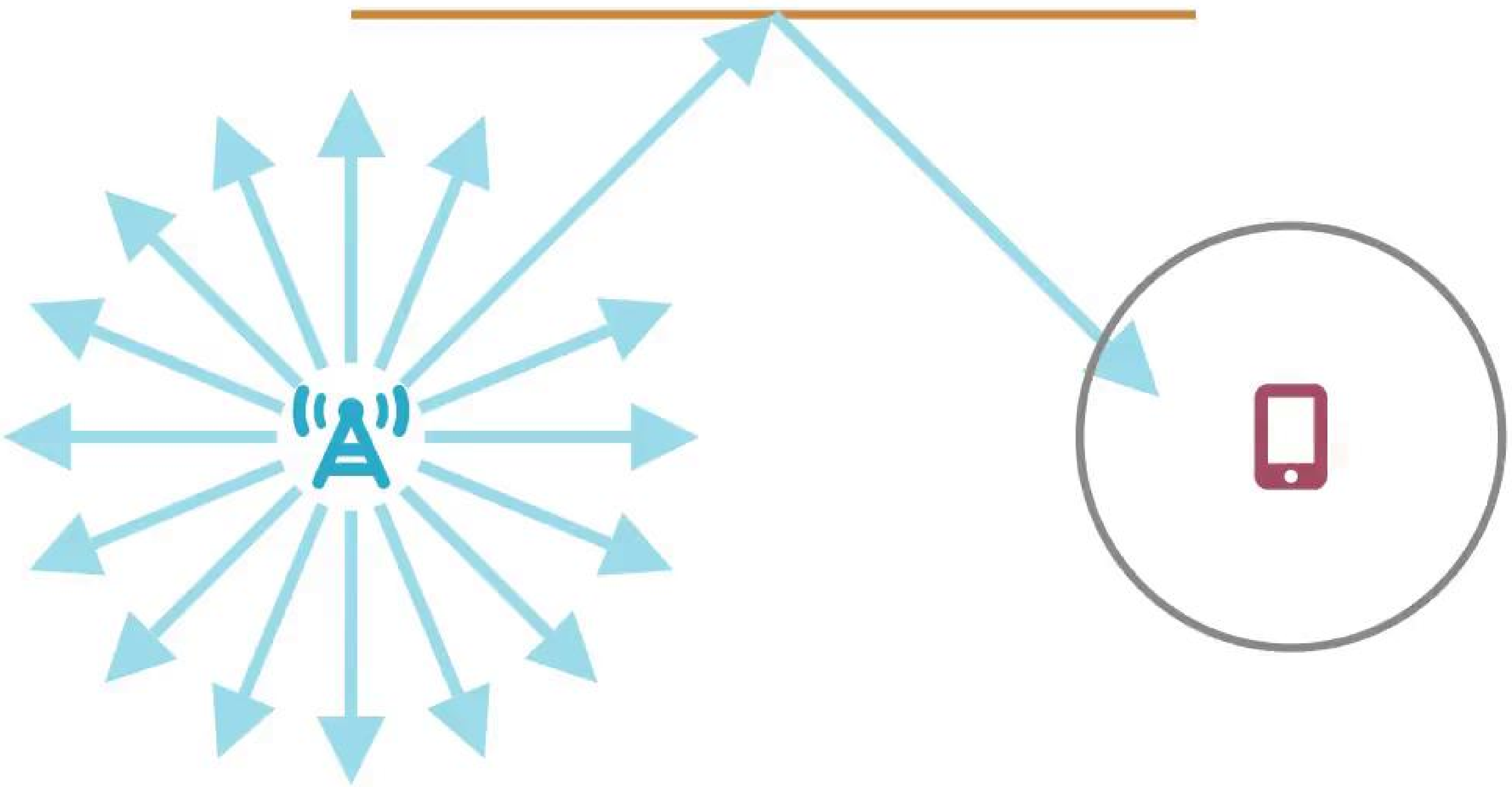
How to trace paths



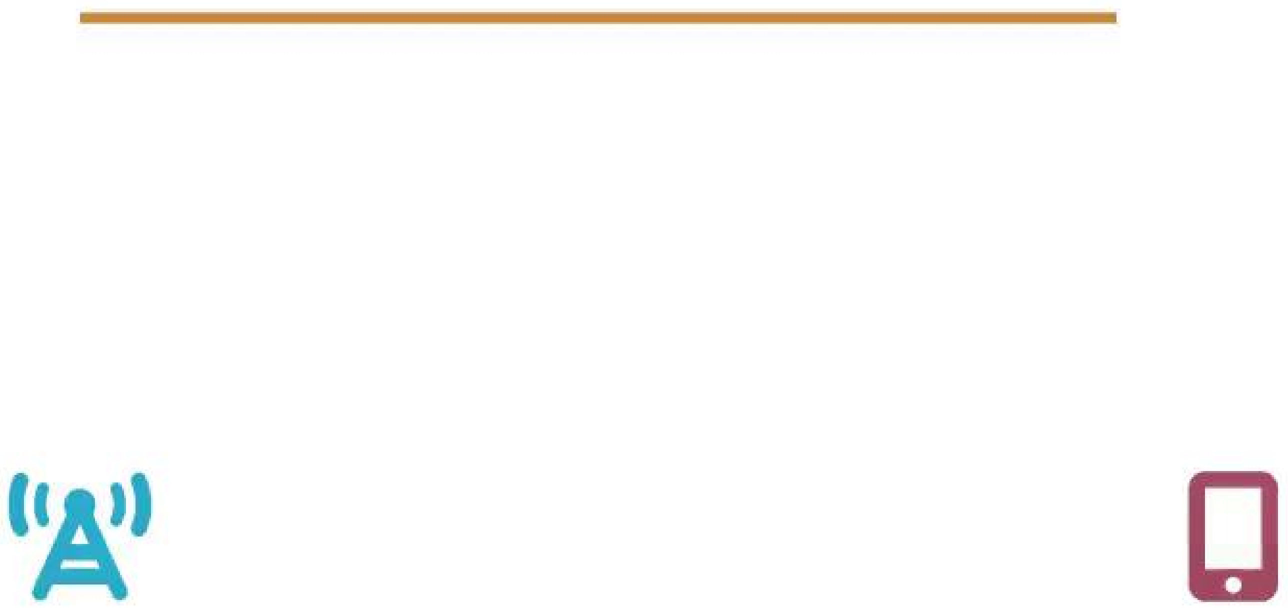
How to trace paths



How to trace paths

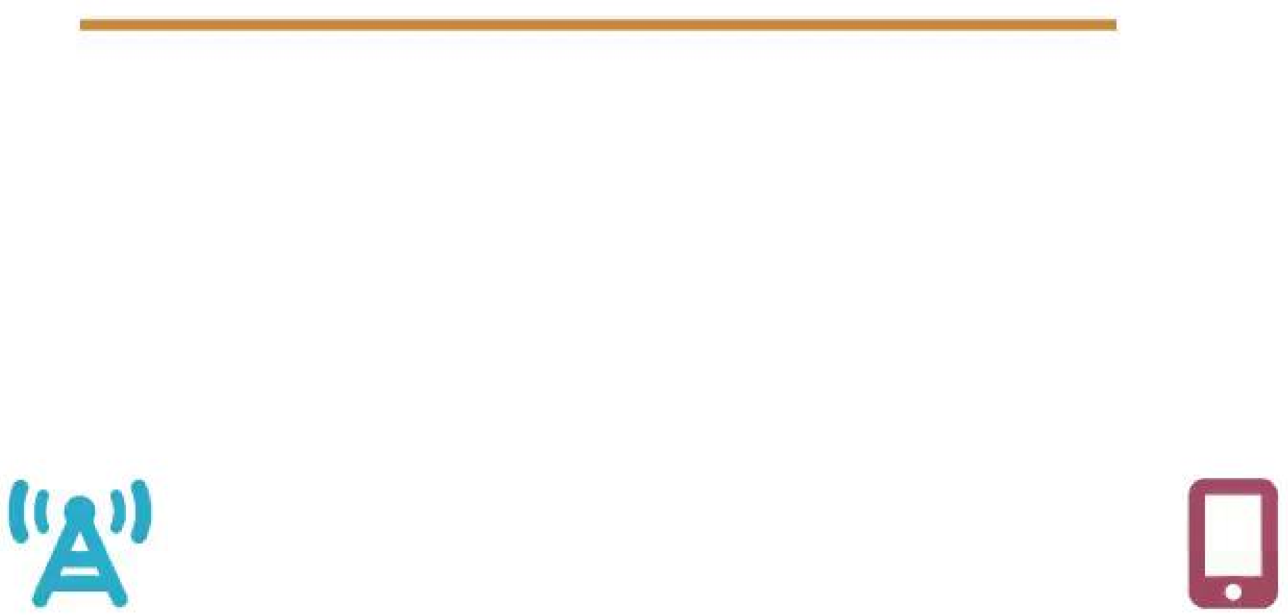


How to trace paths



Not very efficient for "point-to-point" RT

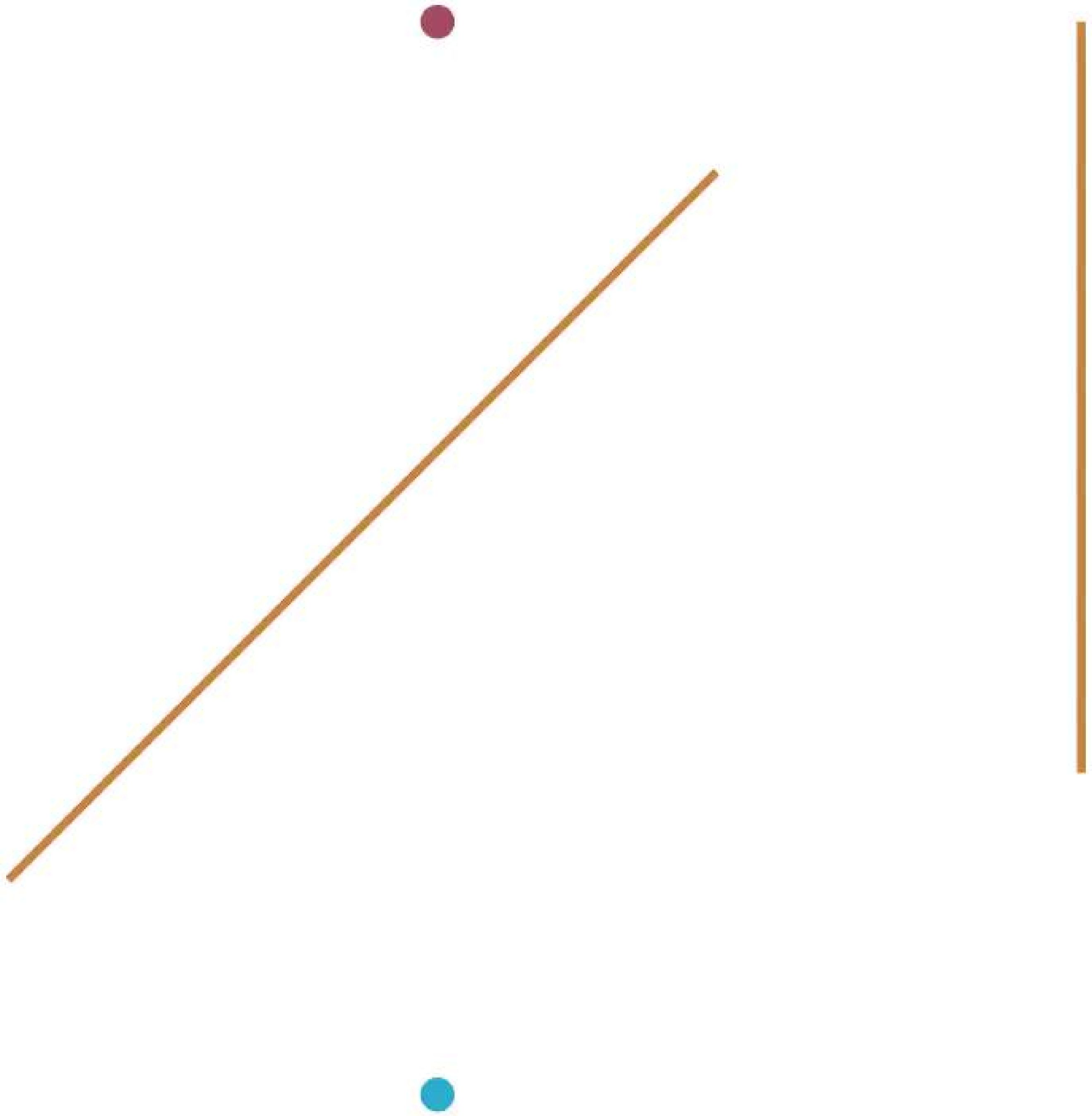
How to trace paths



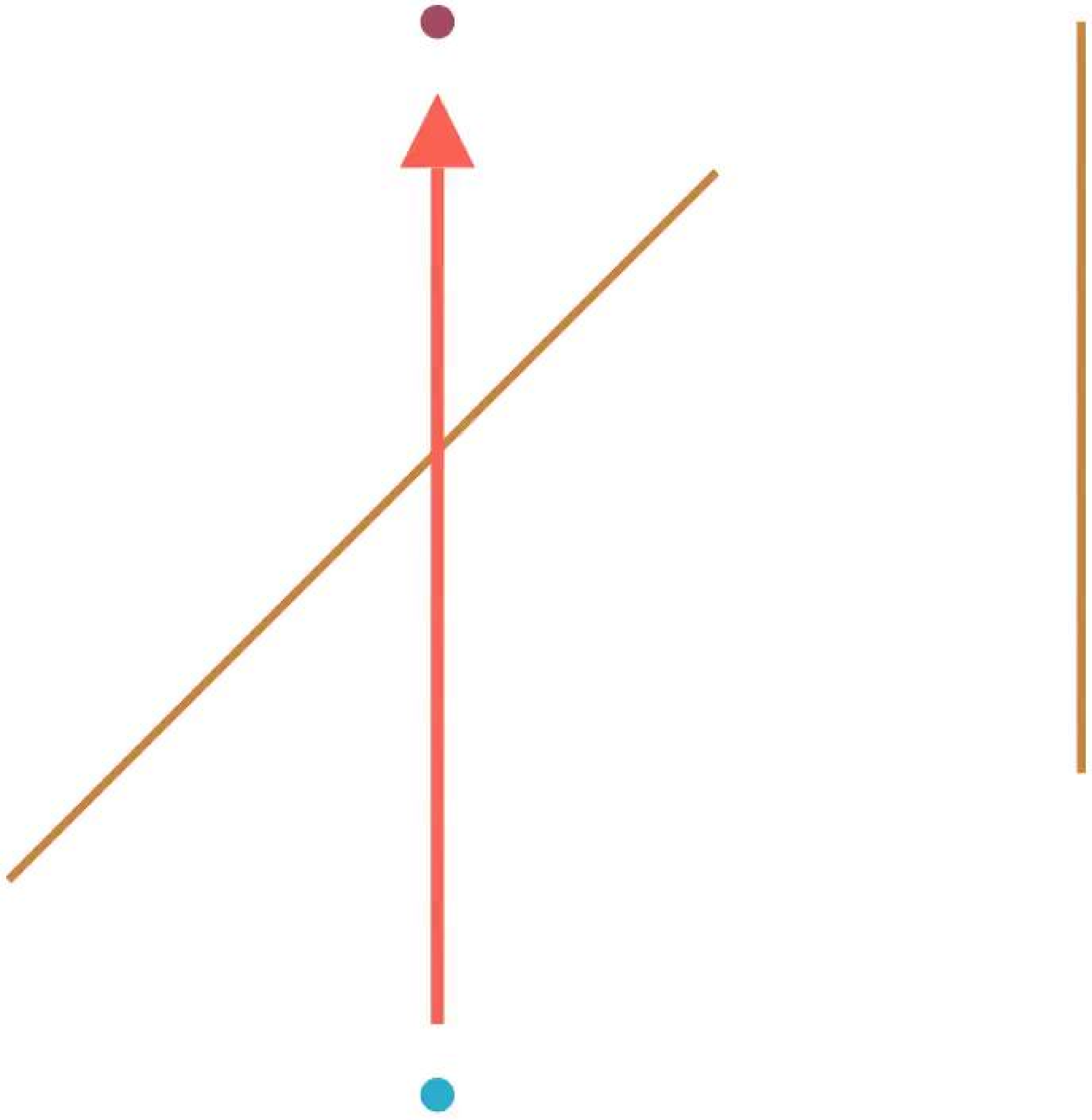
Not very efficient for "point-to-point" RT

How to exactly find paths?

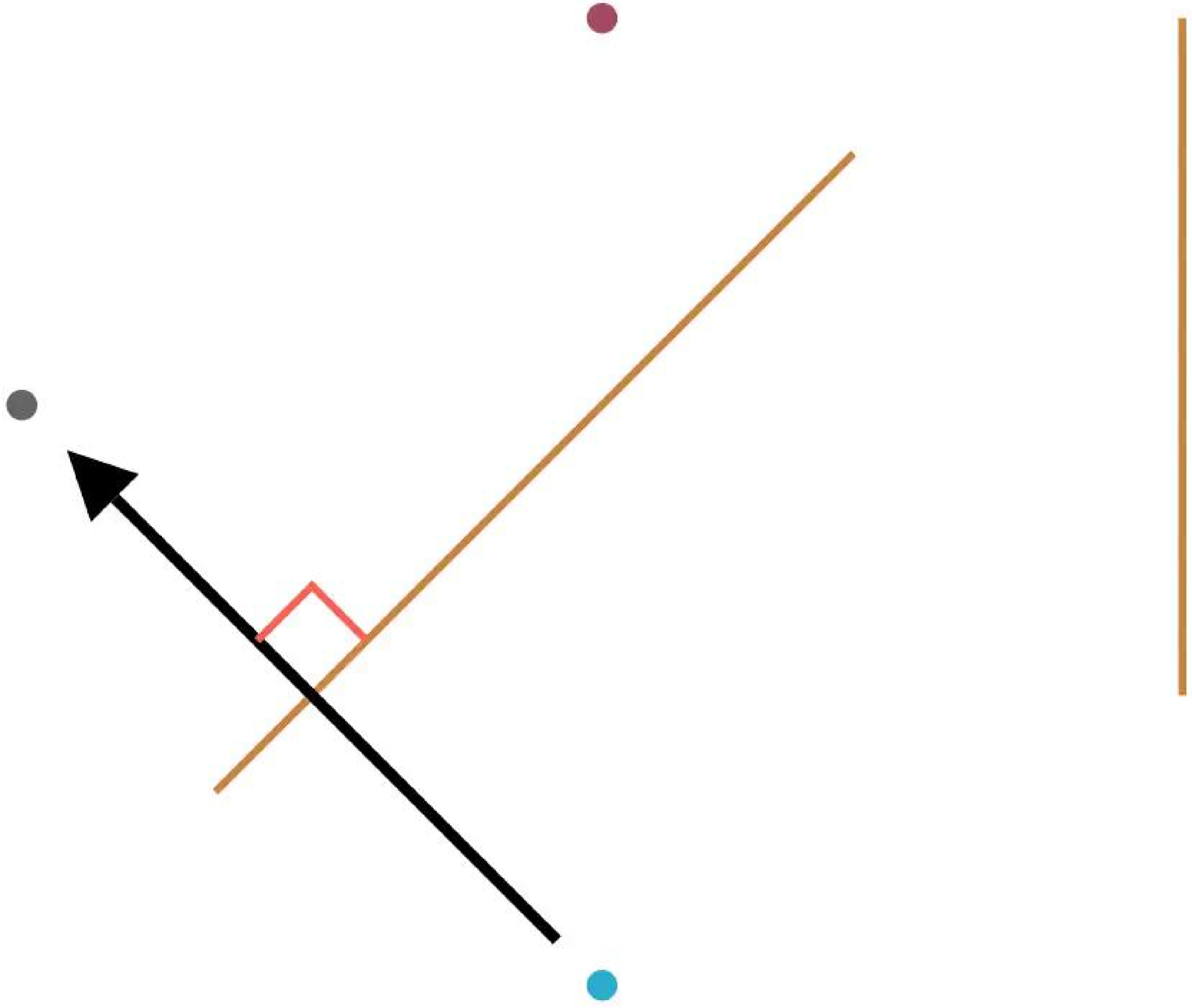
How to trace paths



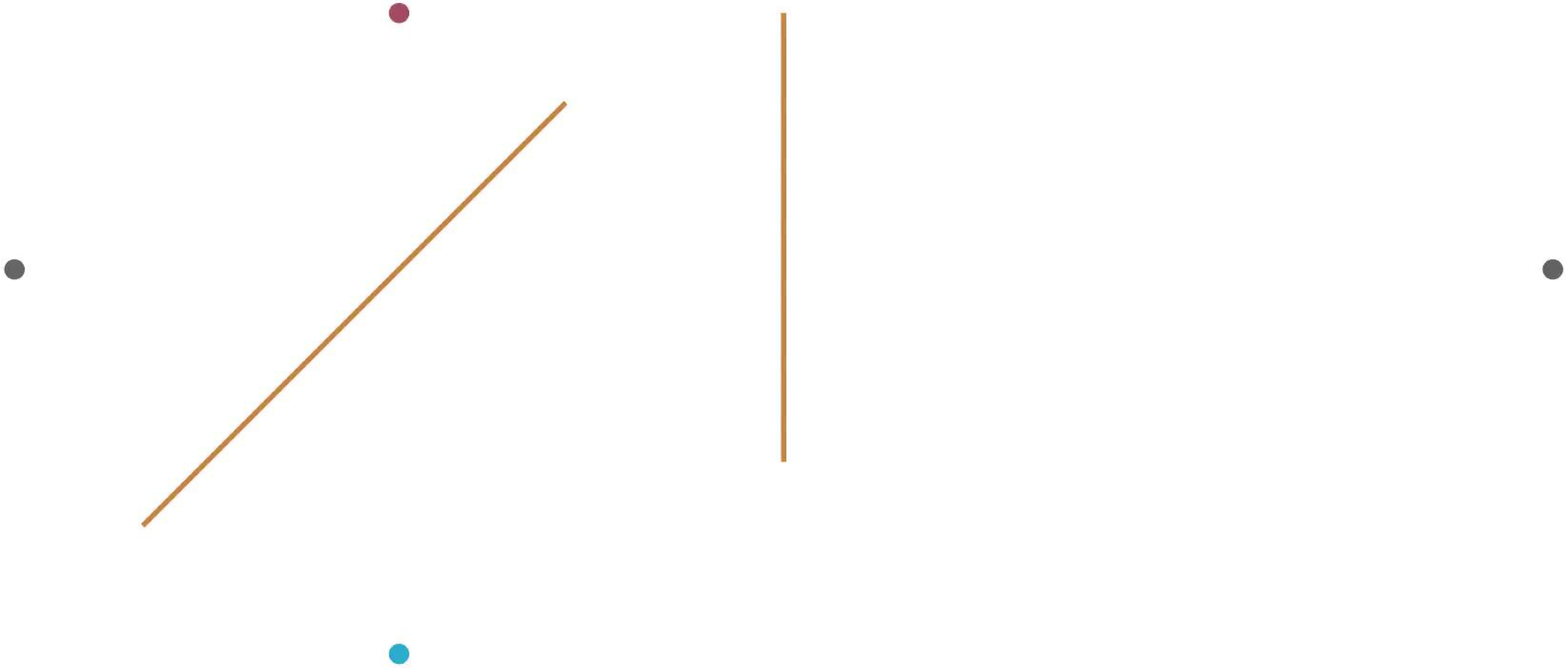
How to trace paths



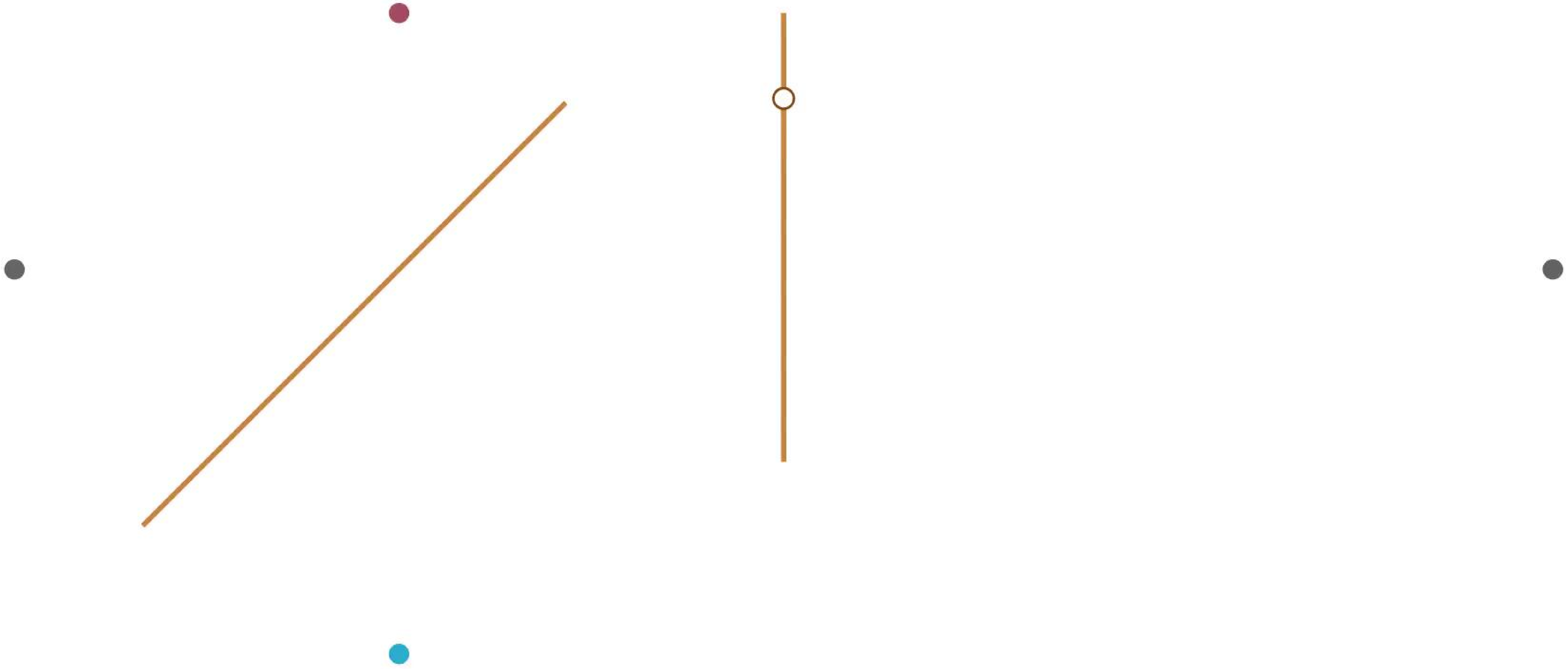
How to trace paths



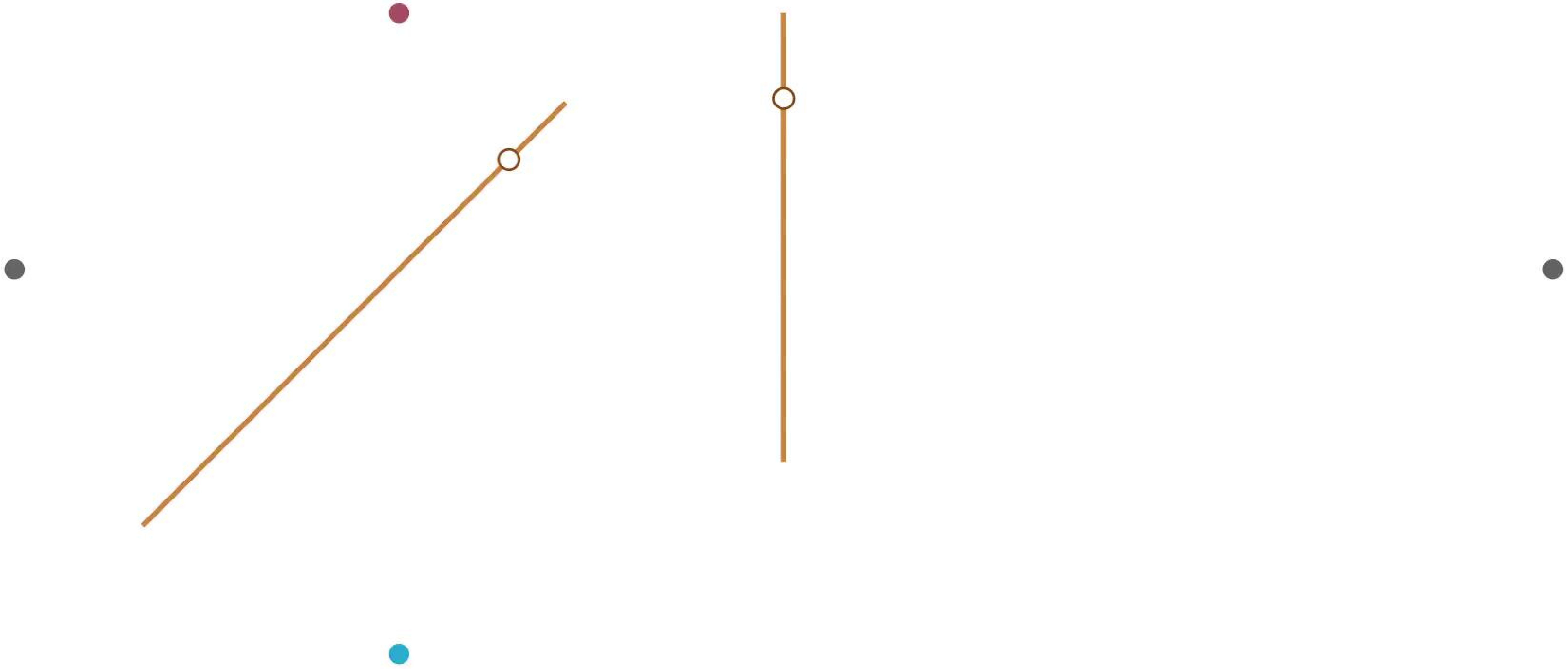
How to trace paths



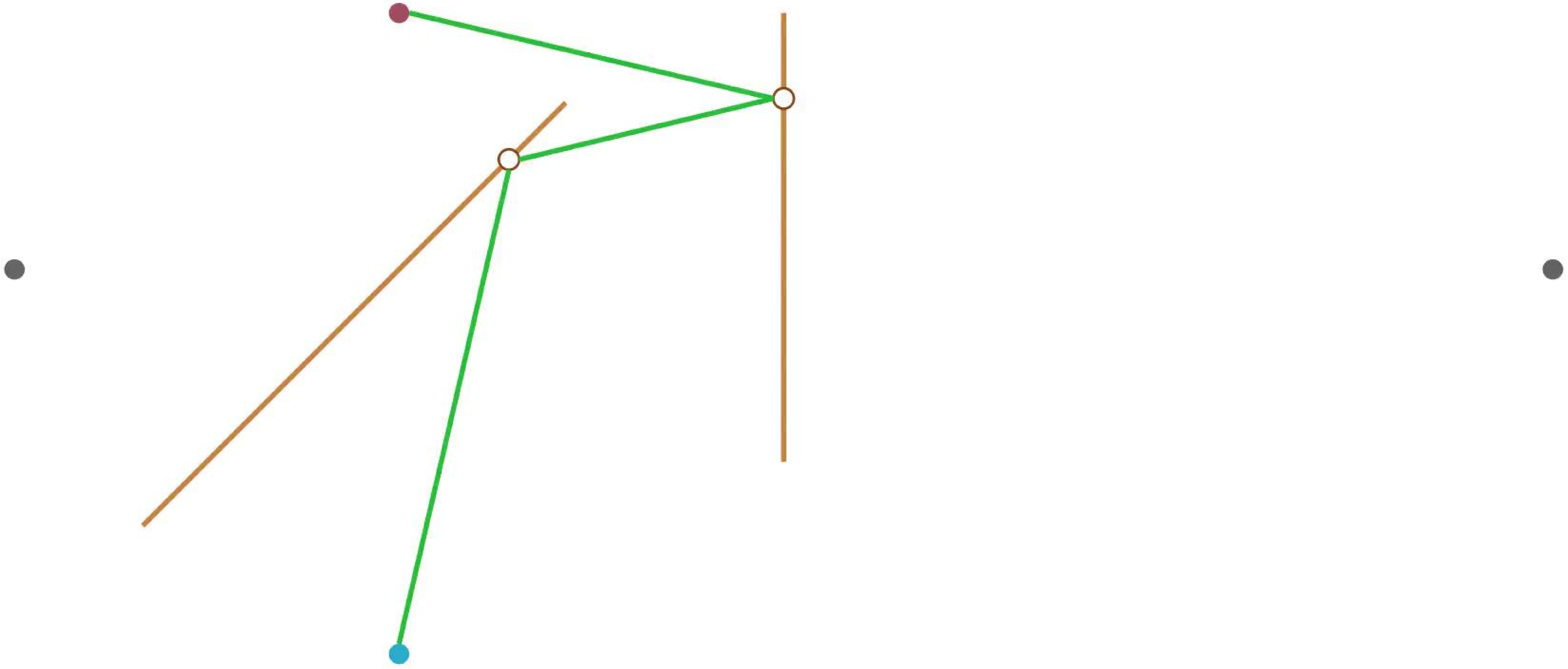
How to trace paths



How to trace paths



How to trace paths



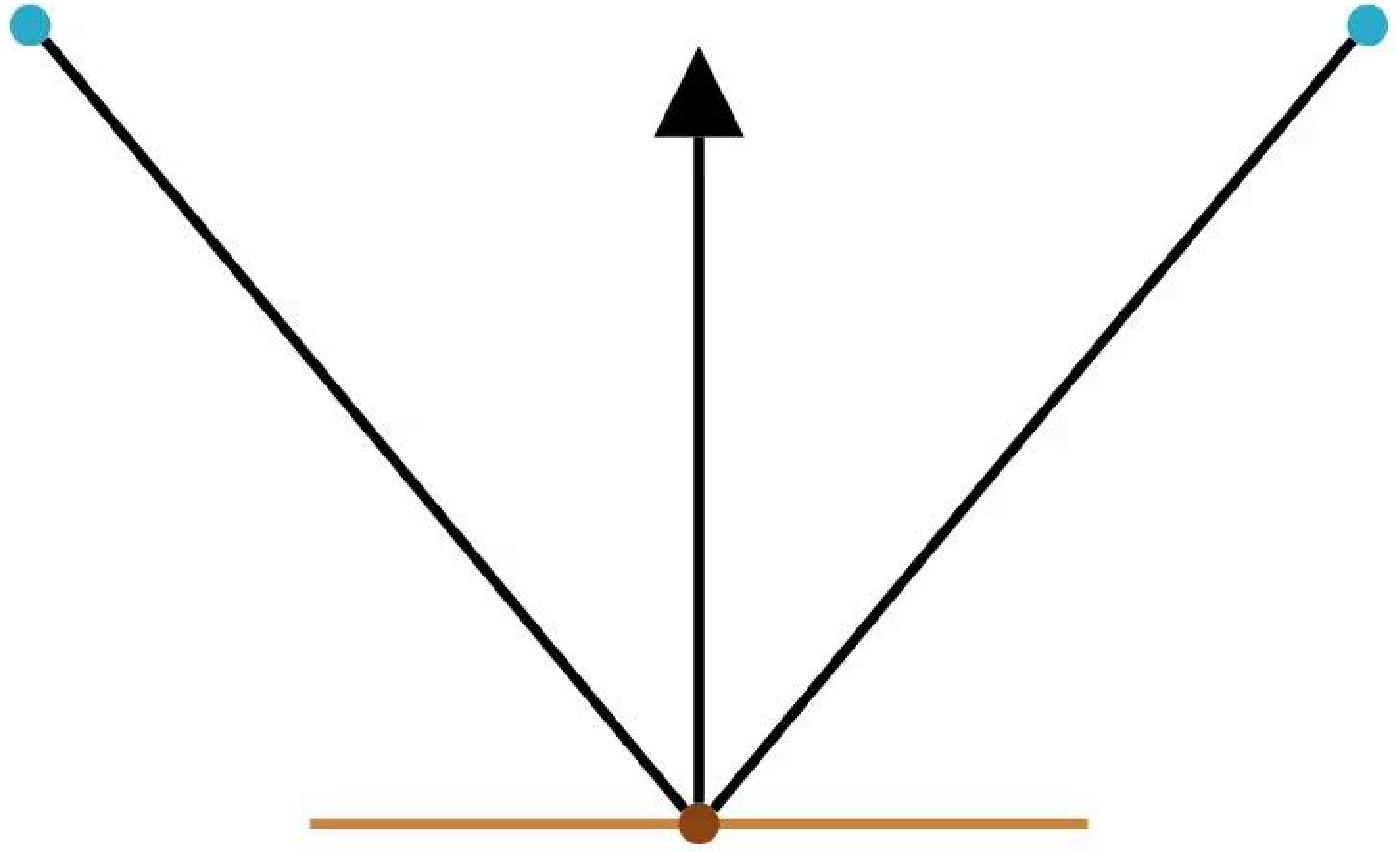
How to trace paths

	Ray Launching	Ray Tracing
Complexity	$\mathcal{O}(N_R)$	$\mathcal{O}(N^o)$
Paths missed	Unknown	None
Scalability	Good	Bad
Accuracy	Good	Excellent

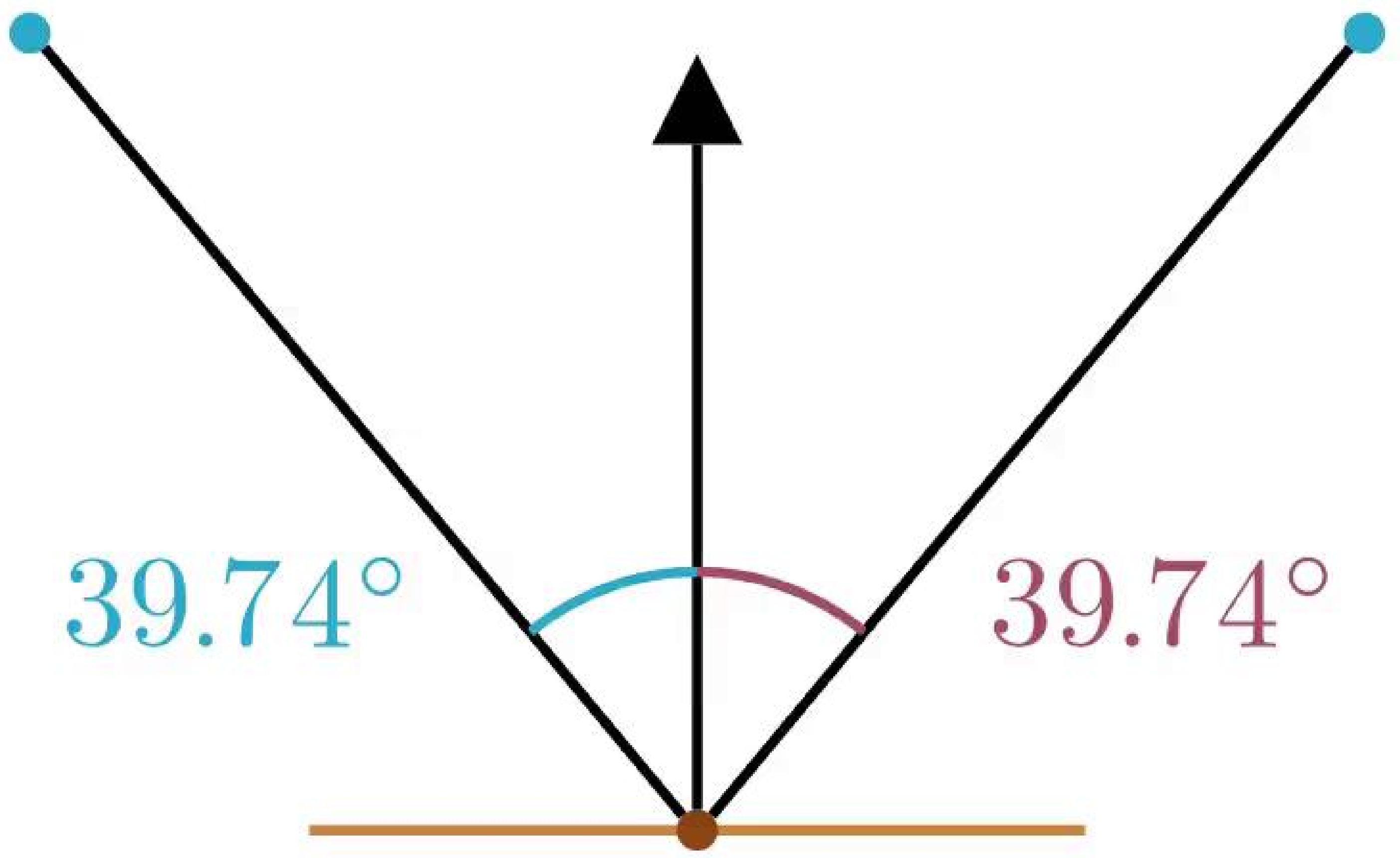
How to trace paths

What if we want to simulate something else than reflection on planar surfaces?

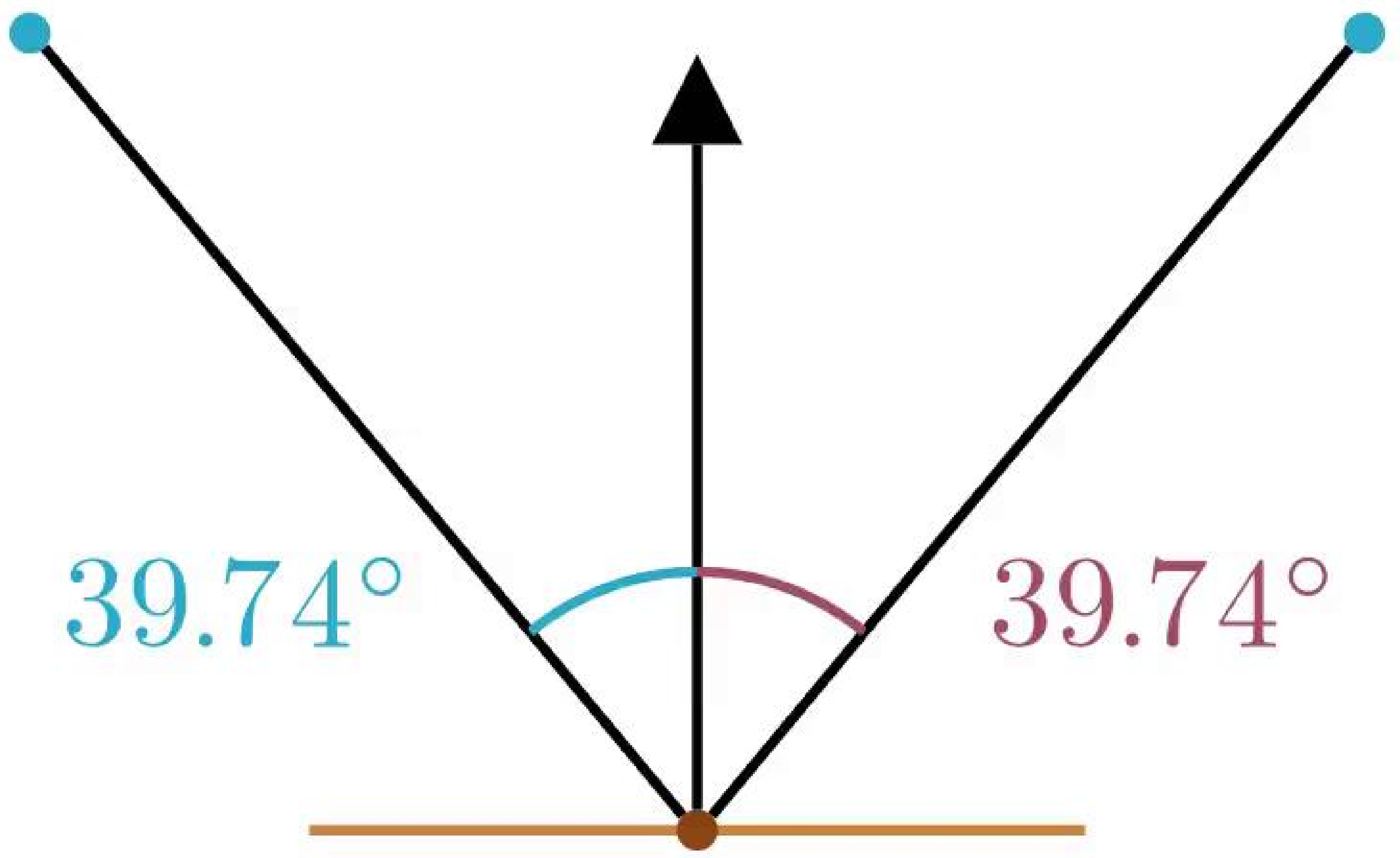
How to trace paths



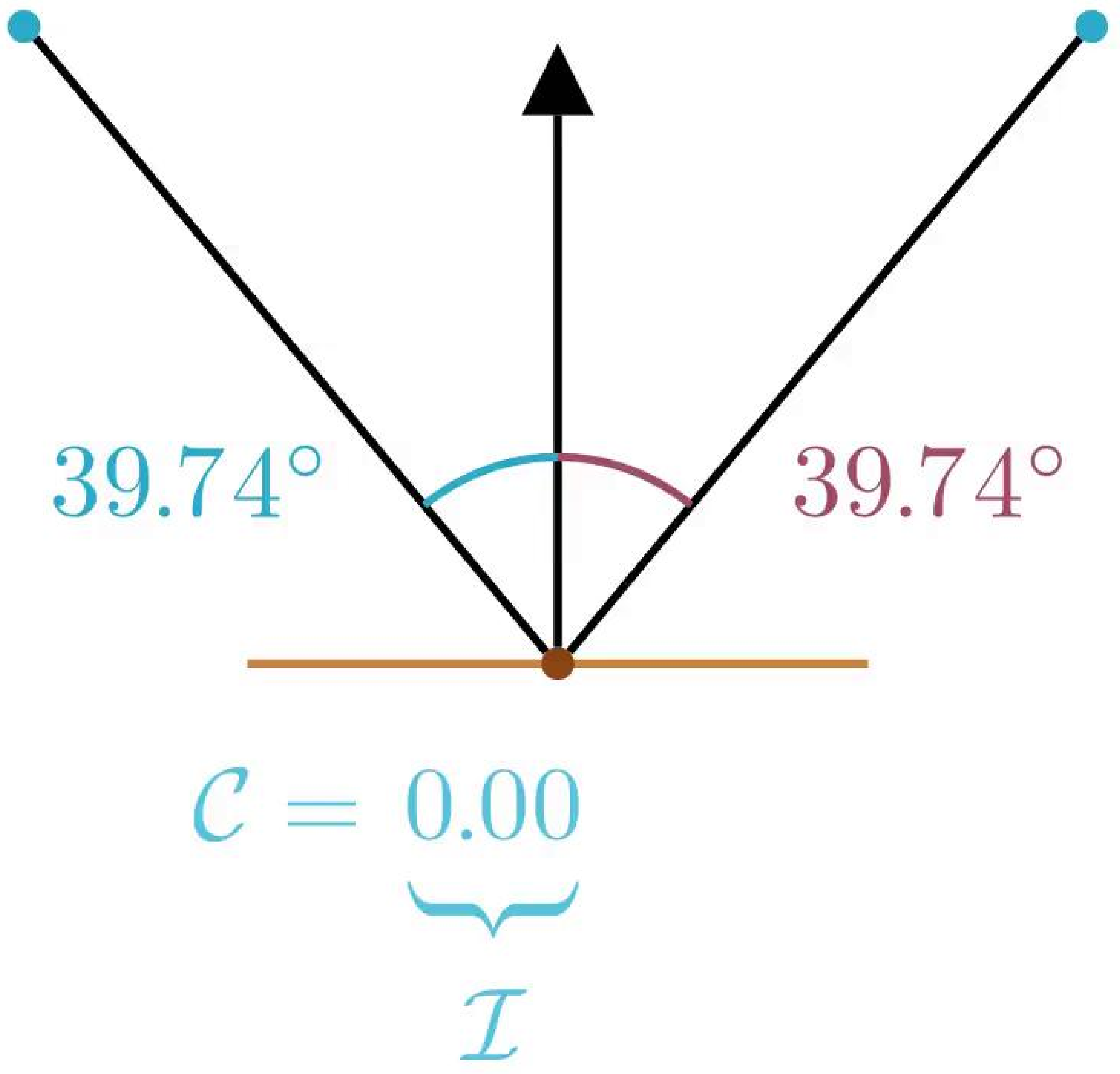
How to trace paths



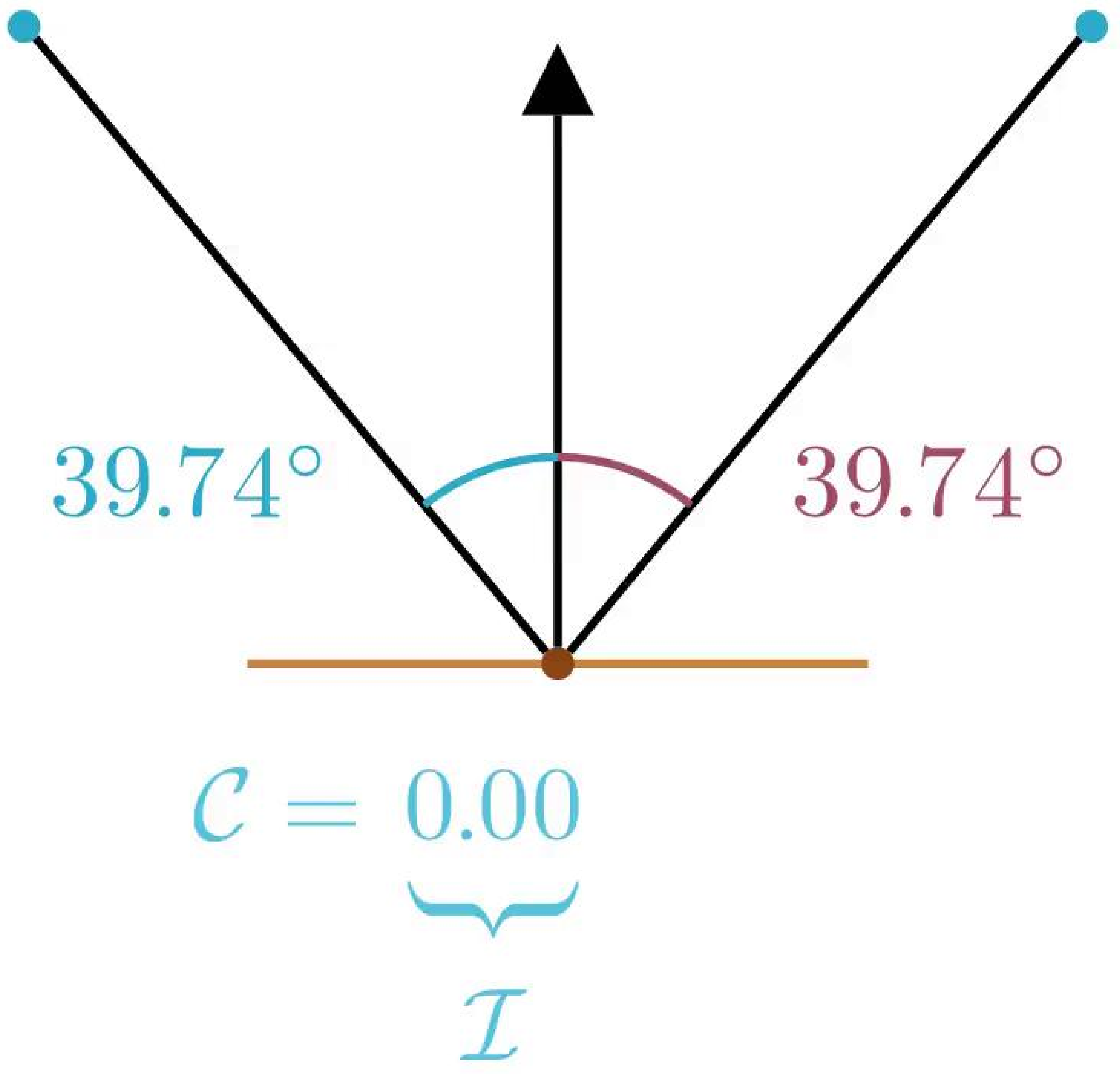
How to trace paths



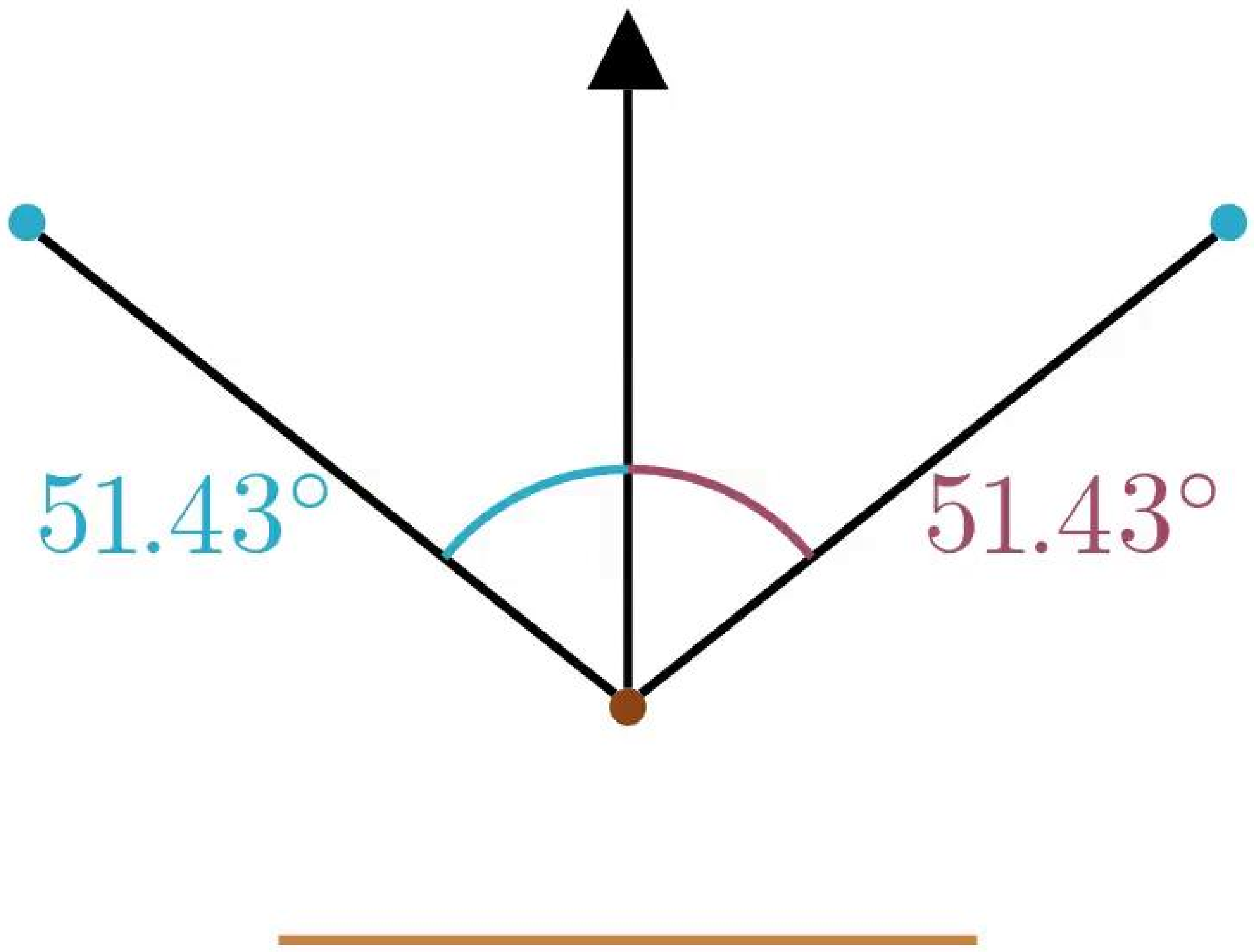
How to trace paths



How to trace paths

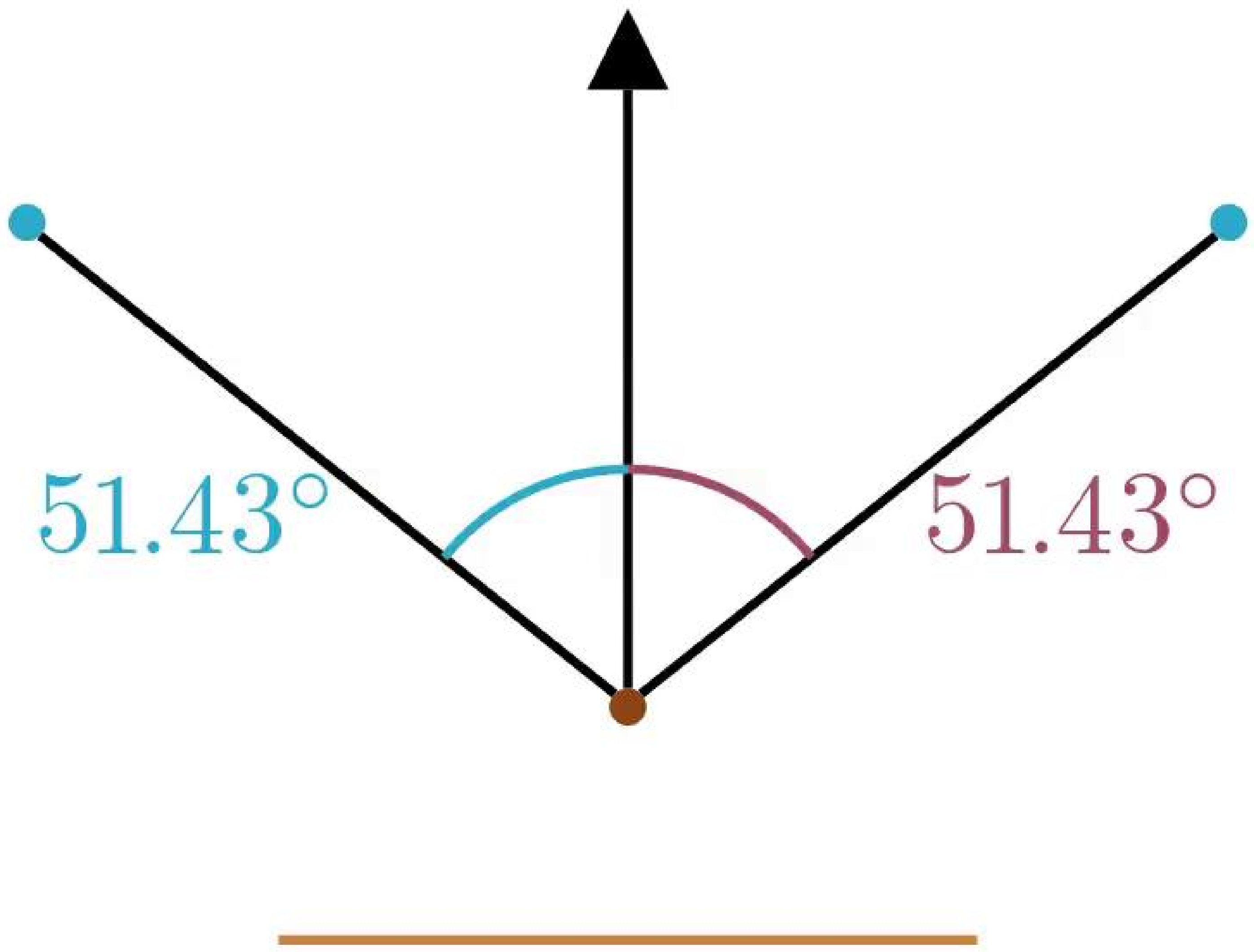


How to trace paths



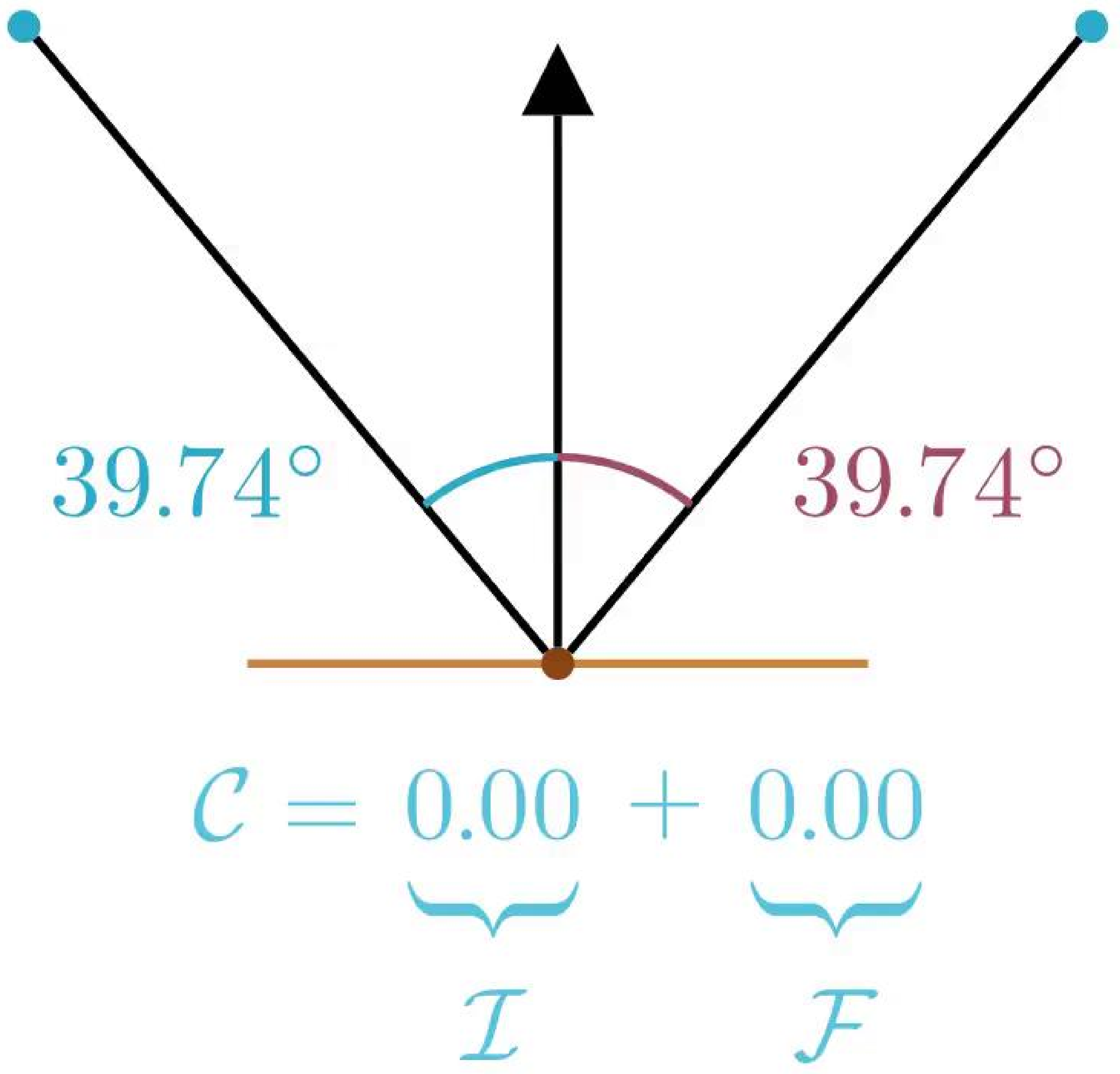
$$\underbrace{\mathcal{C} = 0.00}_{\mathcal{I}}$$

How to trace paths



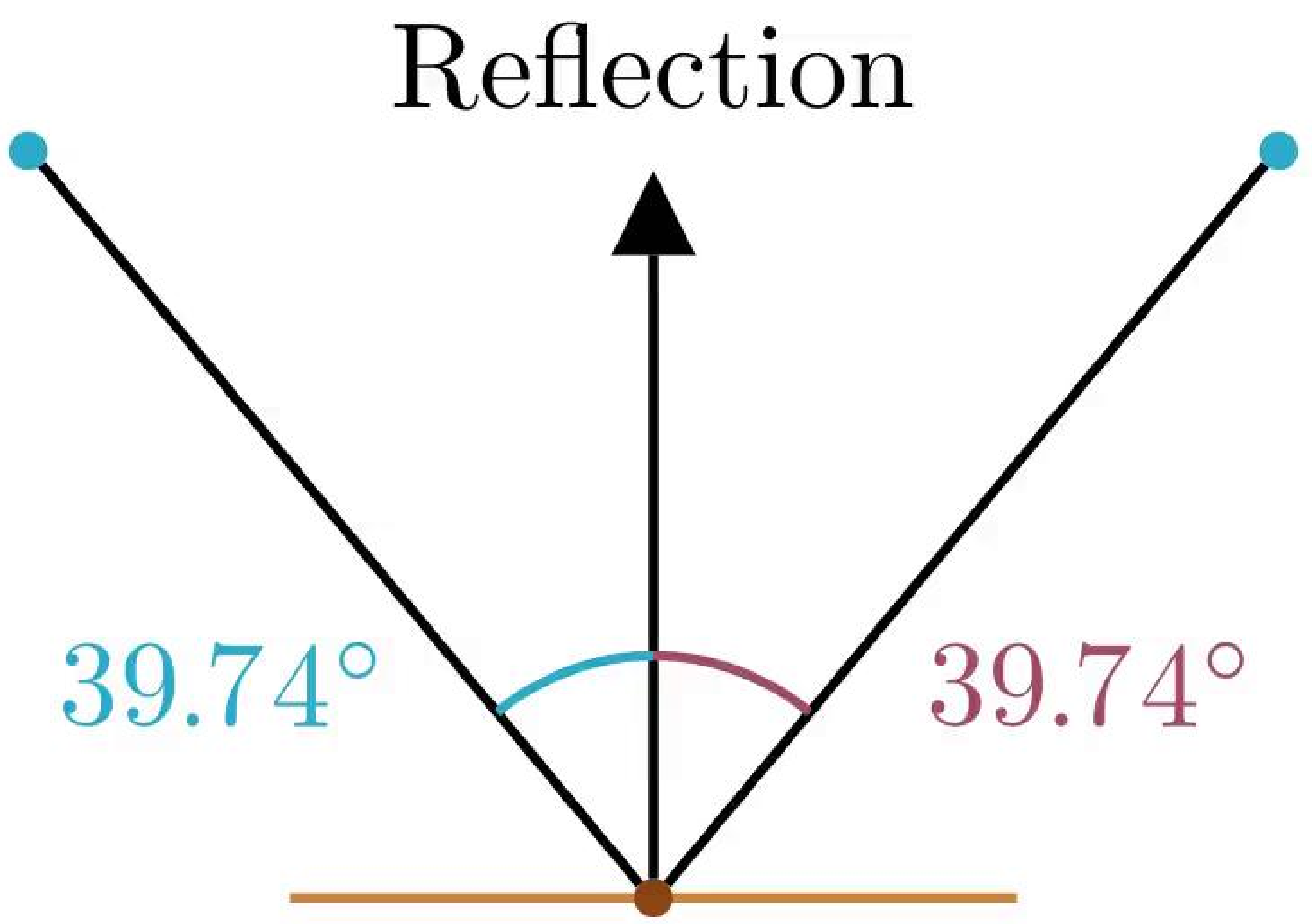
$$\mathcal{C} = \underbrace{0.00}_{\mathcal{I}} + \underbrace{1.00}_{\mathcal{F}}$$

How to trace paths



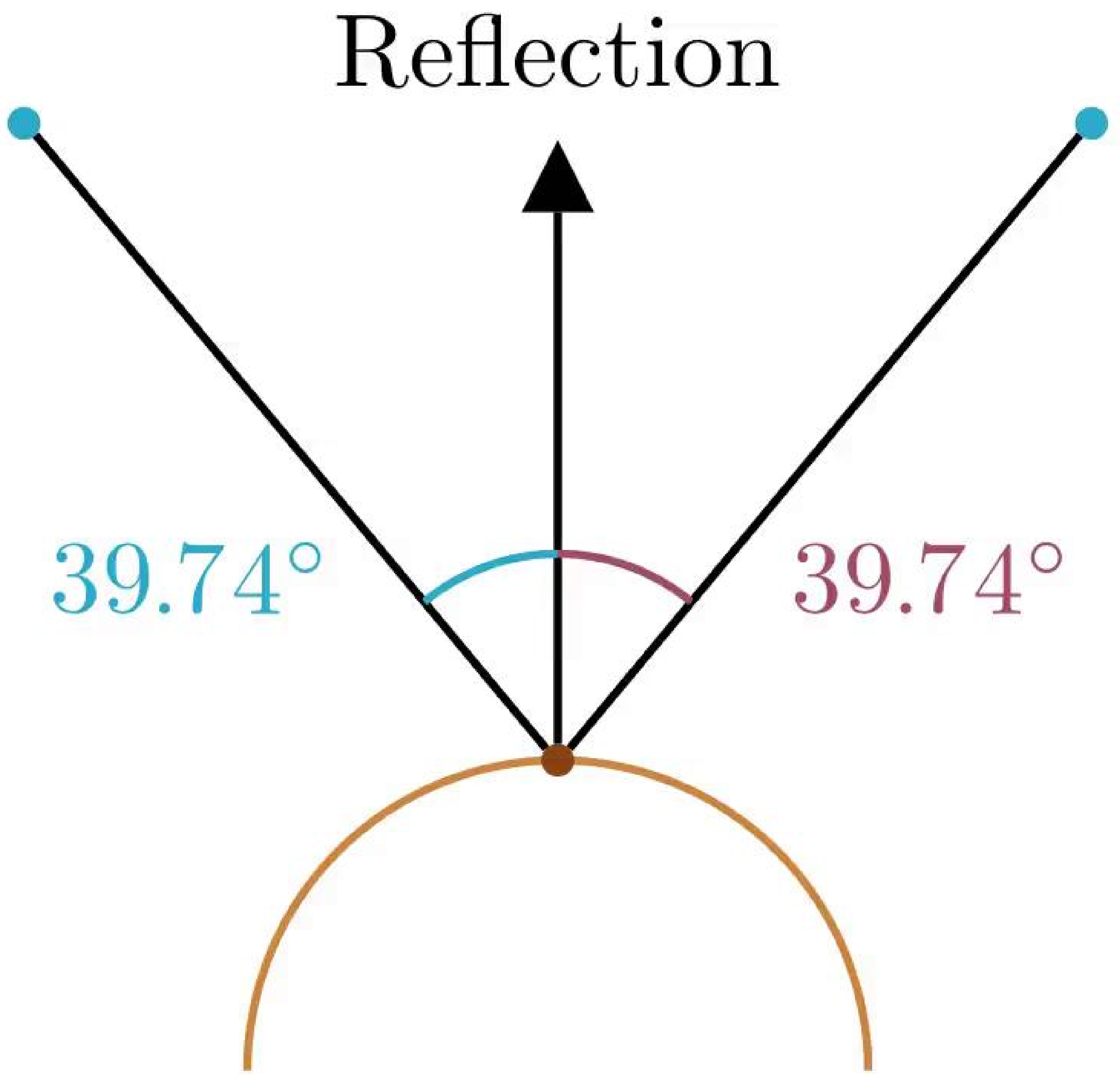
How to trace paths

$$\mathcal{I} \sim \hat{\boldsymbol{r}} = \hat{\boldsymbol{i}} - 2\langle \hat{\boldsymbol{i}}, \hat{\boldsymbol{n}} \rangle \hat{\boldsymbol{n}}$$



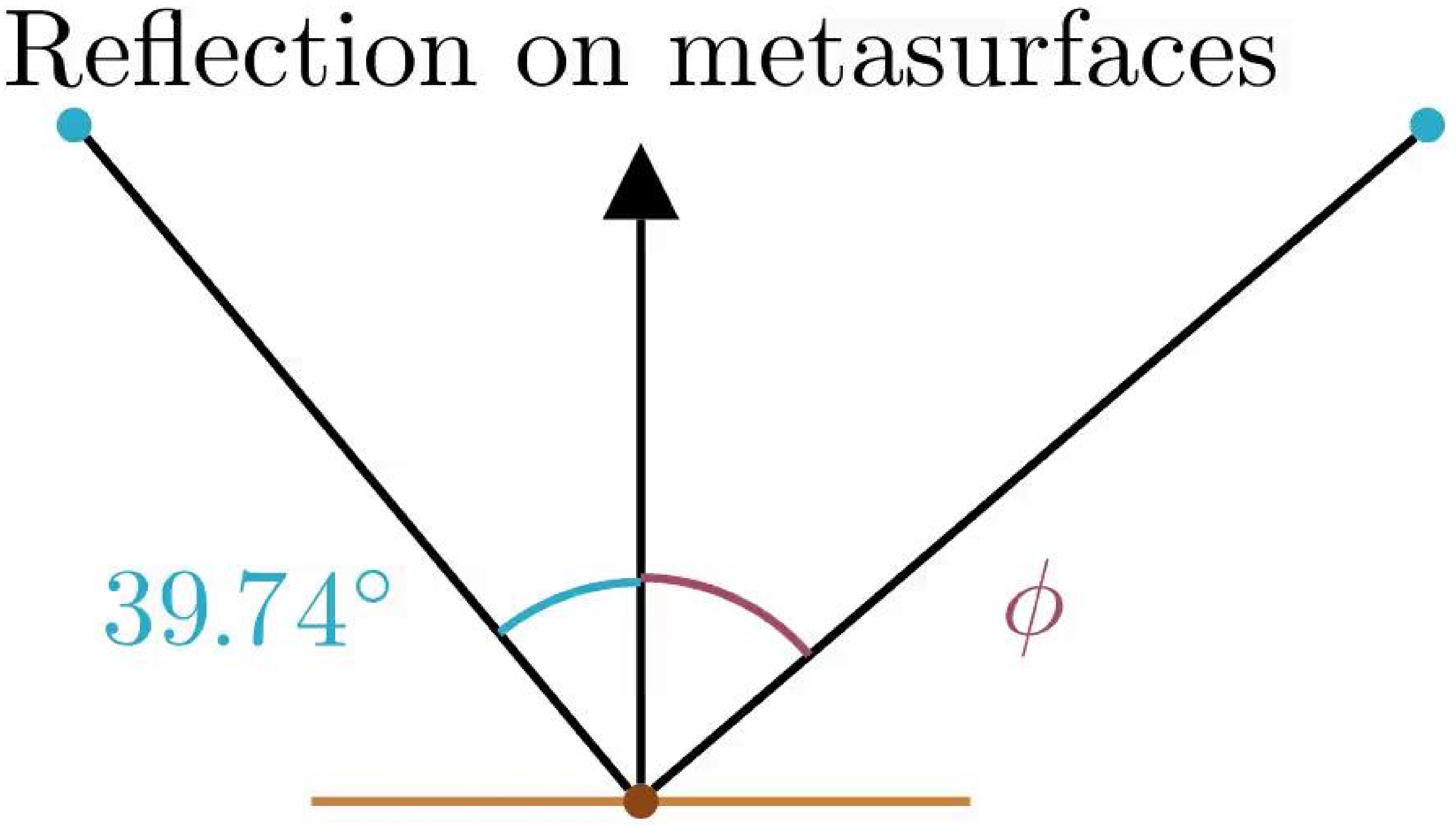
How to trace paths

$$\mathcal{I} \sim \hat{\boldsymbol{r}} = \hat{\boldsymbol{\imath}} - 2\langle \hat{\boldsymbol{\imath}}, \hat{\boldsymbol{n}} \rangle \hat{\boldsymbol{n}}$$



How to trace paths

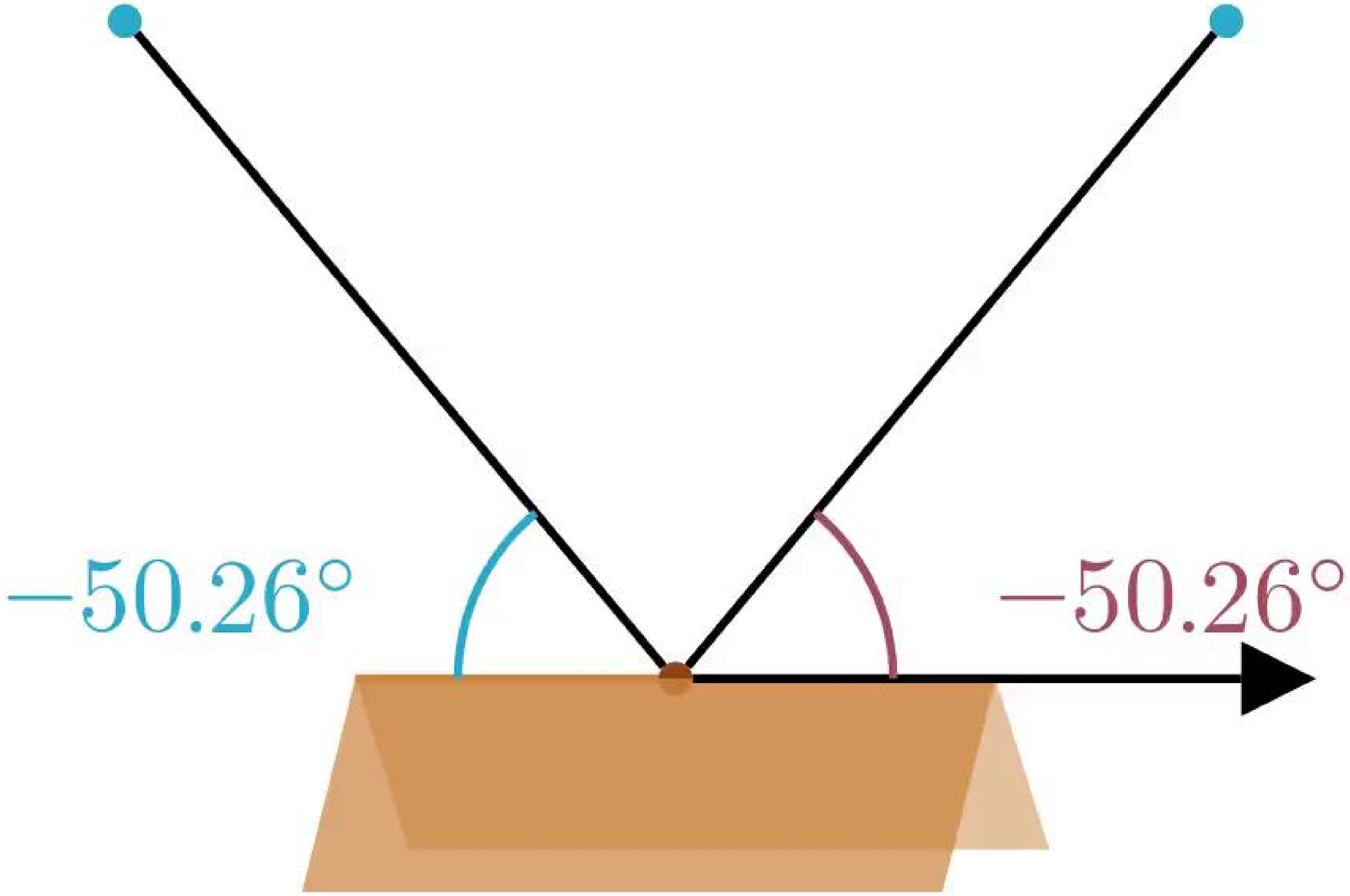
$$\mathcal{I} \sim \boldsymbol{r} = f(\hat{\boldsymbol{n}}, \phi)$$



How to trace paths

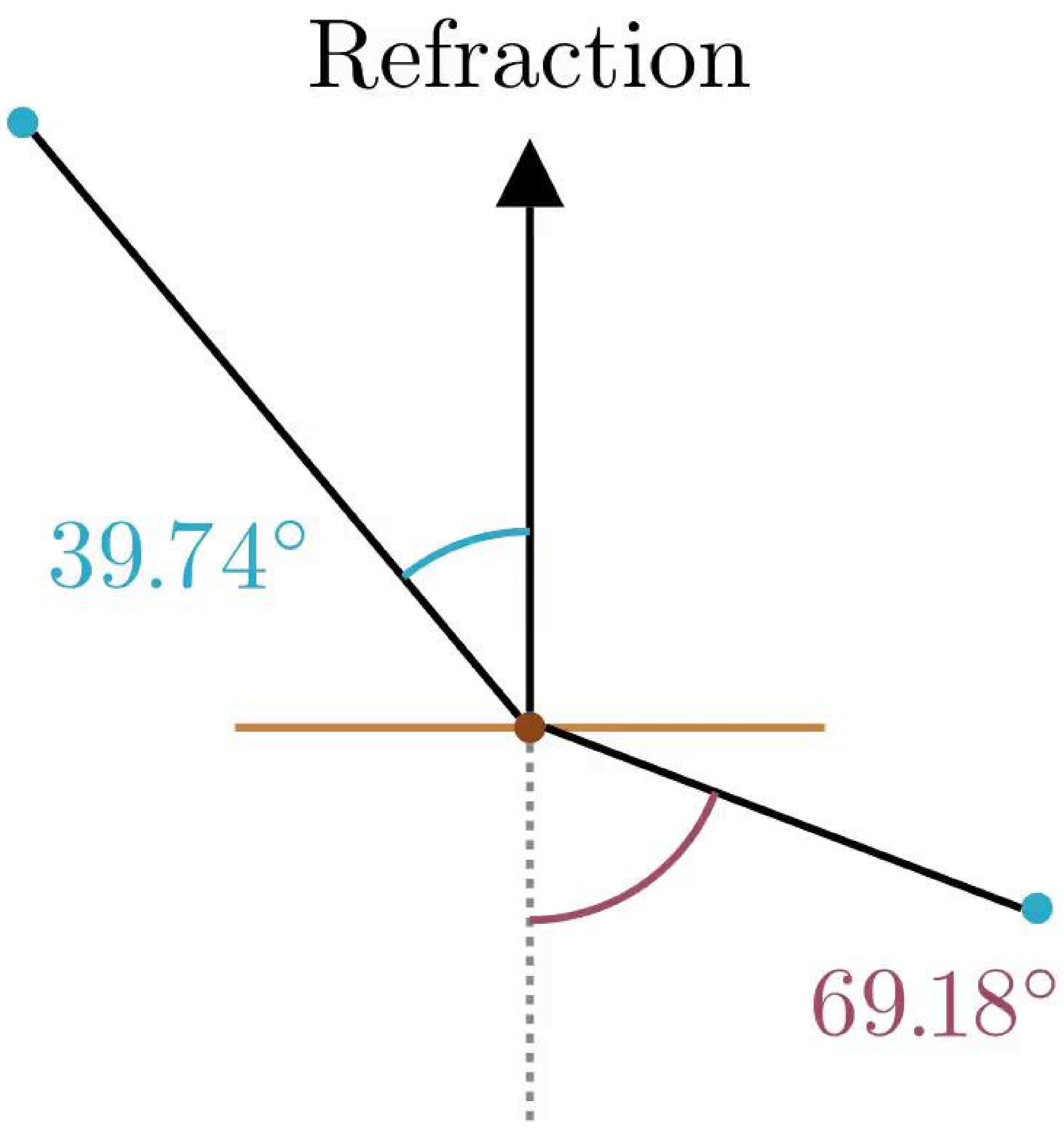
$$\mathcal{I} \sim \frac{\langle \boldsymbol{i}, \hat{\boldsymbol{e}} \rangle}{\|\boldsymbol{i}\|} = \frac{\langle \boldsymbol{d}, \hat{\boldsymbol{e}} \rangle}{\|\boldsymbol{d}\|}$$

Diffraction



How to trace paths

$$\mathcal{I} \sim v_1 \sin(\theta_2) = v_2 \sin(\theta_1)$$



How to trace paths

How to trace paths

$$\underset{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{nt}}{\text{minimize}}\ \mathcal{C}(\boldsymbol{\mathcal{X}}) := \|\mathcal{I}(\boldsymbol{\mathcal{X}})\|^2 + \|\mathcal{F}(\boldsymbol{\mathcal{X}})\|^2$$

How to trace paths

$$\underset{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_t}}{\text{minimize}}\ \mathcal{C}(\boldsymbol{\mathcal{X}}) := \|\mathcal{I}(\boldsymbol{\mathcal{X}})\|^2 + \|\mathcal{F}(\boldsymbol{\mathcal{X}})\|^2$$

where n_t is the total number of unknowns

How to trace paths

$$\underset{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_t}}{\text{minimize}}\ \mathcal{C}(\boldsymbol{\mathcal{X}}) := \|\mathcal{I}(\boldsymbol{\mathcal{X}})\|^2 + \|\mathcal{F}(\boldsymbol{\mathcal{X}})\|^2$$

where n_t is the total number of unknowns

$$\mathcal{C}(\boldsymbol{\mathcal{X}}) = 0$$

How to trace paths

$$\underset{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_t}}{\text{minimize}}\ \mathcal{C}(\boldsymbol{\mathcal{X}}) := \|\mathcal{I}(\boldsymbol{\mathcal{X}})\|^2 + \|\mathcal{F}(\boldsymbol{\mathcal{X}})\|^2$$

where n_t is the total number of unknowns

$$\mathcal{C}(\boldsymbol{\mathcal{X}}) \leq \epsilon$$

How to trace paths

If we know a mapping s.t. $(x_k, y_k) \leftrightarrow t_k$

$$\underset{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_t}}{\text{minimize}}\; \mathcal{C}(\boldsymbol{\mathcal{X}}) := \|\mathcal{I}(\boldsymbol{\mathcal{X}})\|^2 + \|\mathcal{F}(\boldsymbol{\mathcal{X}})\|^2$$

where n_t is the total number of unknowns

$$\mathcal{C}(\boldsymbol{\mathcal{X}}) \leq \epsilon$$

How to trace paths

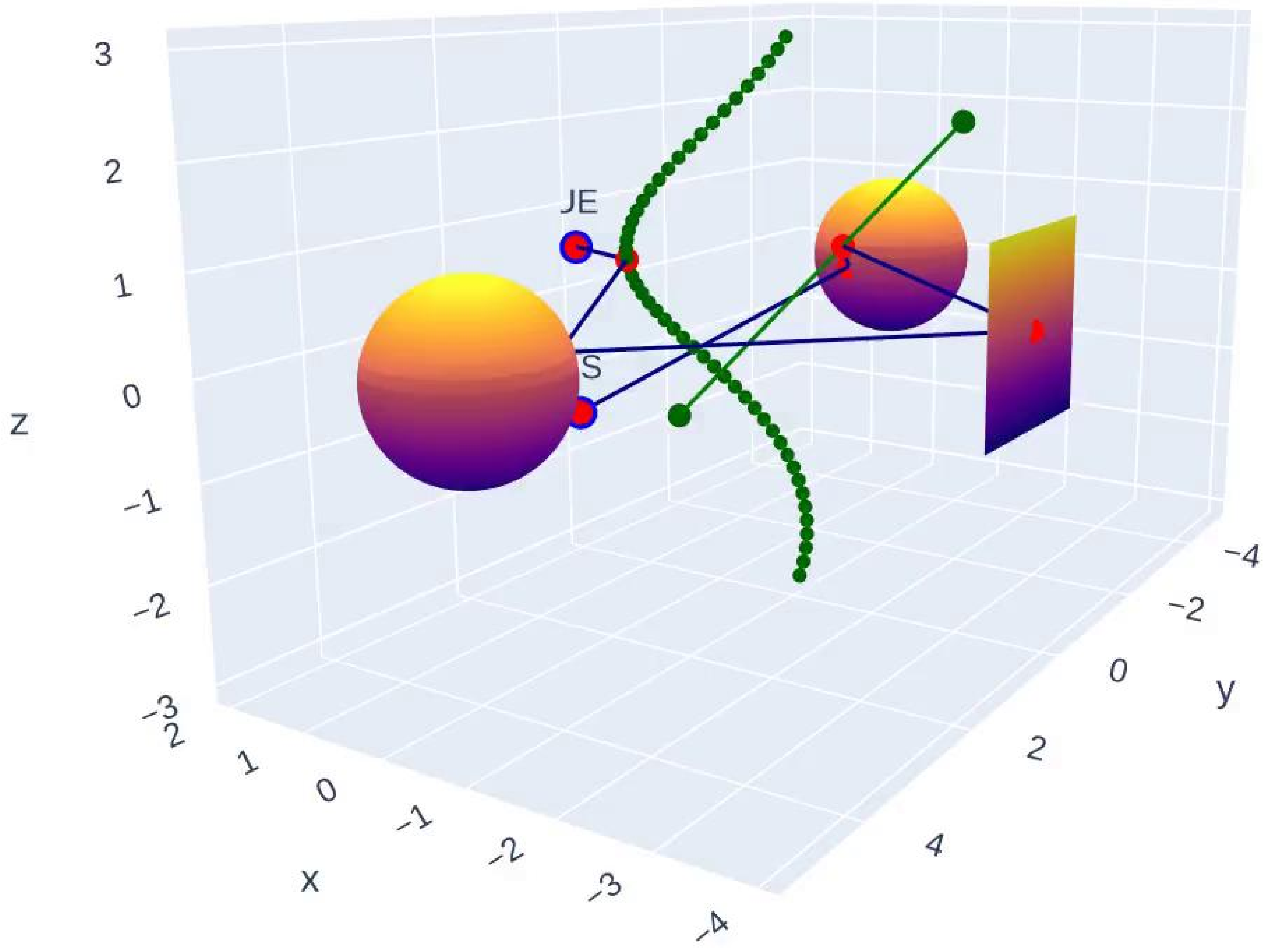
If we know a mapping s.t. $(x_k, y_k) \leftrightarrow t_k$

$$\underset{\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_r}}{\text{minimize}}\, \mathcal{C}(\boldsymbol{\mathcal{X}}(\boldsymbol{\mathcal{T}})) := \|\mathcal{I}(\boldsymbol{\mathcal{X}}(\boldsymbol{\mathcal{T}}))\|^2$$

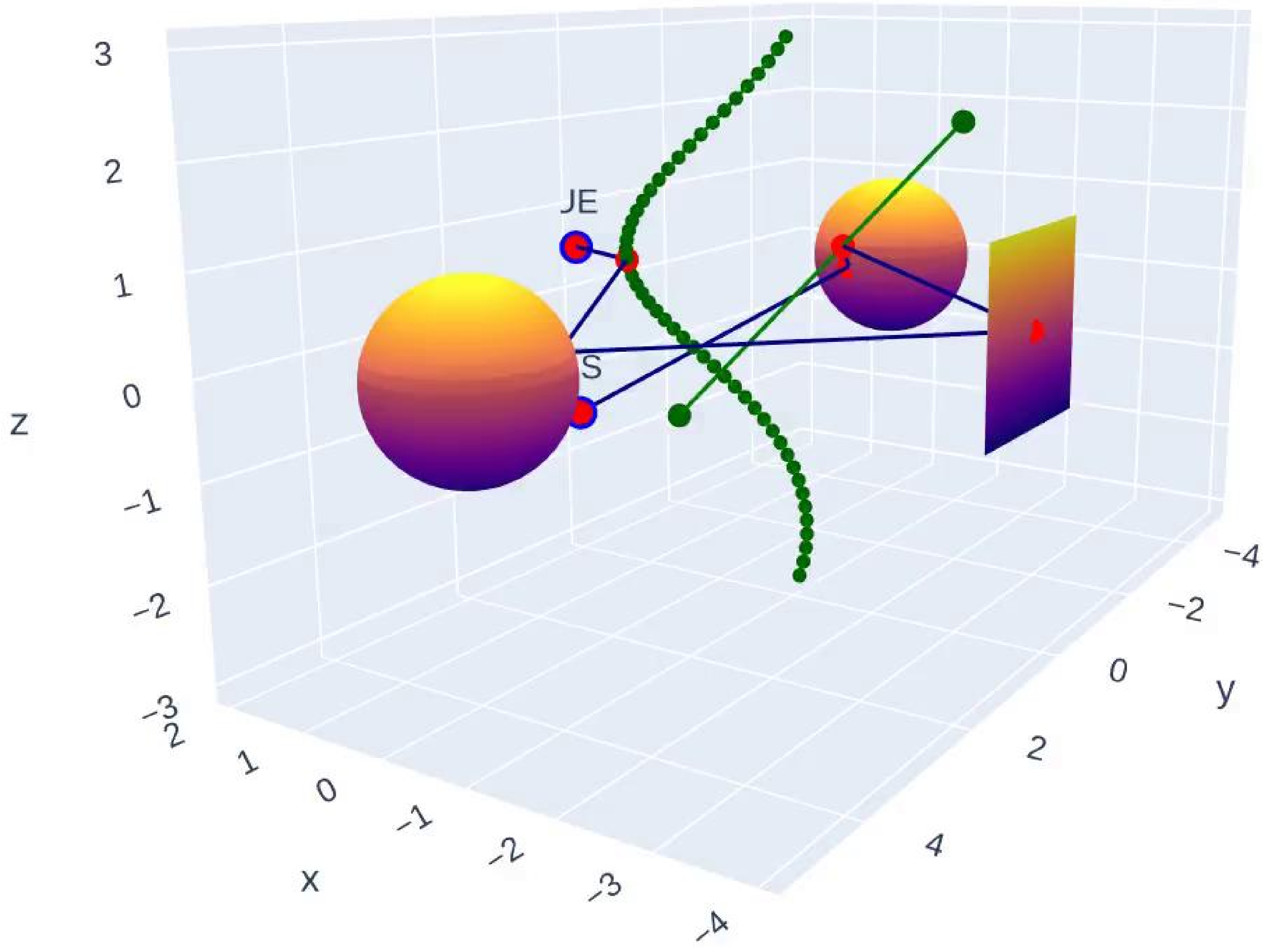
where n_r is the total number of (2d) reflections

$$\mathcal{C}(\boldsymbol{\mathcal{X}}(\boldsymbol{\mathcal{T}})) \leq \epsilon$$

How to trace paths



How to trace paths



How to trace paths

	Image	FPT	MPT
Complexity	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot n_{\text{iter}})$	$\mathcal{O}(n \cdot n_{\text{iter}})$
Objects	Planes	All	Any*
Types	LOS+R	All	All+Custom
Convexity	N/A	Convex on planar	Non convex
Convergence check	N/A or MPT	None or MPT	self

Differentiable Ray Tracing

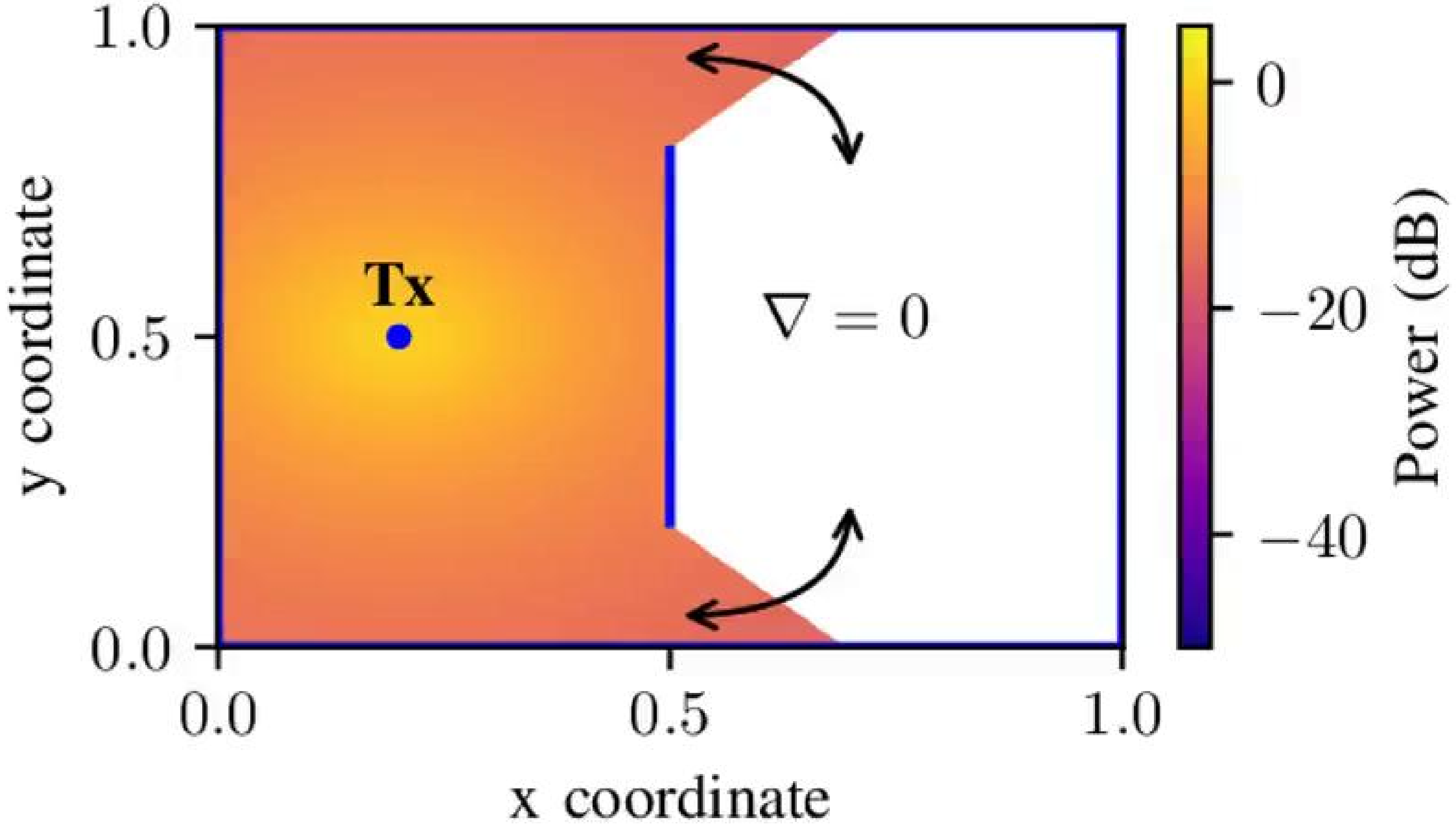
- How to compute derivatives;
- Zero-gradient and discontinuity issues;
- Smoothing technique;
- Optimization example.

Differentiable Ray Tracing

How to compute derivatives?

- symbolically;
- using finite-differences;
- ... with automatic differentiation!

Differentiable Ray Tracing



$$\theta(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{otherwise}, \end{cases}$$

Differentiable Ray Tracing

$$\lim_{\alpha \to \infty} s(x;\alpha) = \theta(x)$$

[C1] $\lim_{x\to-\infty} s(x;\alpha) = 0$ and $\lim_{x\to+\infty} s(x;\alpha) = 1$;

[C2] $s(\cdot;\alpha)$ is monotonically increasing;

[C3] $s(0;\alpha) = \frac{1}{2}$;

[C4] and $s(x;\alpha) - s(0;\alpha) = s(0;\alpha) - s(-x;\alpha)$.

Differentiable Ray Tracing

$$s(x;\alpha) = s(\alpha x). \tag{1}$$

The sigmoid is defined with a real-valued exponential

$$\mathrm{sigmoid}(x;\alpha) = \frac{1}{1+e^{-\alpha x}}, \tag{2}$$

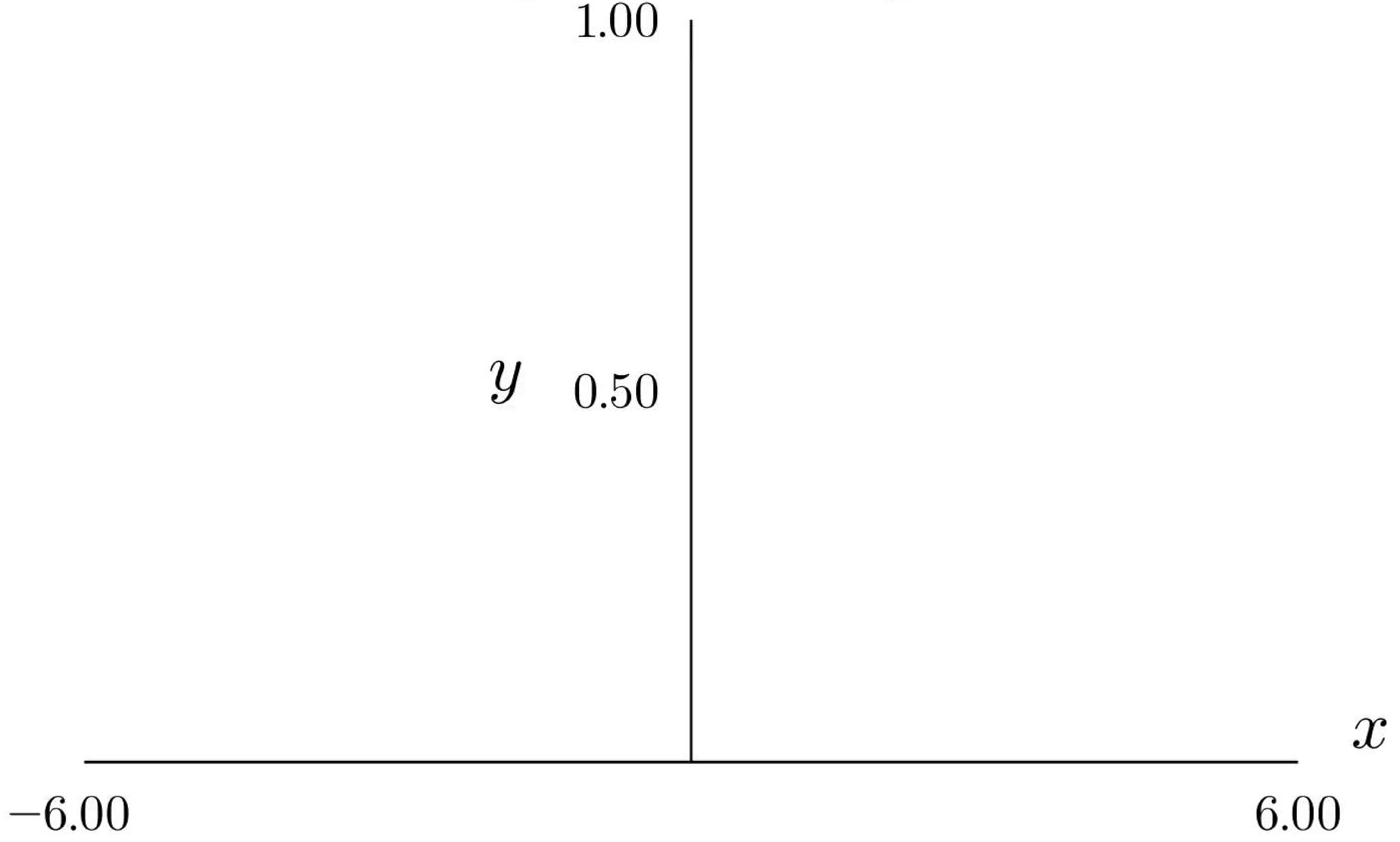
and the hard sigmoid is the piecewise linear function defined by

$$\text{hard sigmoid}(x;\alpha) = \frac{\mathrm{relu6}(\alpha x + 3)}{6}, \tag{3}$$

where

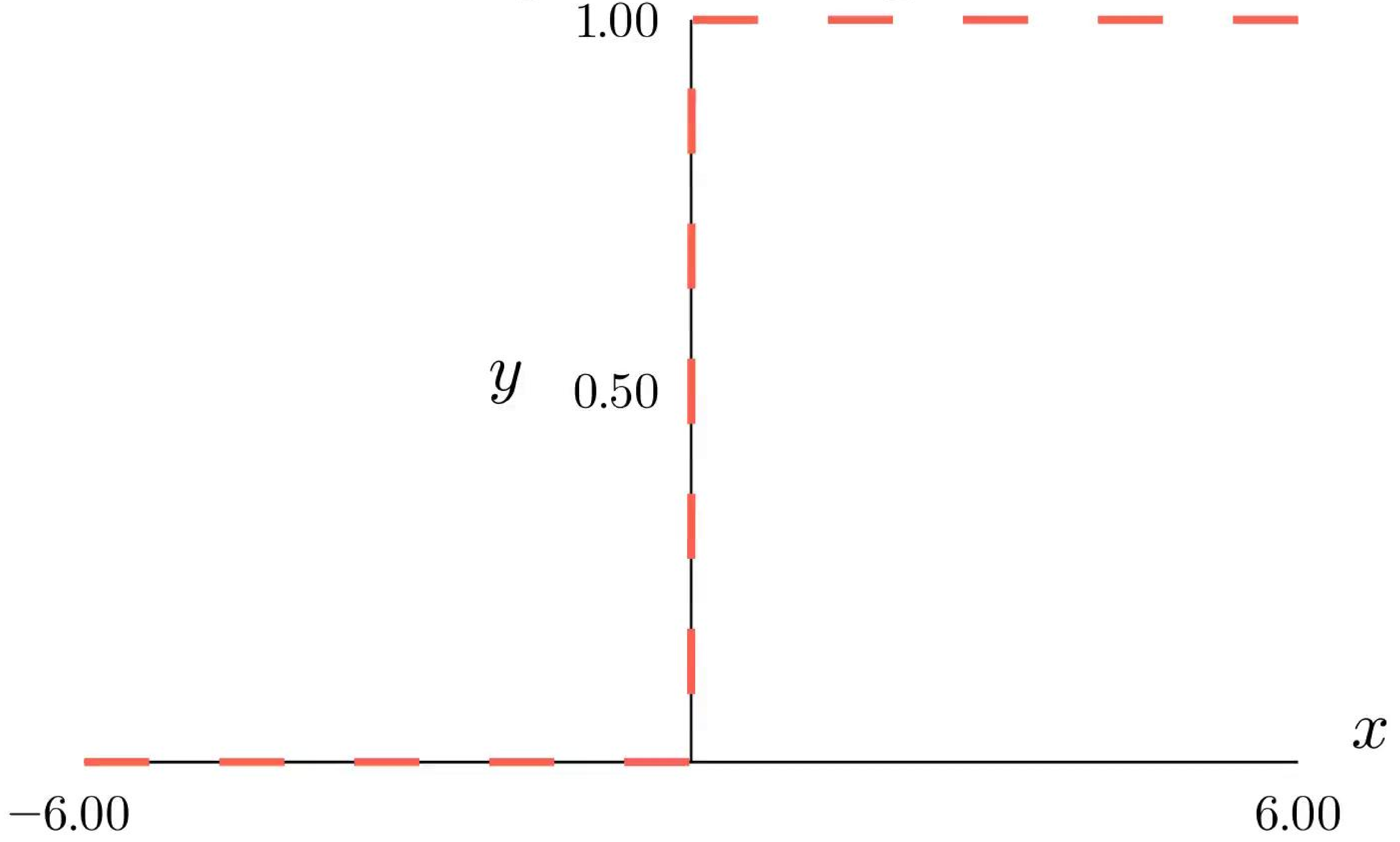
$$\mathrm{relu6}(x) = \min(\max(0,x),6). \tag{4}$$

Differentiable Ray Tracing



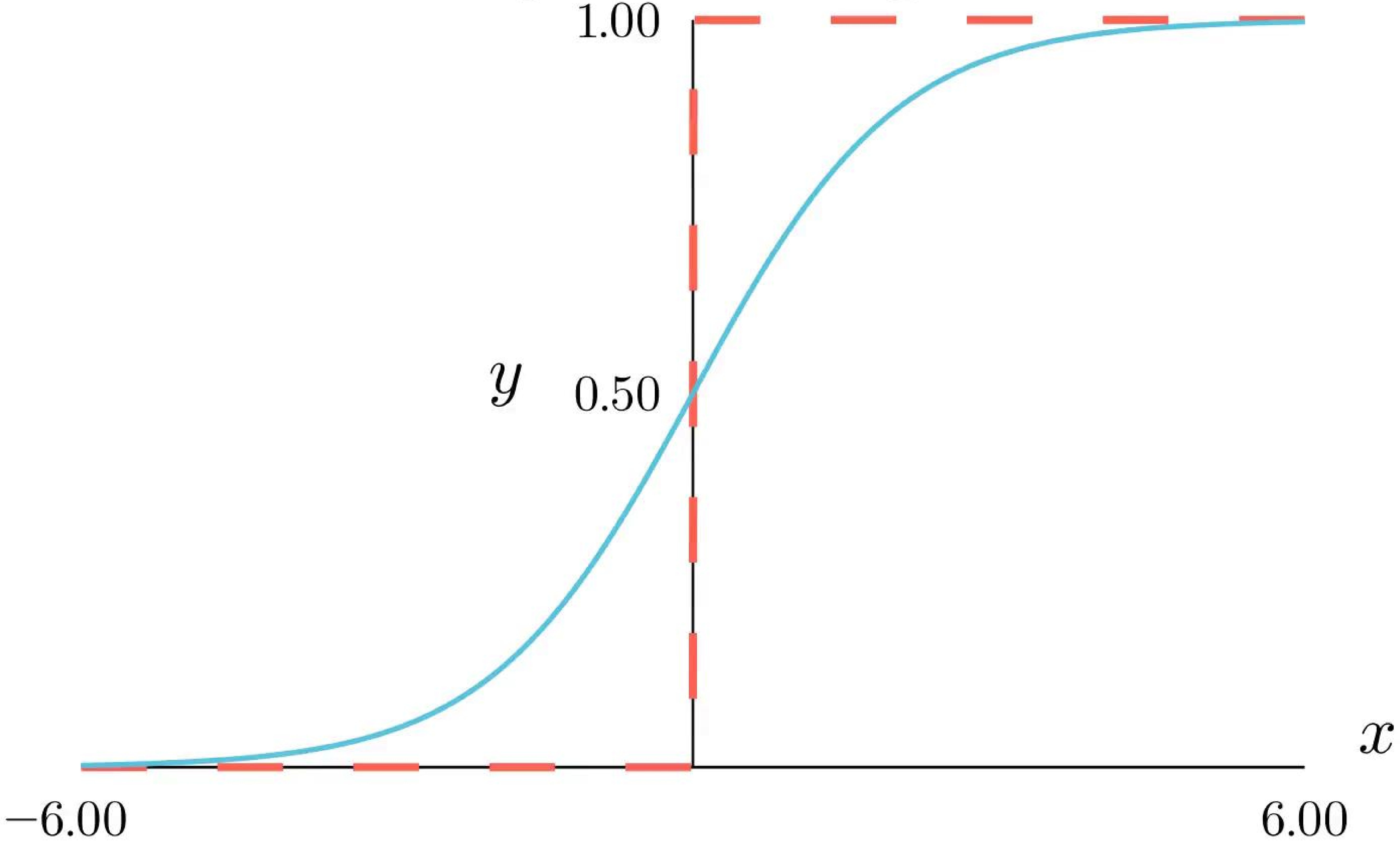
$\alpha = 1.0$

Differentiable Ray Tracing



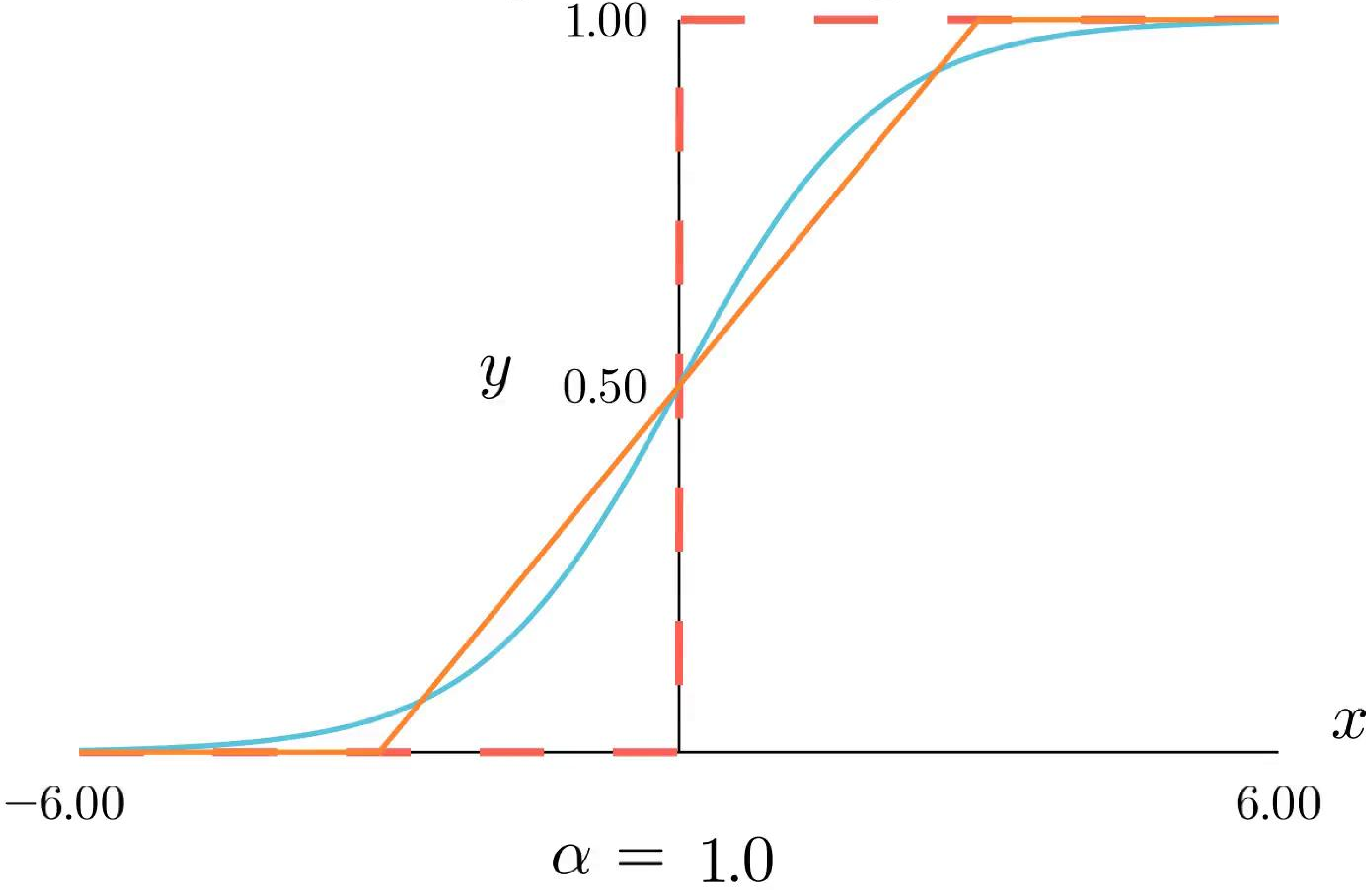
$\alpha = 1.0$

Differentiable Ray Tracing

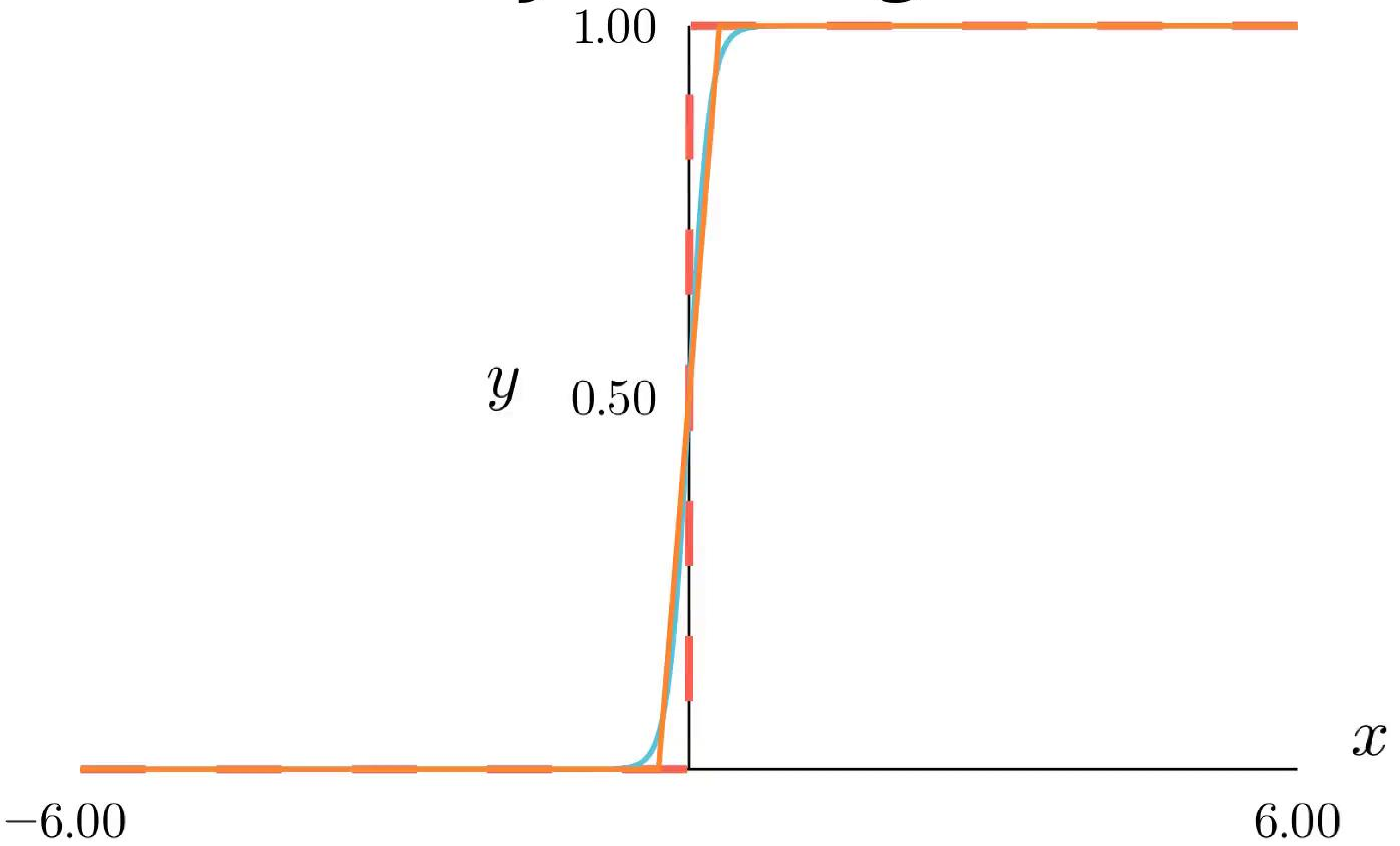


$\alpha = 1.0$

Differentiable Ray Tracing



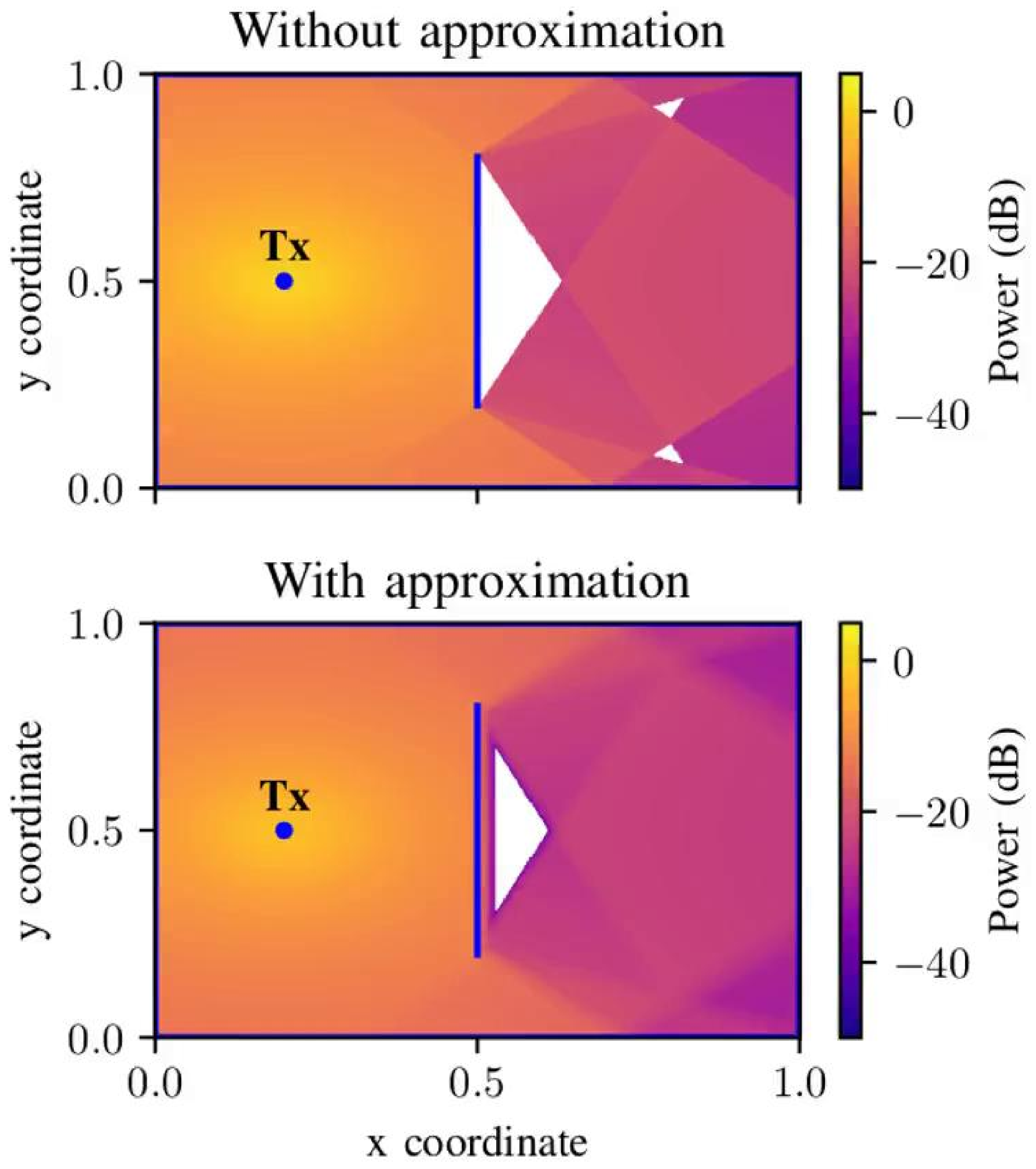
Differentiable Ray Tracing



$\alpha = 10.0$

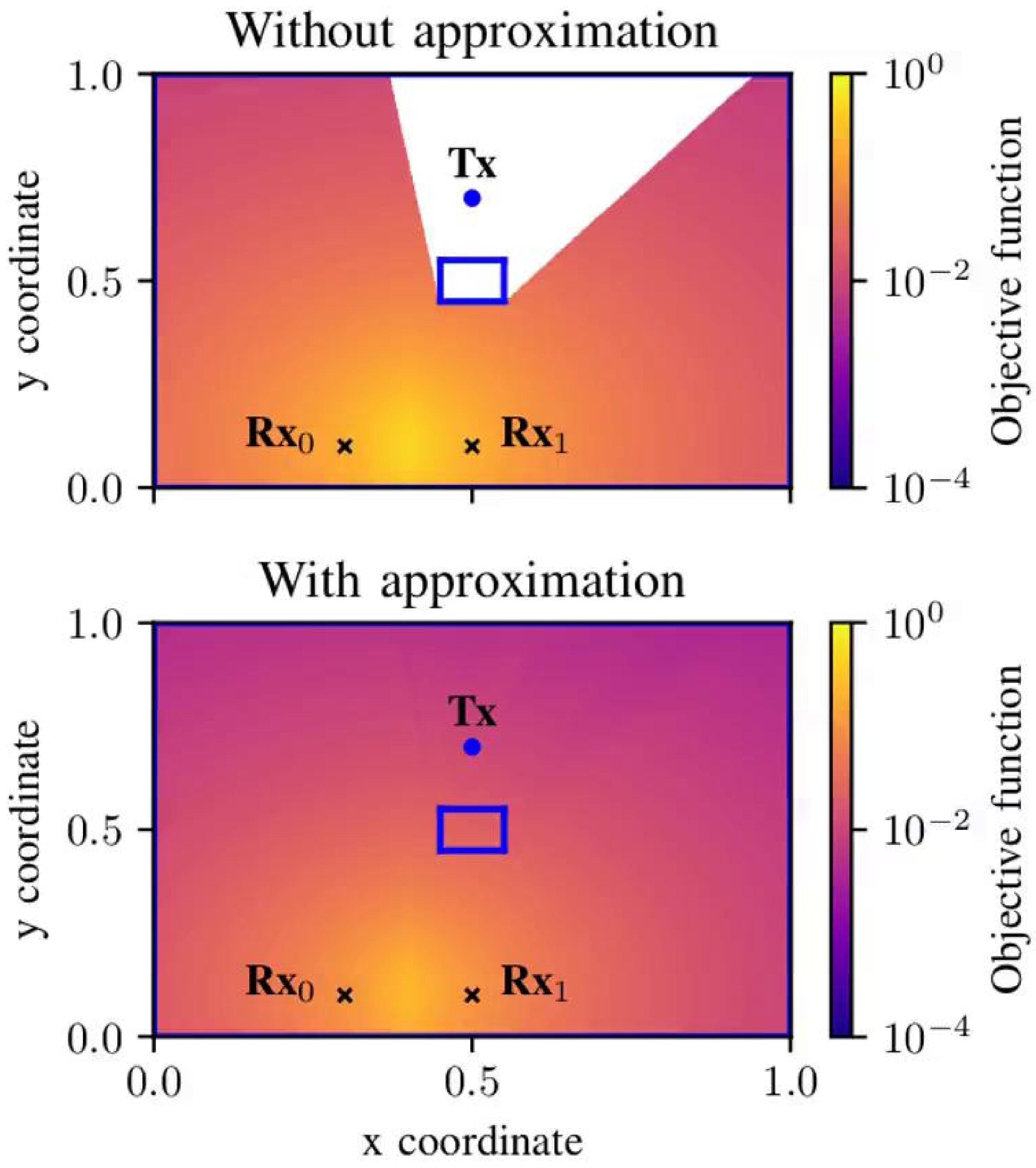
Differentiable Ray Tracing

$$\vec{E}(x,y) = \sum_{\mathcal{P}\in\mathcal{S}} V(\mathcal{P})\left(\, \bar{C}(\mathcal{P}) \cdot \vec{E}(\mathcal{P}_1)\, \right)$$

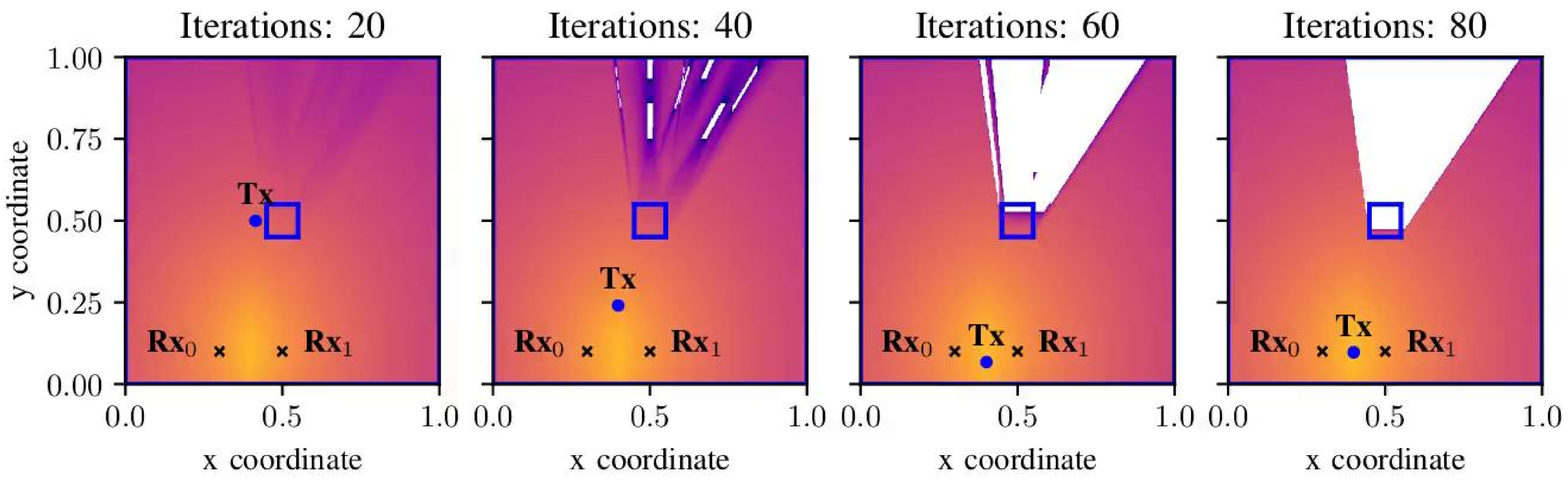


Differentiable Ray Tracing

$$\mathcal{F}(x,y) = \min\left(P_{\mathrm{Rx}_0}(x,y), P_{\mathrm{Rx}_1}(x,y)\right)$$



Differentiable Ray Tracing



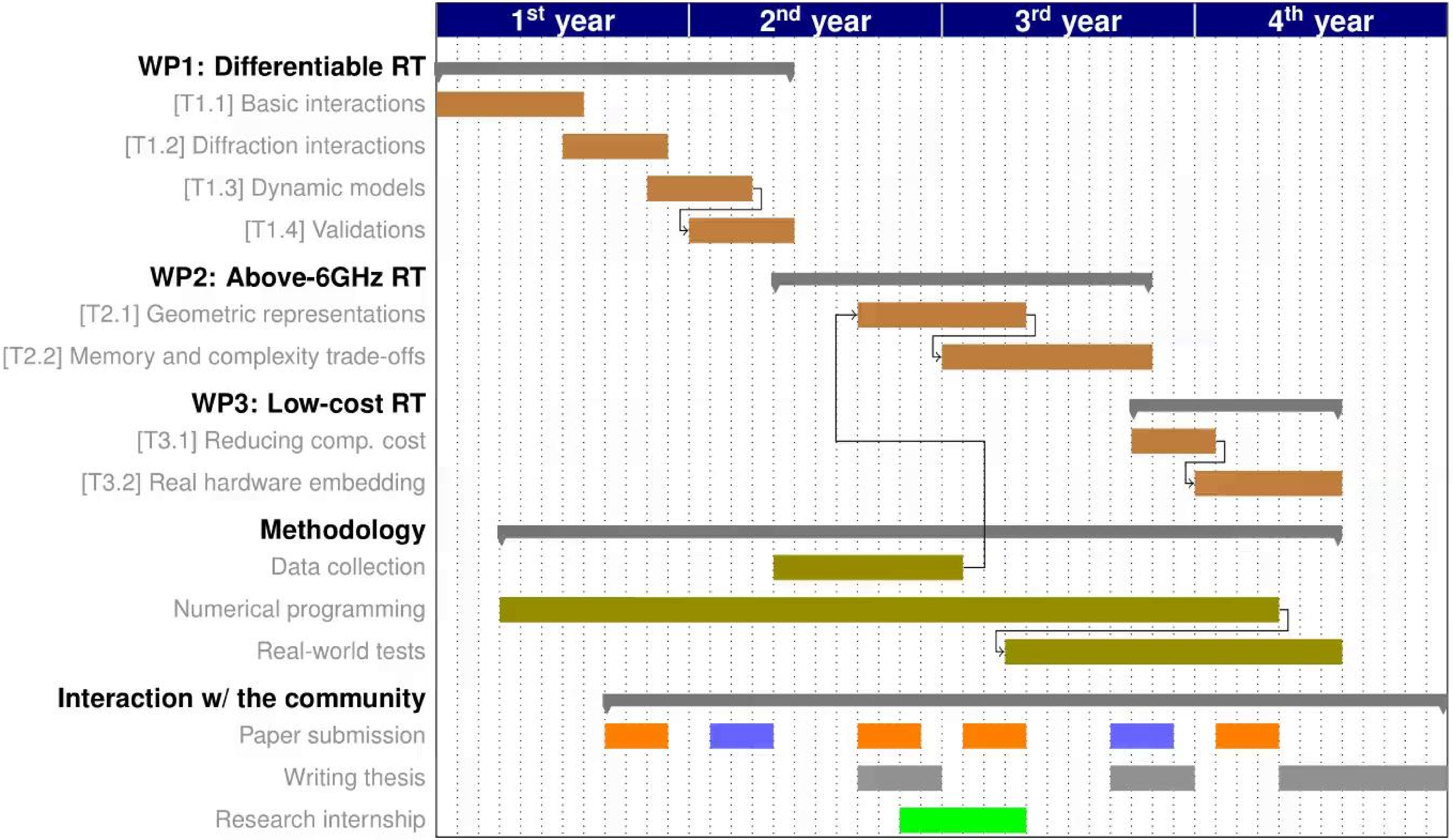
Status of work

Status of work

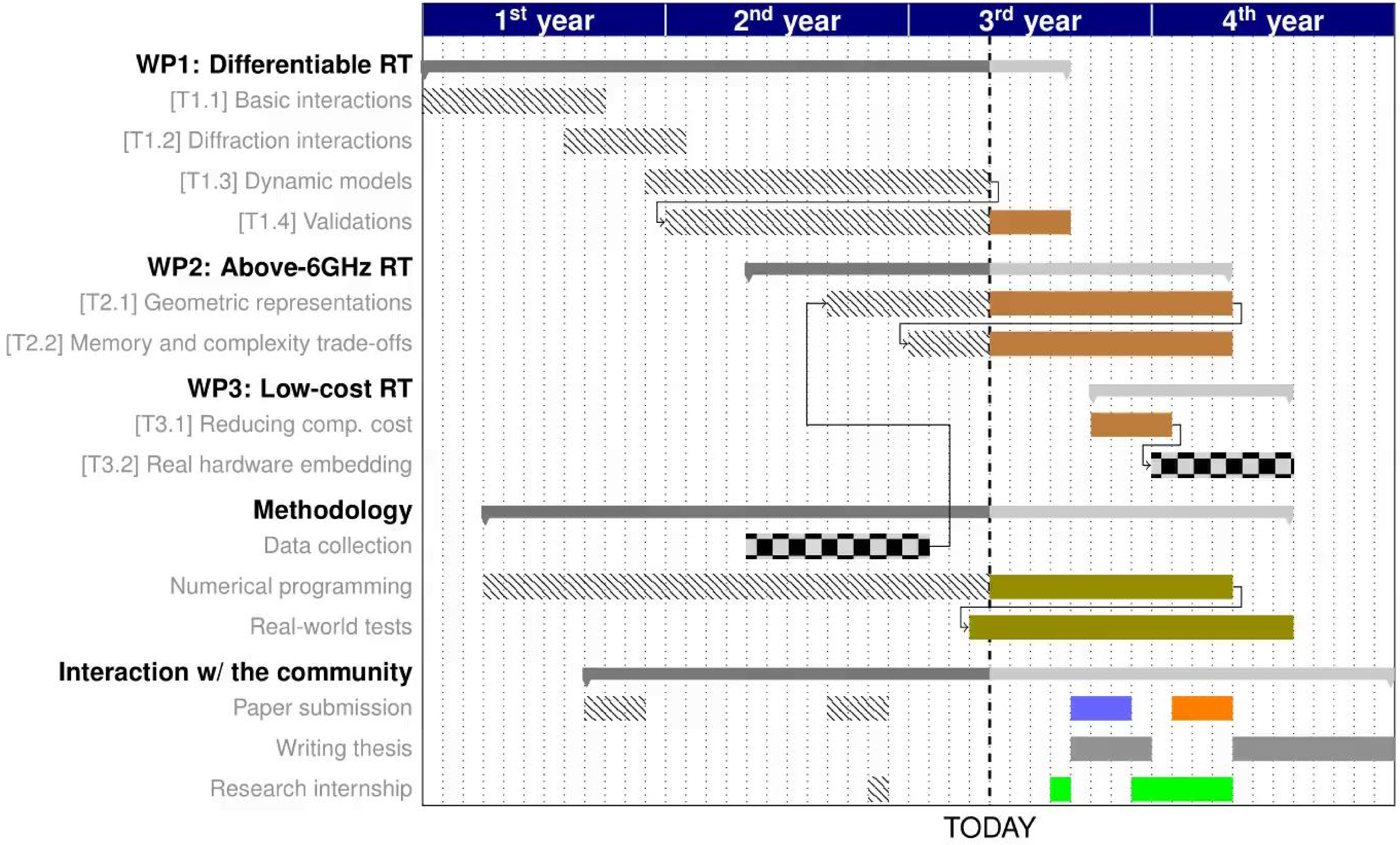
Goals:

- ▷ (G1): Enable RT dynamic scalability;
- ▷ (G2): Novel geometrical environment representations.

Status of work



Status of work



Status of work

Achievements:

- Created general-purpose path tracing method;
- Introduced smoothing techniques in radio-propa. RT;
- Created a 2D Fully DRT open-source Python framework.

Status of work

Future work:

- Extend 2D framework to 3D and realistic scenes;
- Collaborate with Sionna authors for diffraction;
- Cross-validate w/ other tools (Sionna or Huawei's);
- Perform quantitative comp. of RL vs RT;
- Study compat. of MPT w/ good RIS models;
- Learning how to trace paths with ML (deep sets).

Status of work

Collaborations:

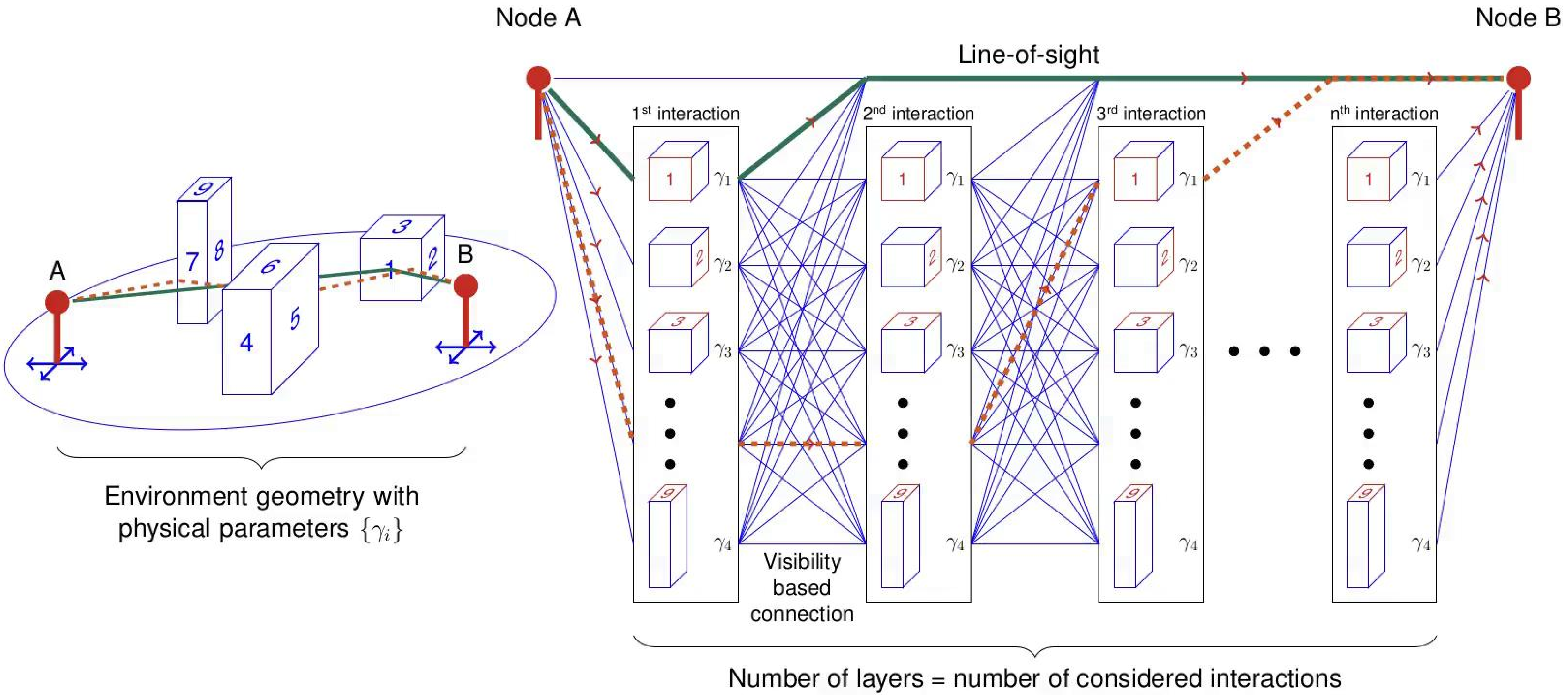
- UniSiegen, Mohammed Saleh (Pr. Andreas Kolb) - 07/2023;
- Unibo, Nicola D. C. (Pr. Vittorio D. E.) - 03/2023-12/2024;
- Huawei, Allan W. M. - 03/2023-?;
- Nvidia, Sionna, Jakob Hoydis - ?

Conclusion

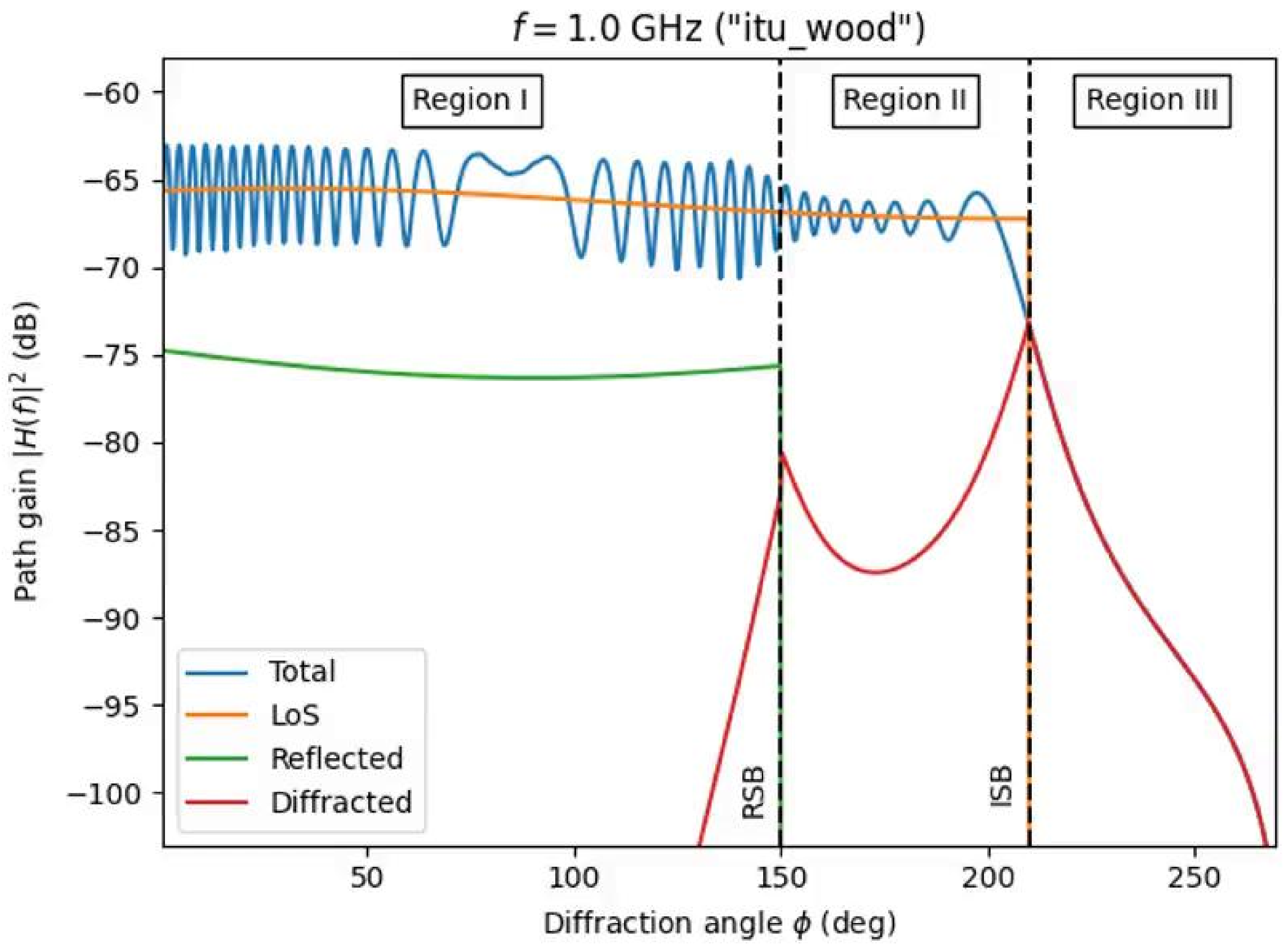
Conclusion

Questions time!

ML-like structure

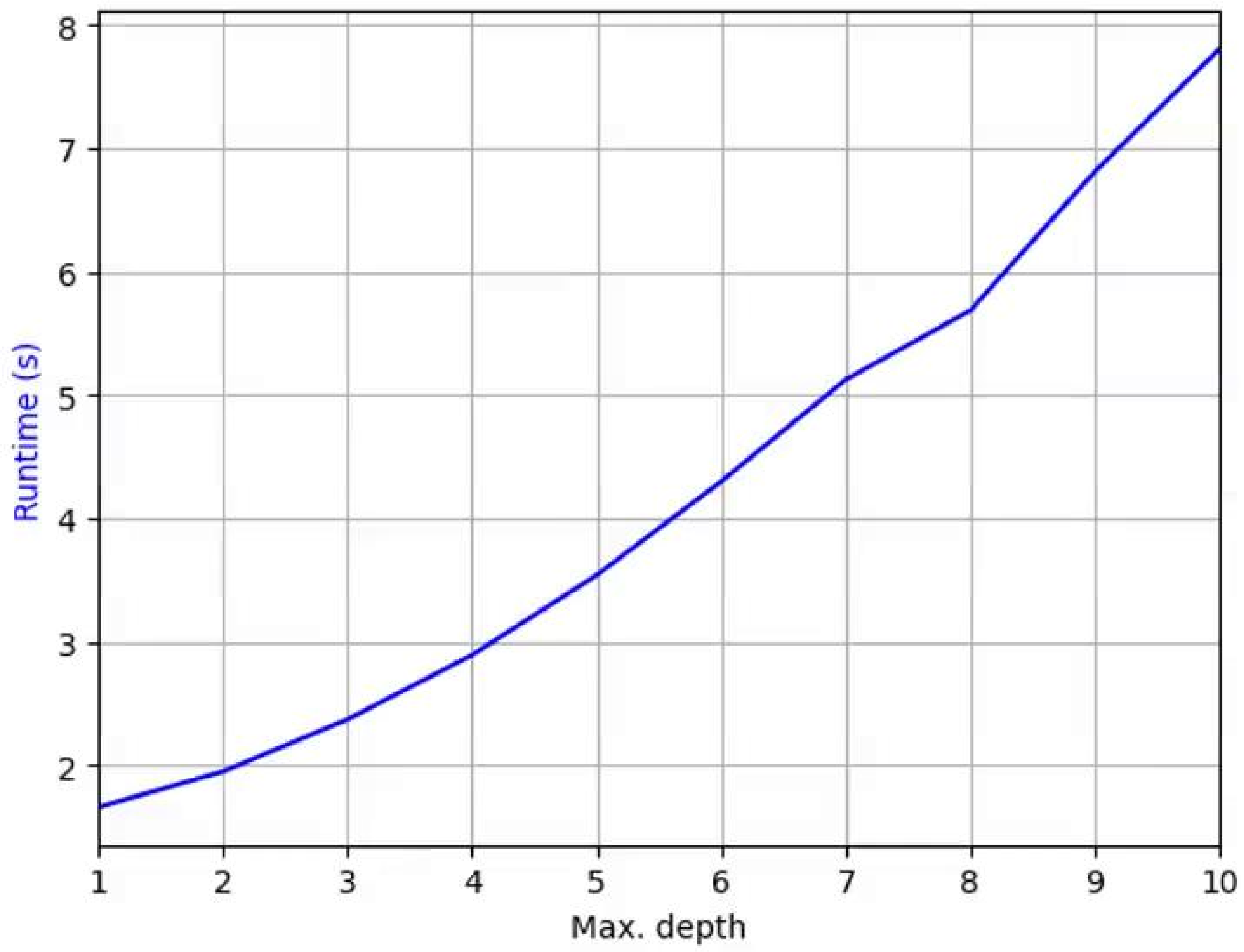


Diffraction regions



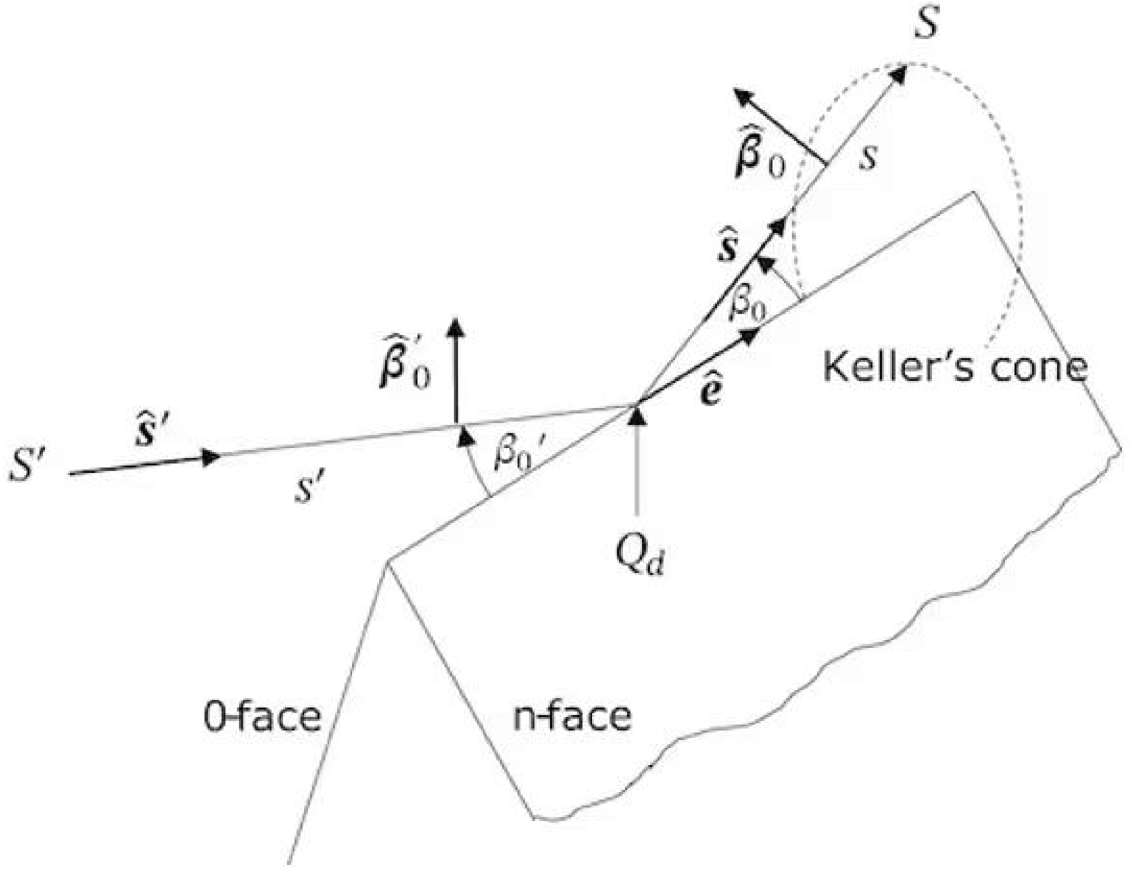
Credits: Sionna authors, Nvidia.

RT runtime



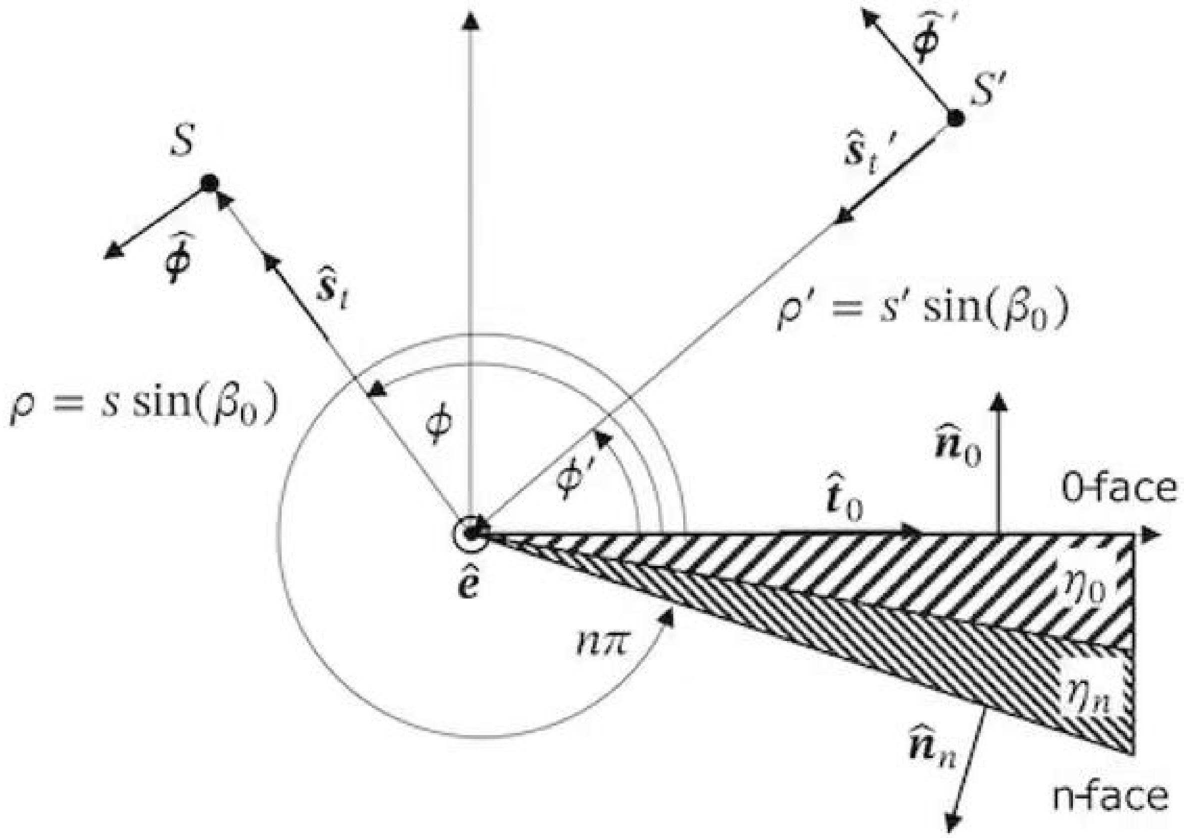
Credits: Sionna authors, Nvidia.

Keller cone



Credits: Sionna authors, Nvidia.

Edge diffraction



Credits: Sionna authors, Nvidia.