Ray tracing and Monte Carlo methods for uncertainty quantification in the pointing direction of radio telescopes for CMB observations.

Lorenzo Liuzzo

18 Luglio 2025

Motivations for this thesis

Cosmology and The Big Bang Theory

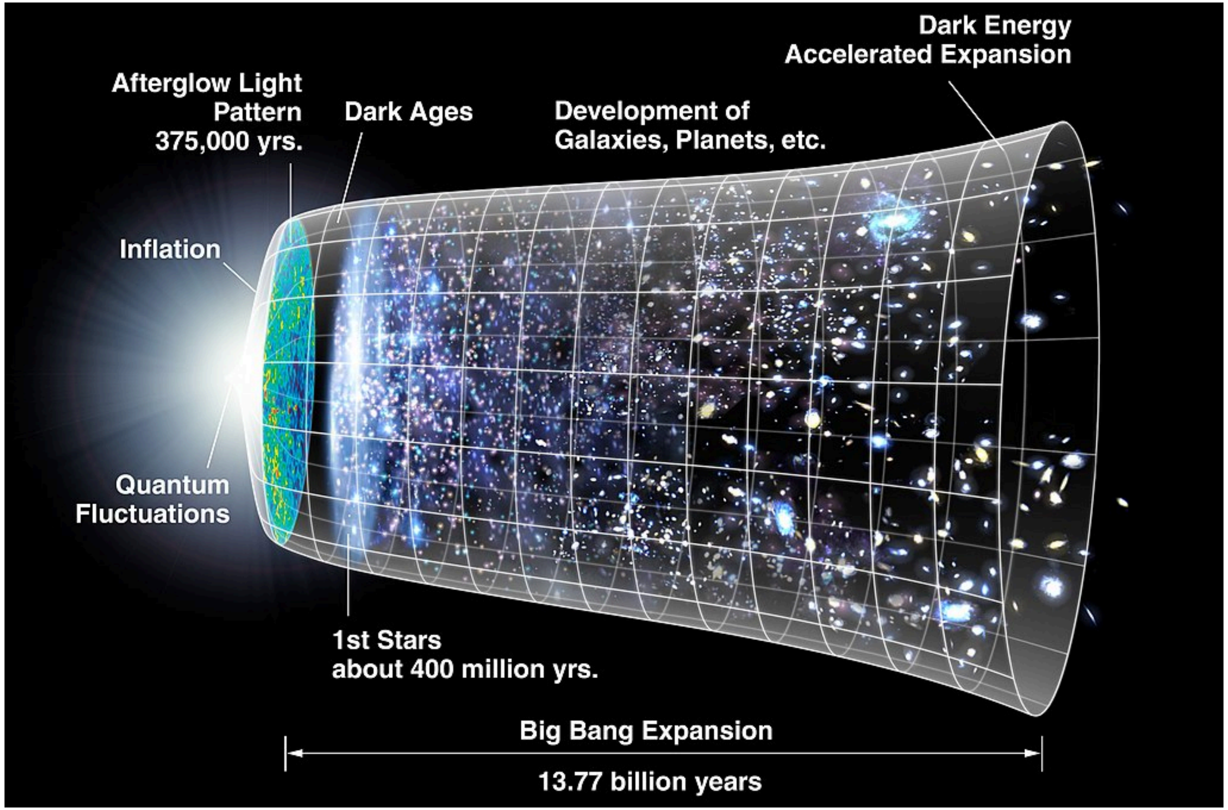


Figure 1: Evolution of the Universe according to the Big Bang theory and inflation.

Cosmic Microwave Background

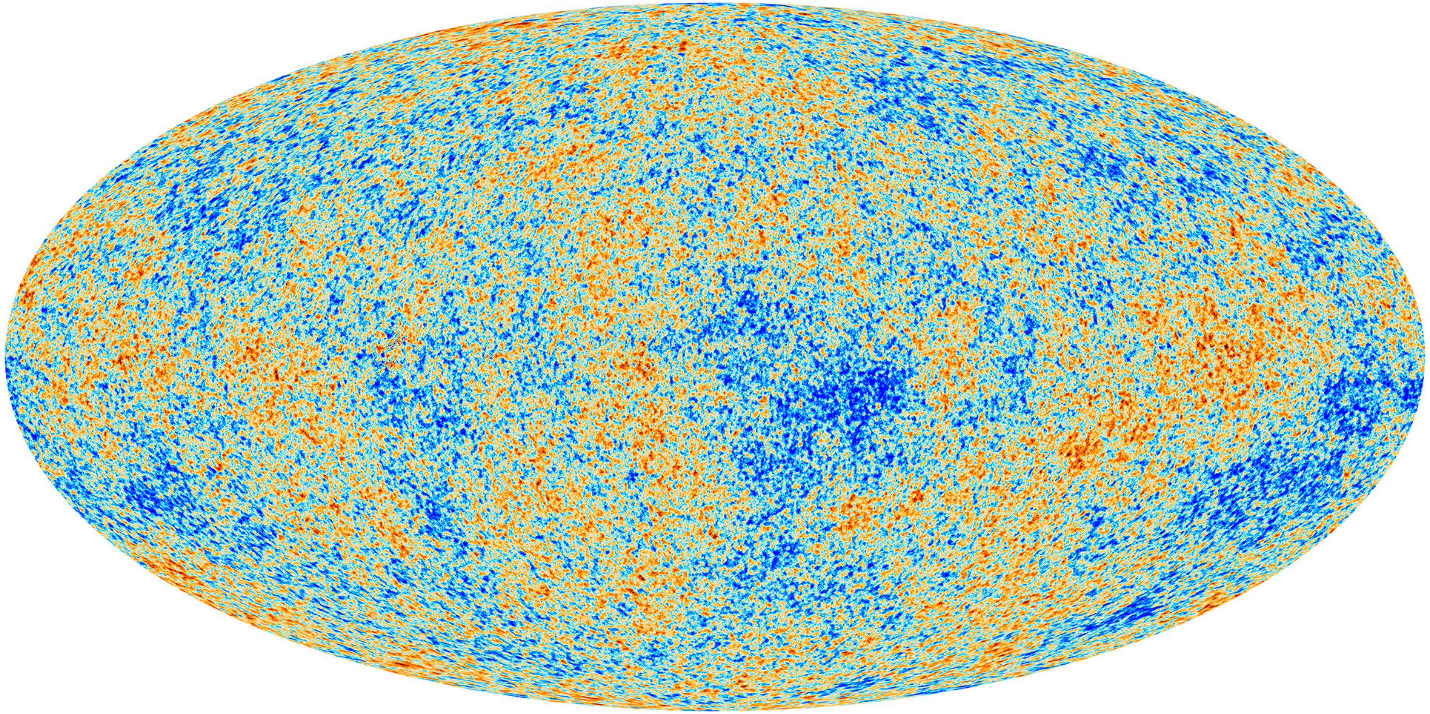


Figure 2: CMB observed by Planck instrument, image from ESA.

CMB Radio Telescopes

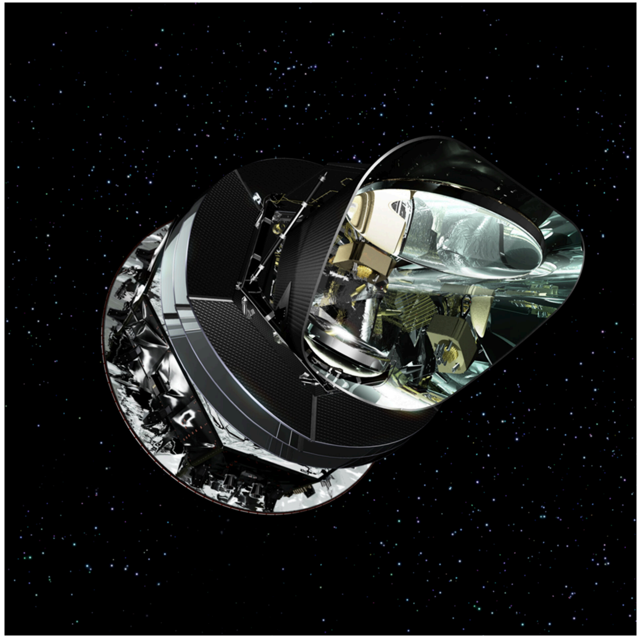


Figure 3: The Planck spacecraft, image from ESA.

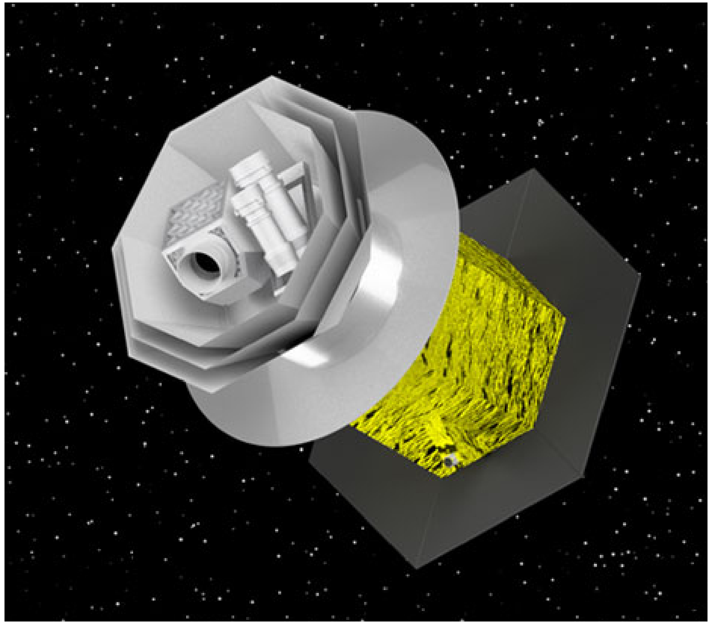


Figure 4: The LiteBIRD spacecraft, image from ISAS.

CMB Radio Telescopes

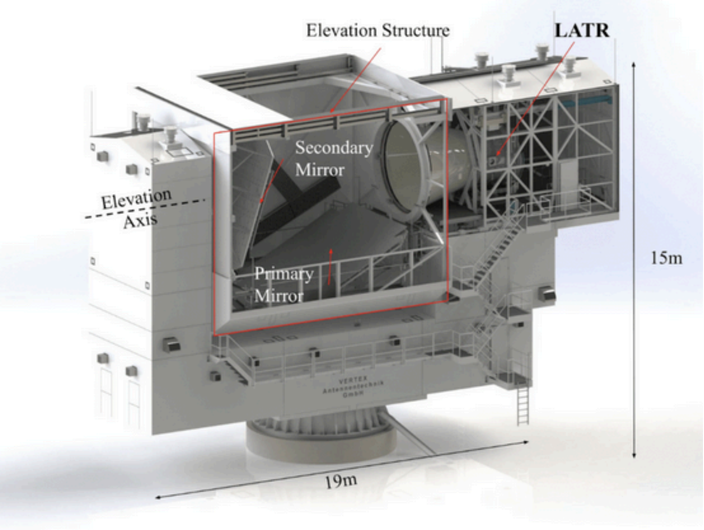


Figure 5: LATR telescope at Simons Observatory, in the Atacama Desert of Northern Chile.

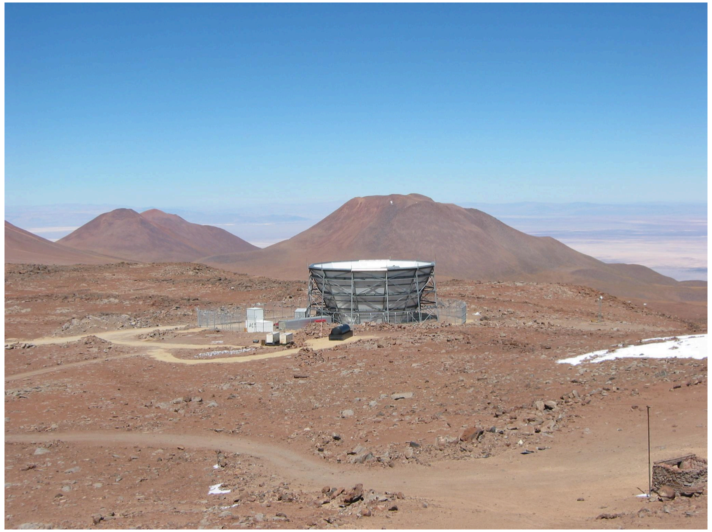


Figure 6: Atacama Cosmology Telescope (ACT), in the Atacama Desert of Northern Chile.

The Goal

- The goal is to investigate how **mechanical perturbations** in the **optical system** of a **radio telescope** influence the measurements of the **Cosmic Microwave Background**.

The Goal

- The goal is to investigate how **mechanical perturbations** in the **optical system** of a **radio telescope** influence the measurements of the **Cosmic Microwave Background**.
- There are already **proprietary softwares** (e.g., TICRA) that offer both ray-tracing and more accurate full-wave analysis, but they are computationally and **economically expensive**.

The Goal

- The goal is to investigate how **mechanical perturbations** in the **optical system** of a **radio telescope** influence the measurements of the **Cosmic Microwave Background**.
- There are already **proprietary softwares** (e.g., TICRA) that offer both ray-tracing and more accurate full-wave analysis, but they are computationally and **economically expensive**.
- In many cases we can just rely on **geometrical optics**, or often called **ray optics**.

Ray Optics

- Fermat's **Principle of Least Time**
  $\rightarrow$ laws of **Reflection** and **Refraction**.

Ray Optics

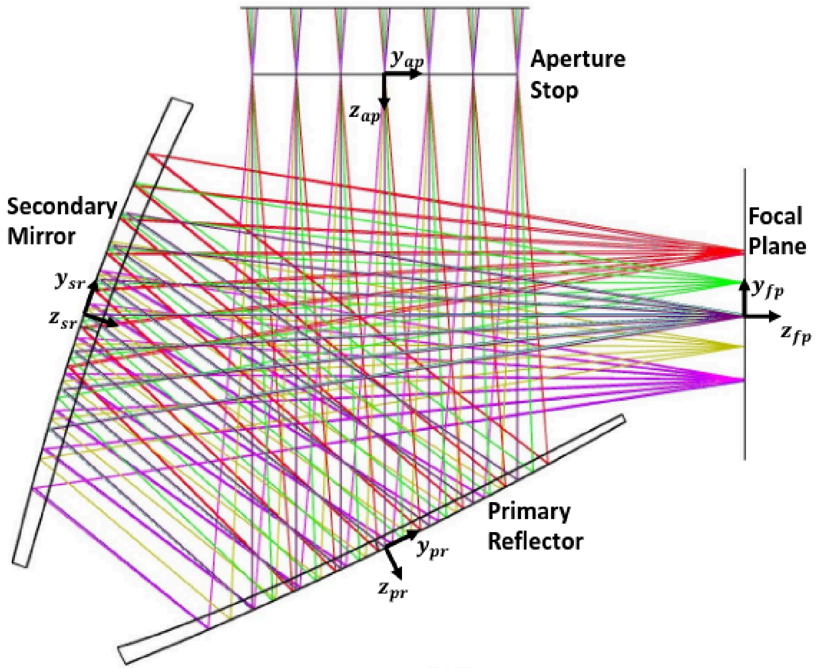
- Fermat's **Principle of Least Time**
  $\rightarrow$ laws of **Reflection** and **Refraction**.

- Helmholtz's **Reciprocity Principle**
  $\rightarrow$ optical path taken by light is **reversible**

Ray Optics

- Fermat's **Principle of Least Time**
  → laws of **Reflection** and **Refraction**.

- Helmholtz's **Reciprocity Principle**
  → optical path taken by light is **reversible**
  → trace rays from the focal plane to the sky



Ray-Tracing + Monte Carlo

- Using the **Julia** programming language, we created a **software** that can model **any telescope's reflective optical system** and implements a ray-tracing algorithm to assess the **pointing direction**.

- Jointly with **Monte Carlo simulations**, it is possible to explore **many configurations** of the system under a vast range of perturbations, in order to **propagate the uncertainty** in the telescope configuration into the measurements.

The LSPE/STRIP telescope

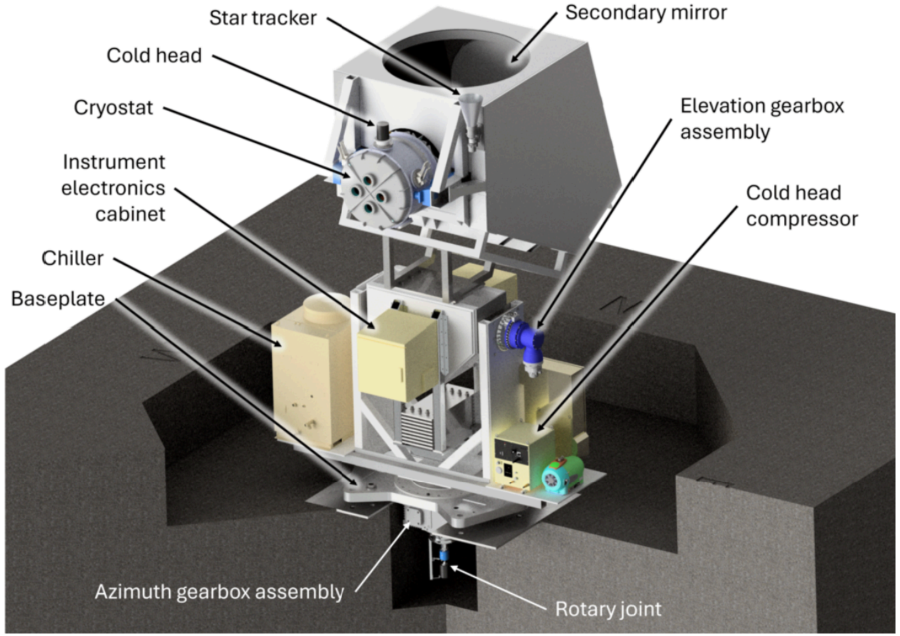


Figure 7: Rendering of LSPE-STRIP full instrument.

- As a case study, we investigated the pointing accuracy of the ***LSPE/STRIP***'s dual-reflector crossed-Dragone optical system under **tilting perturbations**.

The LSPE/STRIP telescope

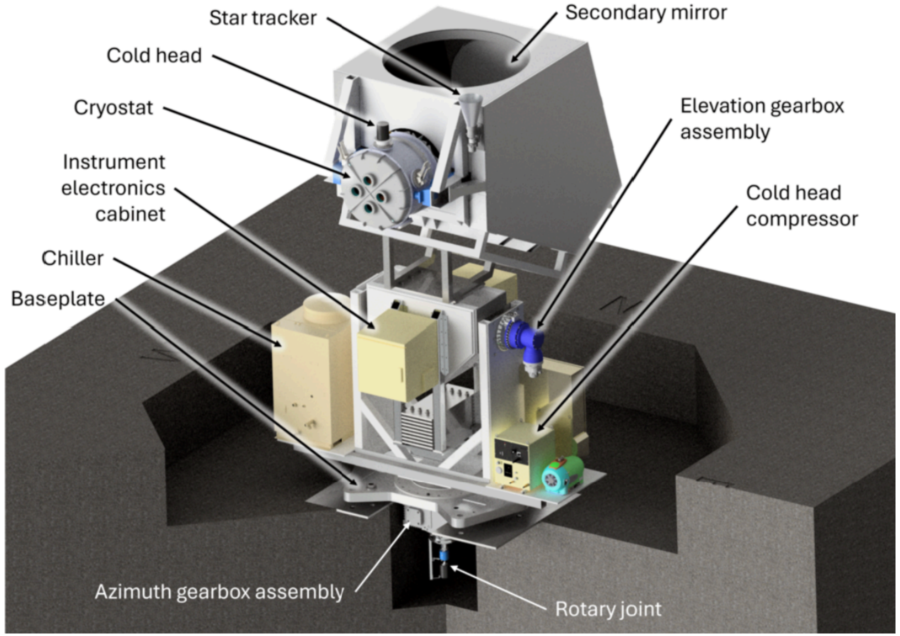


Figure 7: Rendering of LSPE-STRIP full instrument.

- As a case study, we investigated the pointing accuracy of the ***LSPE/STRIP***'s dual-reflector crossed-Dragone optical system under **tilting perturbations**.
- We **integrated** the pointing direction, evaluated with the ray-tracing software, directly into the STRIP **end-to-end pipeline** to simulate sky observations.

Ray-Tracing & MC Simulations

Tilted Reflectors and Pointing Error

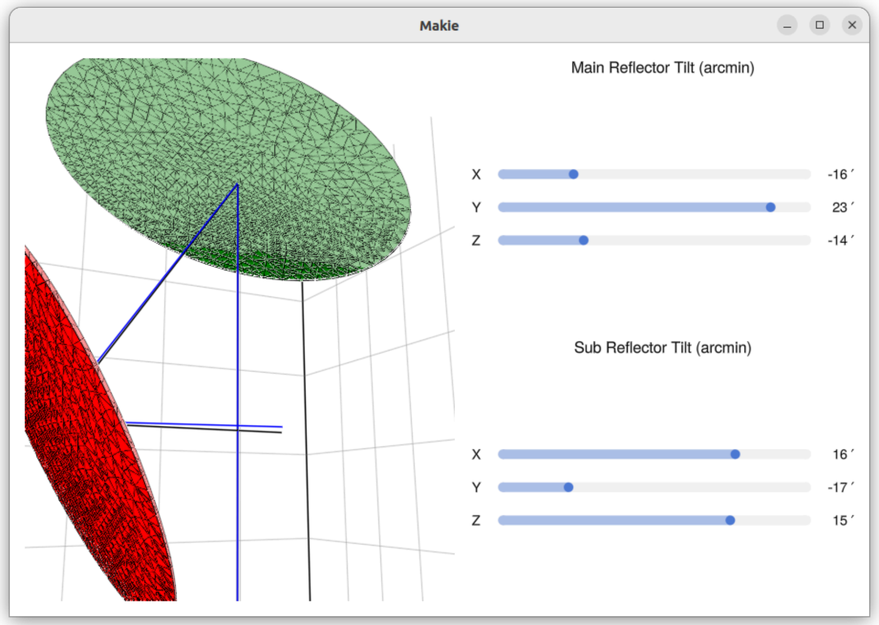


Figure 8: Interactive demo of a tilting simulation implemented with our ray-tracing engine and `Makie.jl`.

Tilted Reflectors and Pointing Error

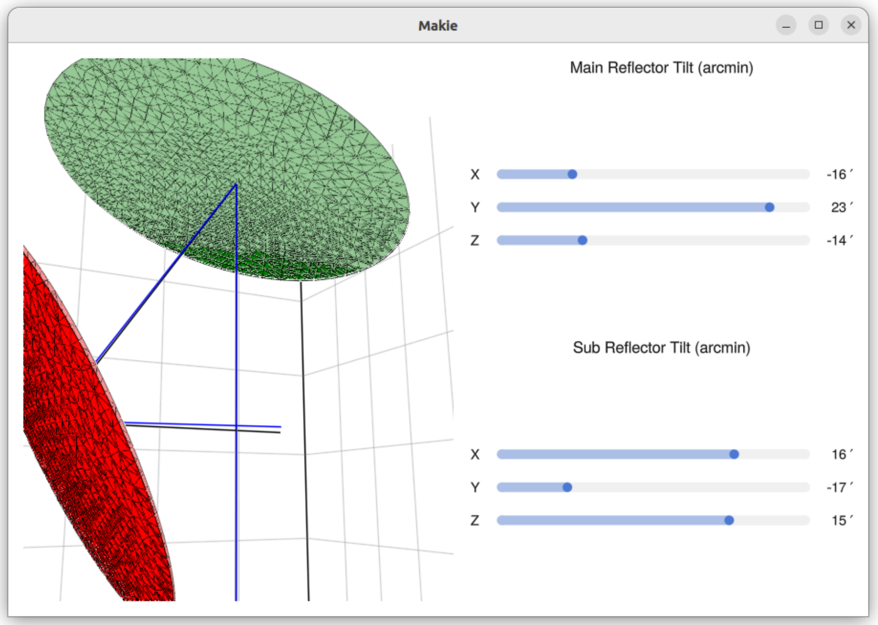


Figure 8: Interactive demo of a tilting simulation implemented with our ray-tracing engine and `Makie.jl`.

- We first create the **ideal configuration** of the optical system and trace a ray from the focal plane of the instrument to get the **ideal pointing direction** in the sky.

Tilted Reflectors and Pointing Error

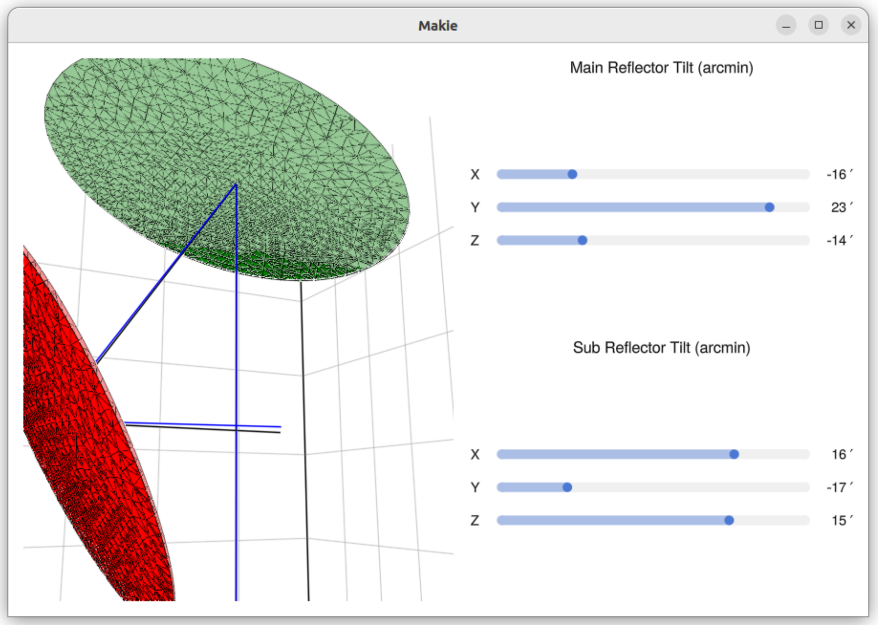


Figure 8: Interactive demo of a tilting simulation implemented with our ray-tracing engine and `Makie.jl`.

- We first create the **ideal configuration** of the optical system and trace a ray from the focal plane of the instrument to get the **ideal pointing direction** in the sky.
- Then we **perturb** the ideal system and repeat the process, obtaining a **tilted direction**.

Tilted Reflectors and Pointing Error

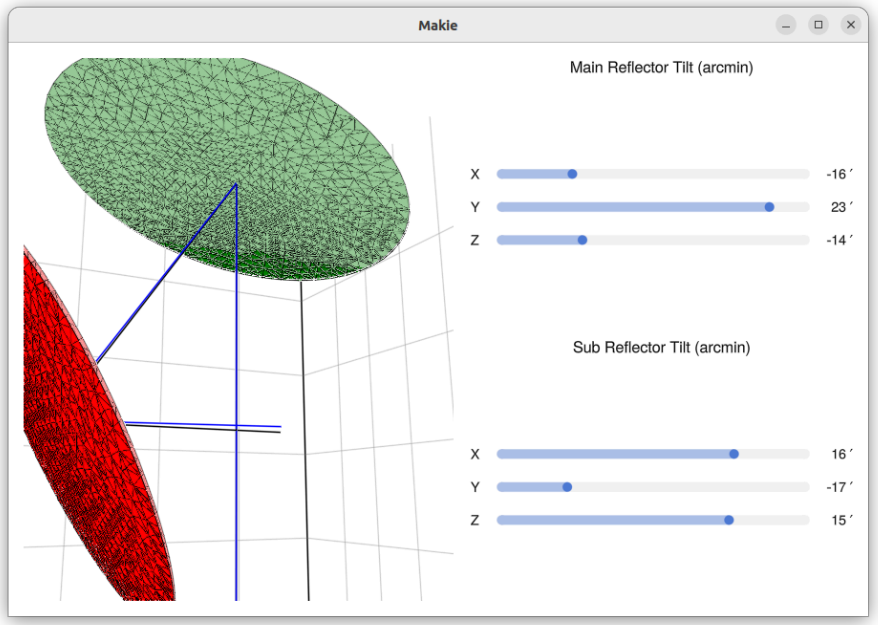
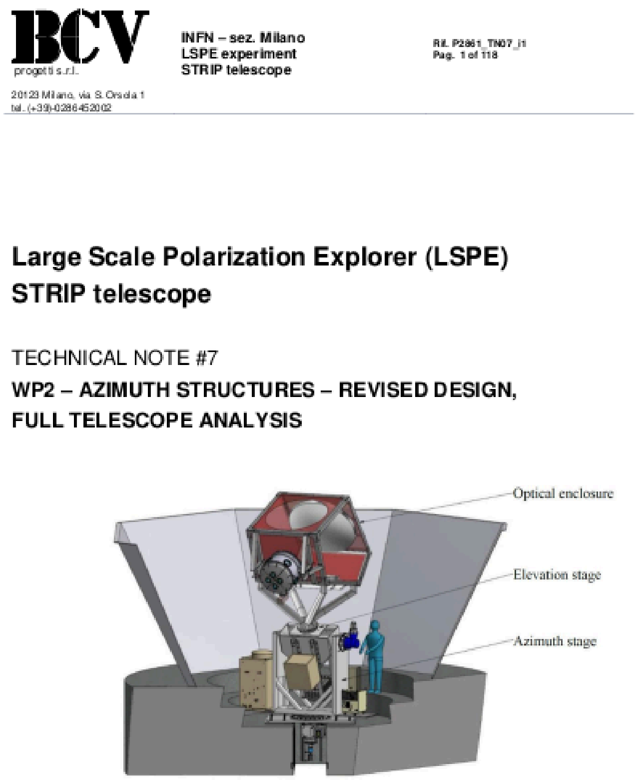


Figure 8: Interactive demo of a tilting simulation implemented with our ray-tracing engine and `Makie.jl`.

- We first create the **ideal configuration** of the optical system and trace a ray from the focal plane of the instrument to get the **ideal pointing direction** in the sky.
- Then we **perturb** the ideal system and repeat the process, obtaining a **tilted direction**.

$$\text{POINTING ERROR} \equiv \arccos\left(\hat{P}_{\text{ideal}} \cdot \hat{P}_{\text{tilted}}\right)$$

Simulations SetUp


BCV progetti s.r.l.
20123 Milano, via S. Orsola 1
tel. (+39)-0286452002

INFN – sez. Milano
LSPE experiment
STRIP telescope

Rif. P2861_TN07_i1
Pag. 1 of 118

Large Scale Polarization Explorer (LSPE)
STRIP telescope

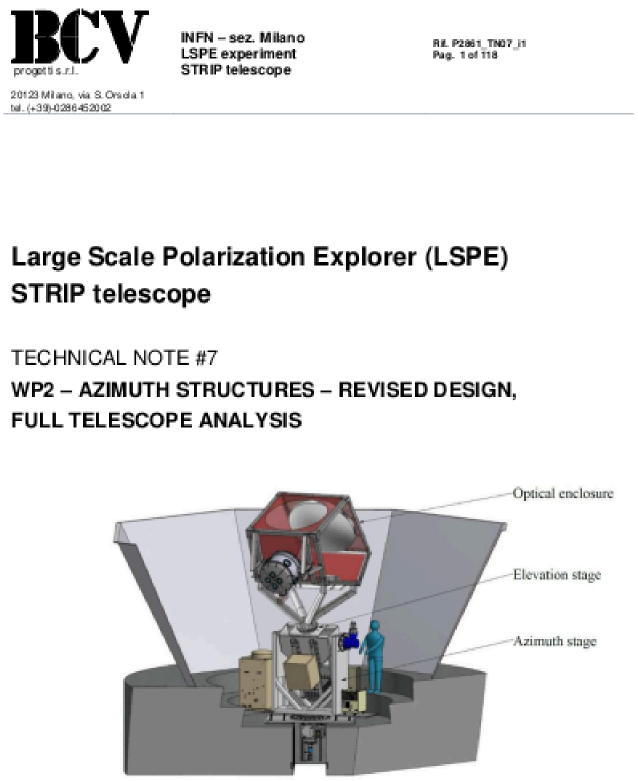
TECHNICAL NOTE #7
WP2 – AZIMUTH STRUCTURES – REVISED DESIGN, FULL TELESCOPE ANALYSIS

- We consider **independent** tilt angles

$$\theta_{a_i} \sim \mathcal{N}(0, \sigma^2)$$

$$a \in \{\text{primary}, \text{secondary}\}, \quad i \in \{\text{x}, \text{y}, \text{z}\}$$

Simulations SetUp



BCV progetti s.r.l.

20123 Milano, via S. Orsola 1
tel. (+39)-0286452002

INFN – sez. Milano
LSPE experiment
STRIP telescope

Rif. P2861_TN07_r1
Pag. 1 of 118

Large Scale Polarization Explorer (LSPE)
STRIP telescope

TECHNICAL NOTE #7

WP2 – AZIMUTH STRUCTURES – REVISED DESIGN, FULL TELESCOPE ANALYSIS

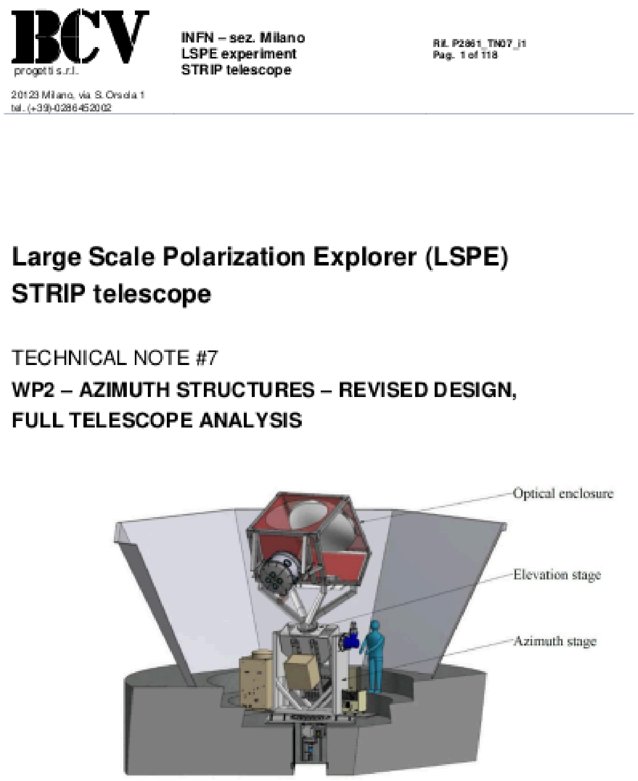
- We consider **independent** tilt angles

$$\theta_{a_i} \sim \mathcal{N}(0, \sigma^2)$$

$$a \in \{\text{primary}, \text{secondary}\}, \quad i \in \{\text{x}, \text{y}, \text{z}\}$$

- We start the simulations with an **exaggerated value** of $\sigma = 1°$ for debugging purposes, but we expect tilting errors approximately of the order of 10^2 arcsec.

Simulations SetUp



BCV progetti s.r.l.
20123 Milano, via S. Orsola 1
tel. (+39)-0286452002

INFN – sez. Milano
LSPE experiment
STRIP telescope

Rif. P2861_TN07_i1
Pag. 1 of 118

Large Scale Polarization Explorer (LSPE)
STRIP telescope

TECHNICAL NOTE #7
WP2 – AZIMUTH STRUCTURES – REVISED DESIGN, FULL TELESCOPE ANALYSIS

- We consider **independent** tilt angles

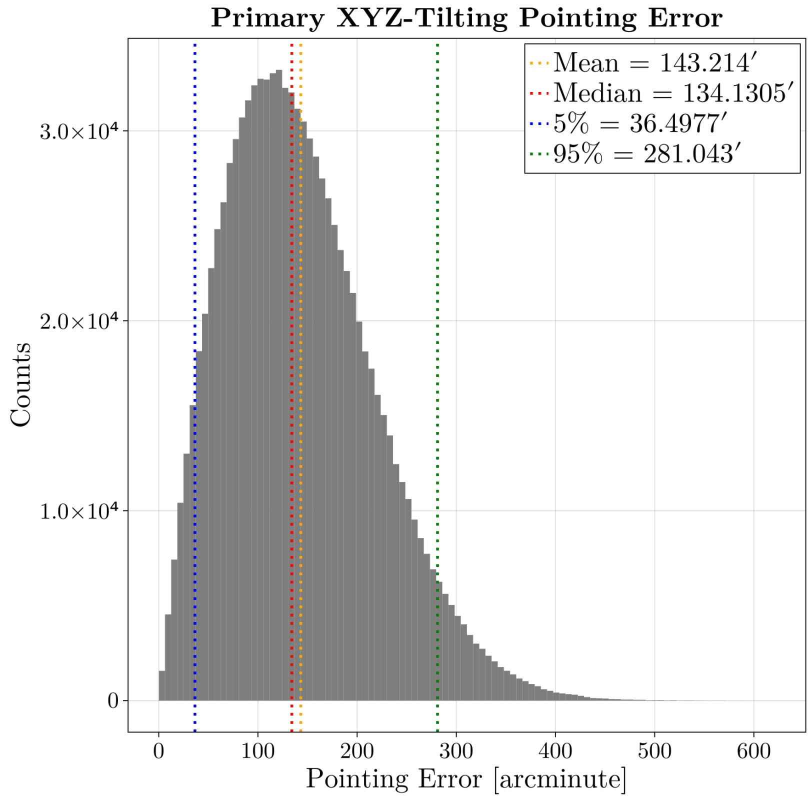
$$\theta_{a_i} \sim \mathcal{N}(0, \sigma^2)$$

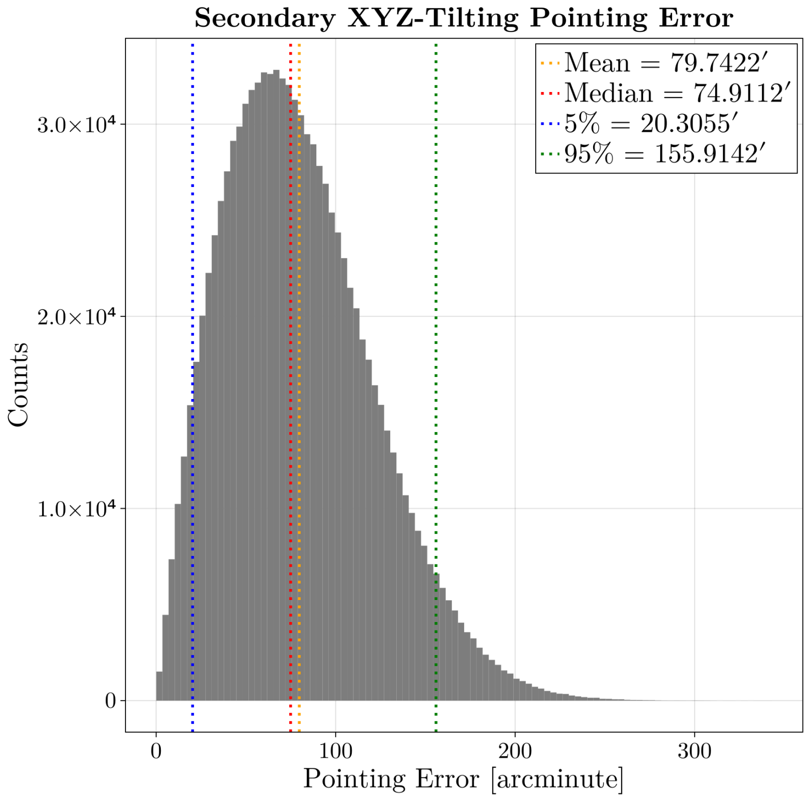
$$a \in \{\text{primary}, \text{secondary}\}, \quad i \in \{\text{x}, \text{y}, \text{z}\}$$

- We start the simulations with an **exaggerated value** of $\sigma = 1^\circ$ for debugging purposes, but we expect tilting errors approximately of the order of 10^2 arcsec.

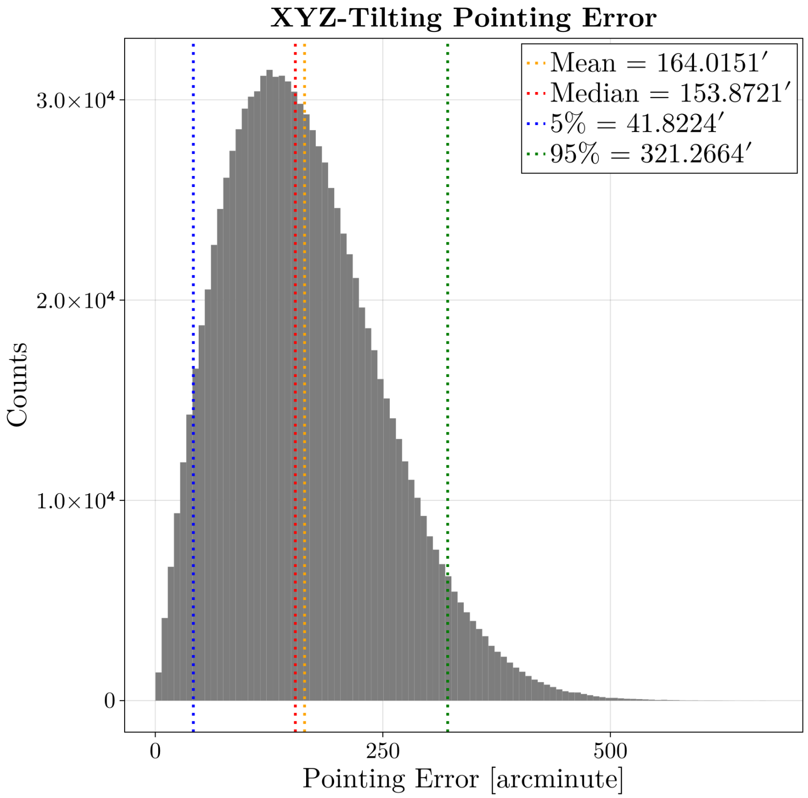
- Later we studied how the **pointing error scales** with the variance σ^2.

Multiple-Axis Single-Reflector Tilt

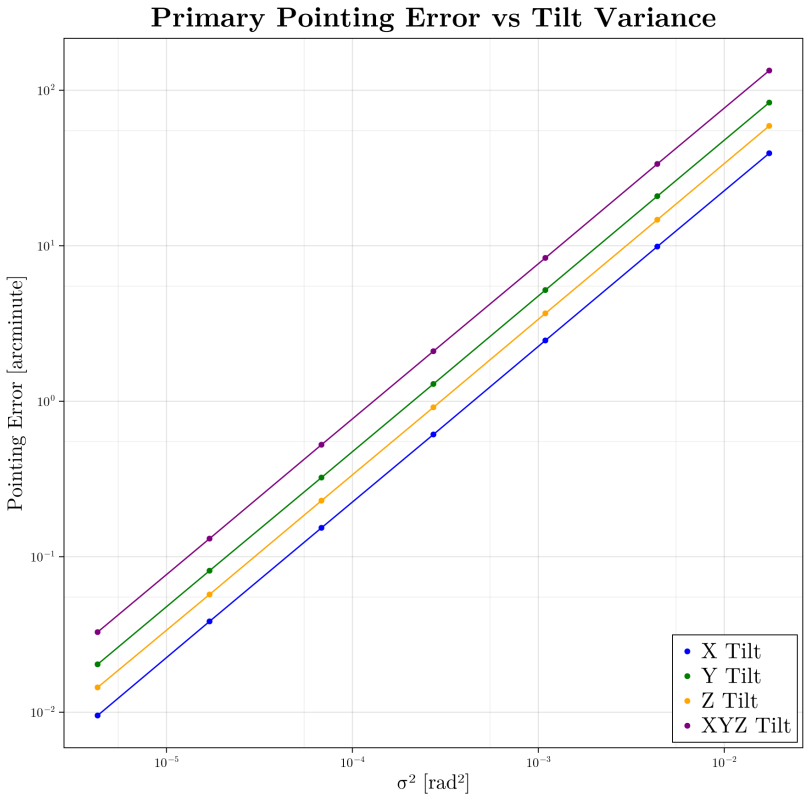


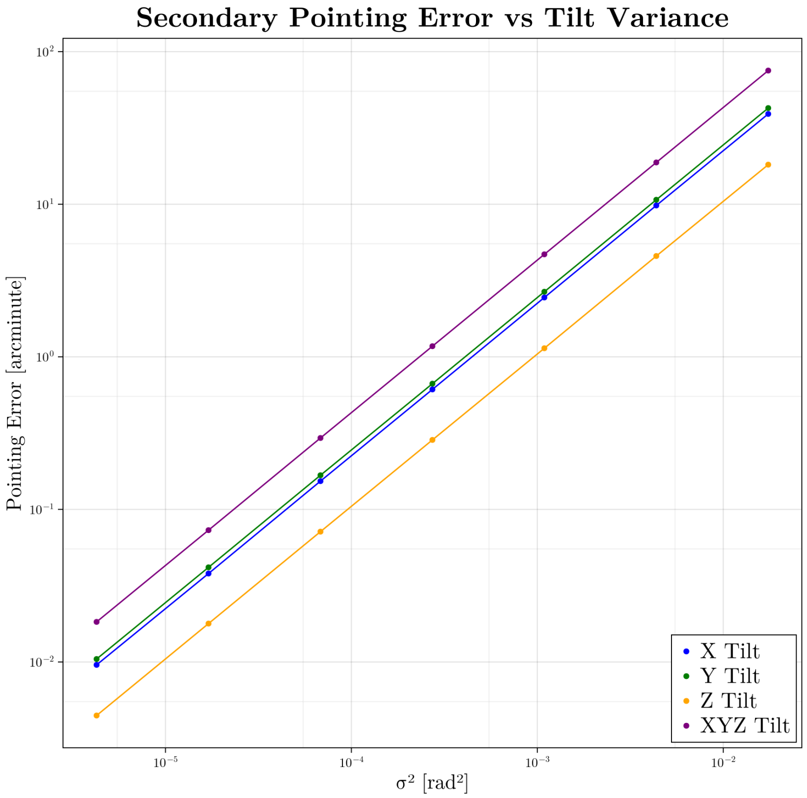


Multiple-Axis Multiple-Reflector Tilt



Pointing Error Trends





CMB Error Maps

Sky Observations using `STRIPLINE.jl`

We integrated the pointing direction evaluated by our ray-tracer into `STRIPLINE.jl`, the end-to-end scanning strategy pipeline for STRIP.

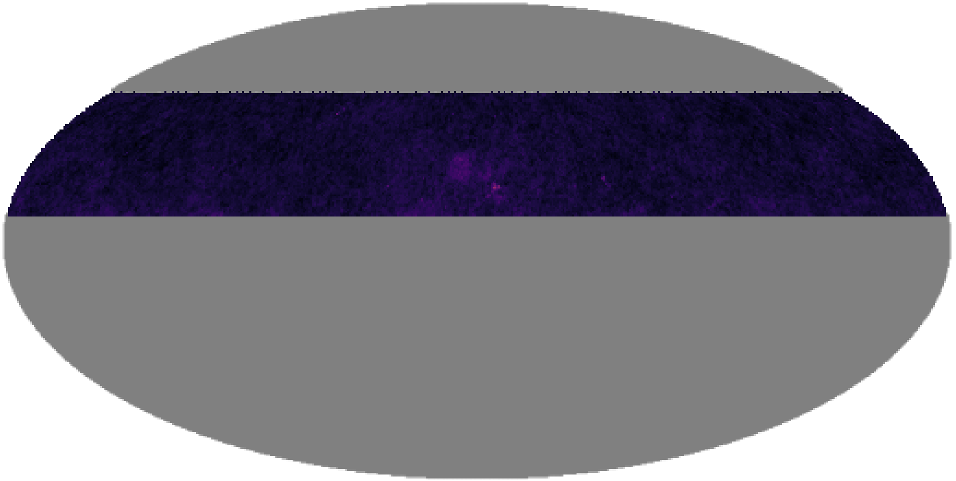


Figure 9: Simulated sky map observed by STRIP in a year of sky observation. The telescope, situated at the Canary Islands, will observe a strip of northern sky.

Sky Observations using `STRIPLINE.jl`

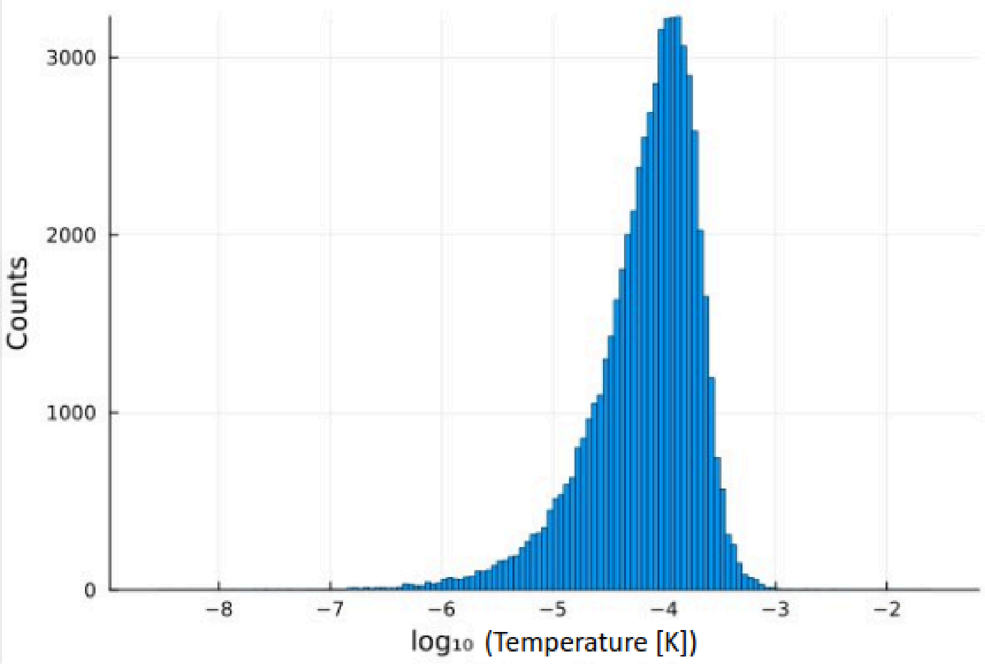


Figure 10: Histogram of the logarithm in base 10 of the error maps pixels.

Conclusion and Outlooks

Conclusion

- Through the **development of a ray-tracing engine** using the Julia programming language, the application with **Monte Carlo techniques** and the **integration into** the STRIP simulation **pipeline**, this study effectively demonstrates a flexible and computationally efficient **framework for optical validation** of CMB measurements.

Conclusion

- Through the **development of a ray-tracing engine** using the Julia programming language, the application with **Monte Carlo techniques** and the **integration into** the STRIP simulation **pipeline**, this study effectively demonstrates a flexible and computationally efficient **framework for optical validation** of CMB measurements.

- To our knowledge, this represents the **first integration** of a geometrical ray-tracing engine into a full end-to-end scanning strategy pipeline.

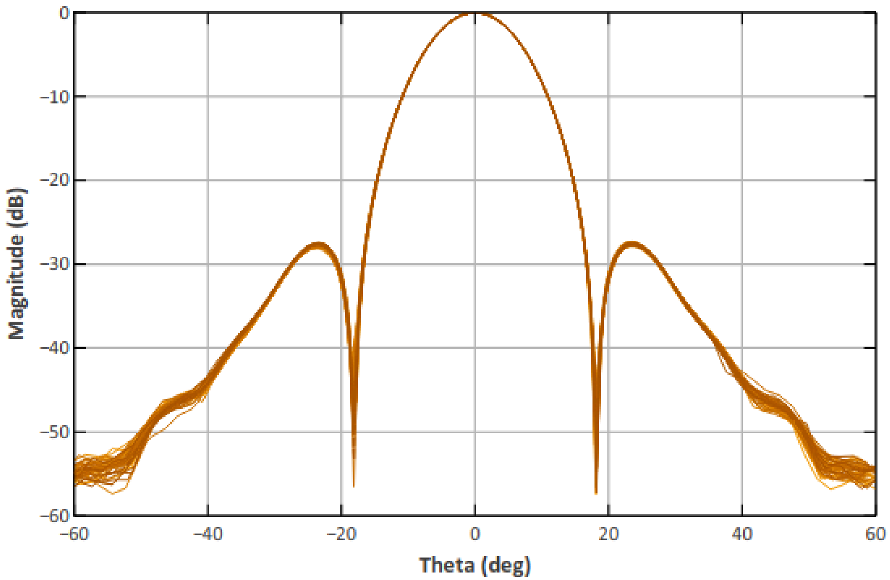
Outlooks

- Investigate **tilt angles correlations**.
- **AD** for uncertainty propagation, optimization problems and visualization of the pointing error gradients.
- Model surface imperfections with **Zernike Polynomials**.
- **Diffraction effects** with PO and MoM.
- Analyze **different kinds of perturbations** on **different telescopes**.

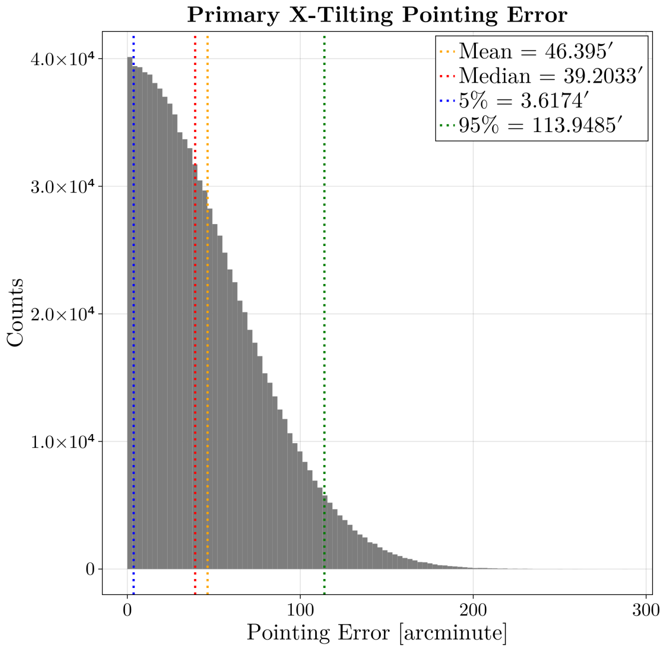
Thank you for the attention

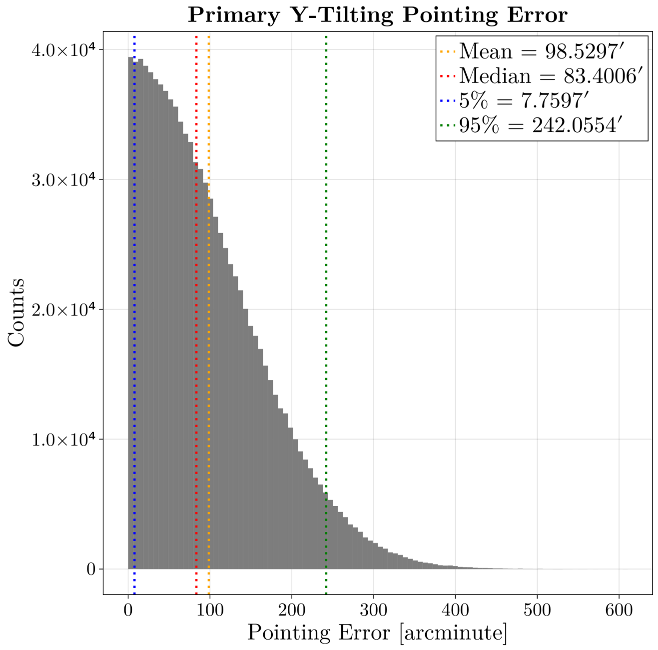
Appendix

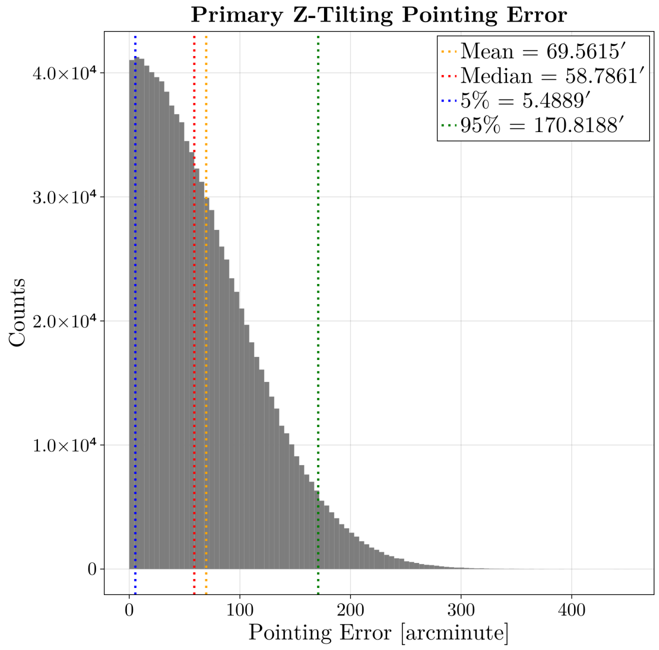
Feed’s radiation pattern



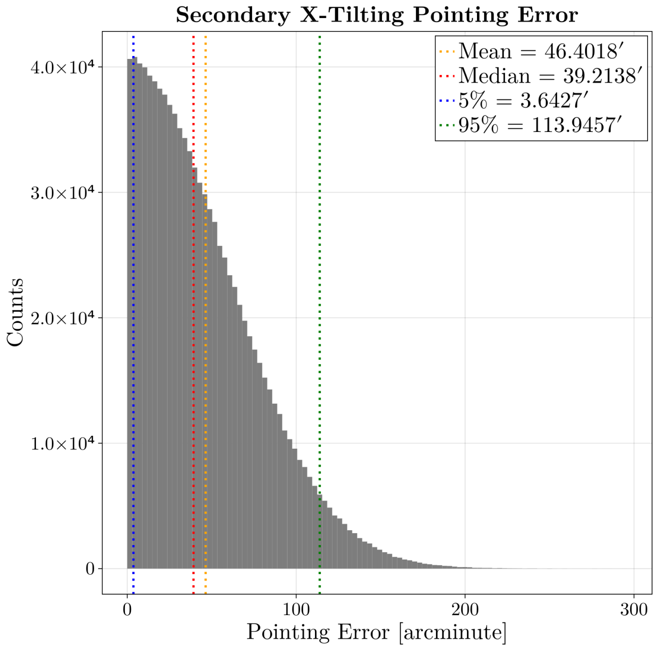
Single-Axis Single-Reflector Tilt

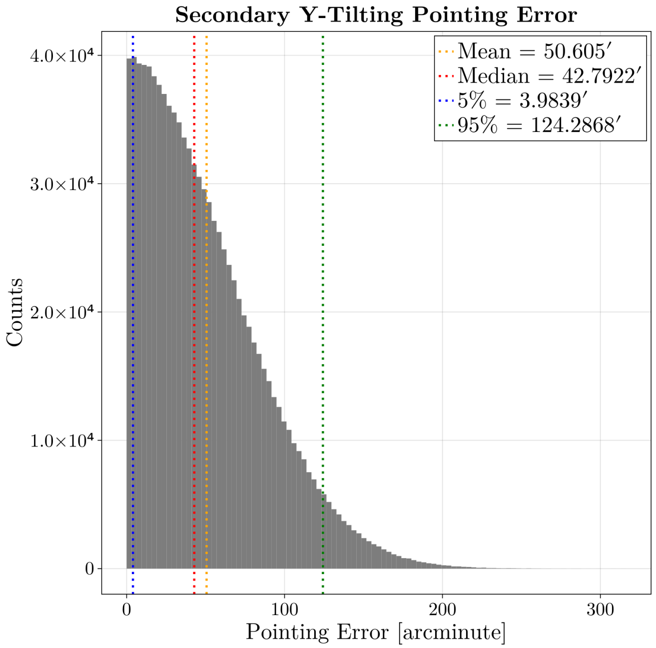


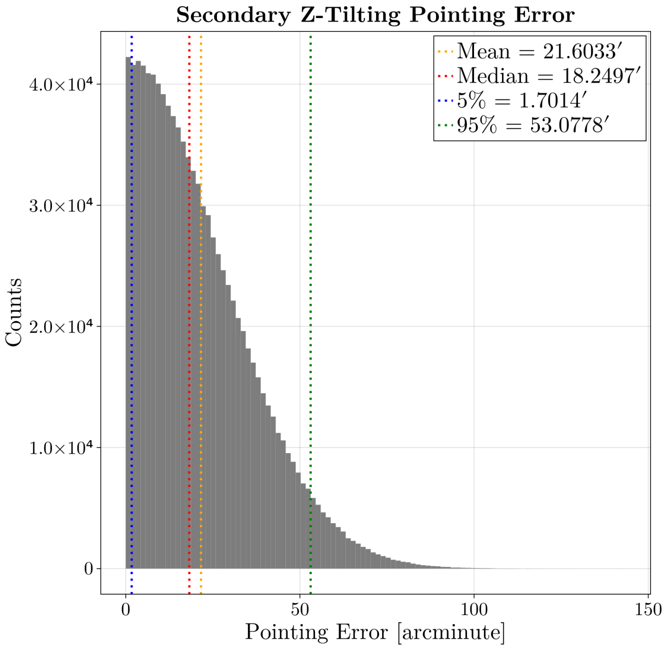




Single-Axis Single-Reflector Tilt







Let’s build a ray-tracing engine!

Julia Programming Language

- Modern language for **scientific** computing.
- Combines Python-like syntax with **C-like performance**.
- Excellent packages: `StaticArrays.jl`, `CoordinateTransformations.jl` and `Makie.jl`.
- Handles **units** and **automatic differentiation** very easily with type dispatch.
- An existing end-to-end simulation, `STRIPLINE.jl`, that models the **scanning strategy** of the telescope up to the creation of **frequency maps**.

Julia implementation of a Ray

Julia

```
using StaticArrays

struct Ray{O, D}
    origin::SVector{3, O}
    direction::SVector{3, D}
end

(ray::Ray)(t::Real) = ray.origin + ray.direction * t
```

Julia implementation of an AbstractSurface

```julia
abstract type AbstractSurface end

function (Σ::AbstractSurface)(u, v) = SVector(u, v, height(Σ, u, v))

function height(Σ::AbstractSurface, u, v) end
function gradient(Σ::AbstractSurface, u, v) end
function normal(Σ::AbstractSurface, u, v) end

function quad_intersection(Σ::AbstractSurface, ray) end
```

Julia implementation of a Reflector

```julia
using CoordinateTransformations

const Mapping{T} = Tuple{T, Transformation}

struct Reflector{Σ <: AbstractSurface, Γ <: AbstractRim}
    surface::Mapping{Σ}
    rim::Mapping{Γ}
    name::String
end
```