Locating Strongholds with Triangulation and Estimating Measurement Error

PixelRayn

@physikdavid

August 4, 2025

Abstract

A formal method for triangulating the stronghold location with two eye of ender throws is derived. To quantify the uncertainty of the measured location a Monte Carlo method is leveraged, accounting for user input error in position and yaw angle. The presented tequnique is relevant to Minecraft speedrunning and tool-assisted gameplay. The accompanying implementation offers an efficient and statistically informed solution for in-game navigation.

Keywords: Minecraft, Speedrunning, Tools, Navigation

1. Introduction

Every Minecraft world has a total of 128 strongholds arranged in concentric rings around the origin of the coordinate system [1]. The intended way for a player to locate the closest stronghold is via the eye of ender item. When activated the eye of ender flies approximately 12 blocks in the direction of the north-west corner of the chunk containing the spiral staircase room of the stronghold. While the eye of ender targets the chunk coordinates (0, 0), the entrance to the stronghold is always generated at coordinates (4, 4). When the player is more than 12 blocks from the target coordinate the eye of ender travels upward giving a clear indication of the direction to travel in [2].

Since the eye of ender has a 20% chance of shattering, which consumes the item, a strategy to minimize the number of pearls thrown is desirable. This paper reiterates the algebraic method for locating a stronghold with as little as two eyes of ender and introduced monte carlo simulations to estimate the uncertainty of the calculated location. The triangulation method discussed here is not new, instead it has been repeated and passed down through the community so long that the original authorship is now unclear. References [3] and [4] date the methods to at least 2013.

2. Triangulation

The location of the stronghold in the X-Z plane is calculated using two consecutive throws of eyes of ender. For each throw the coordinate of the player is noted with the angle of the eyes trajectory via the debug screen [5]. A large separation of the two location improves accuracy of the estimate. Let the target coordinates be

$$\vec{r}_t = \begin{bmatrix} x_t \\ z_t \end{bmatrix} \tag{1}$$

The intersecting headings are parameterized in the X-Z plane by

$$f_i(x) = m_i x - z_i^0 \tag{2}$$

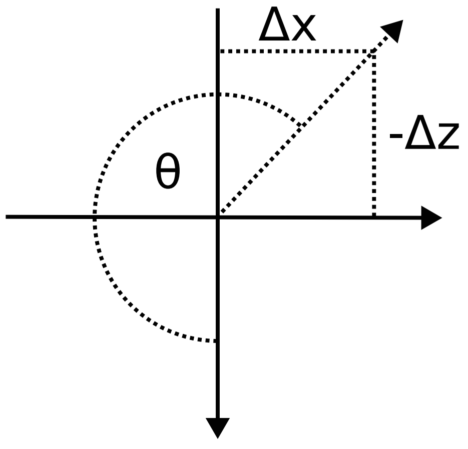
Fulfilling the necessary requirement that

$$f_i(x_i) = m_i x_i - z_i^0 = z_i \implies z_i^0 = m_i x_i - z_i \tag{3}$$

By facing the eye of ender in the air, the yaw direction shown in the debug screen directly corresponds to the travel direction of the eye of ender constraining the target coordinates.

2.1. Calculating the Line Slope from Yaw

The yaw-angle θ is measured from the z-Axis. The south direction (towards positive z) corresponds to $\theta = 0$ and the clockwise direction corresponds to positive angle.



The slope m is calculated as

$$m = \frac{\Delta z}{\Delta x} = -\tan(90 - \theta) \tag{4}$$

2.2. Solving for $\vec{r}_t$

For the two measurement locations two separate functions f_1 and f_2 are constructed. For $\vec{r}_t$ the two functions must intersect:

$$f_1(x_t) = f_2(x_t) = z_t \tag{5}$$

By explicitly writing the functions as a system of linear equations in respect to z_t and isolating all target terms on the left side of the equation the following can be shown:

$$m_1 x_t - z_t = m_1 x_1 - z_1 \tag{6}$$

$$m_2 x_t - z_t = m_2 x_2 - z_2 \tag{7}$$

The system of linear equations is expressed as:

$$\underbrace{\begin{bmatrix} m_1 & -1 \\ m_2 & -1 \end{bmatrix}}_{M} \cdot \underbrace{\begin{bmatrix} x_t \\ z_t \end{bmatrix}}_{\vec{r}_t} = \underbrace{\begin{bmatrix} m_1 x_1 - z_1 \\ m_2 x_2 - z_2 \end{bmatrix}}_{\vec{z}_0} \tag{8}$$

Assuming $m_1 \neq m_2$ and $m_1 \neq -m_2$ M is invertible. Since $M^{-1}M = \mathbb{1}$ the calculation of $\vec{r}_t$ is therefore rephrased as inverting the Matrix M.

$$\vec{r}_t = M^{-1}\vec{z}_0 \tag{9}$$

One procedure for matrix inversion is the Gauss-Jordan algorithm [6, p. 436] which yields the following inverse matrix:

$$M^{-1} = \begin{bmatrix} \frac{1}{m_1 - m_2} & \frac{1}{m_2 - m_1} \\ -\frac{m_2}{m_1 + m_2} & 1 - \frac{m_1}{m_1 + m_2} \end{bmatrix} \tag{10}$$

3. Error Estimation

A Monte Carlo simulation-method commonly known as a “parametric bootstrap” is used to compute the confidence statistics of the estimate [7, pp. 53–56]. The errors from x, z, and θ measurements are assumed to be normally distributed and uncorrelated. Samples are drawn from the simulated probability density functions and the predicted stronghold position is calculated for each simulated measurement. Confidence intervals are chosen at the upper and lower quantile to correspond to a 1σ interval for x and z independently. Drawing a histogram for an example set of measurements reveals a strong correlation between the x and the z coordinate for measurements with little angular separation as an emergent property of the measurement. (Compare fig. 1.)

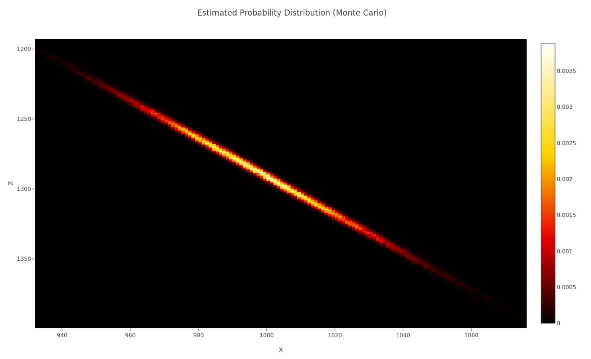


Figure 1: Example measurement of a stronghold position with $x_1 = 10$, $z_1 = 33$, $\theta_1 = -38.2°$, $x_2 = 119$, $z_2 = 19$, $\theta_2 = -34.7°$. Errors are estimated with $\Delta x = \Delta z = 0.5$ and $\Delta\theta = 0.1°$. Graphic is generated with Ref. [8]

4. Conclusion

A method for triangulating the stronghold location is formally reintroduced and a numeric method for estimating the uncertainties is discussed. A locally running implementation using JavaScript can be found in ref. [8].

References

[1] Minecraft Wiki Contributors. *Minecraft Wiki: Stronghold.* Accessed: 2025-08-04. 2025. URL: `https://minecraft.wiki/w/Stronghold`.

[2] Minecraft Wiki Contributors. *Minecraft Wiki: Eye of Ender.* Accessed: 2025-08-04. 2025. URL: `https://minecraft.wiki/w/Eye_of_Ender#Locating_strongholds`.

[3] William Goosen. *How to locate the End Portal.* Accessed: 2025-08-04. 2013. URL: `https://youtu.be/mYD3UmIhIvQ?si=5gjG27PwB4DYKIrO`.

[4] Ben. *End Portal Triangulation for Minecraft Random-Seed Glitchless Speedrunning.* Accessed: 2025-08-04. 2016. URL: `https://youtu.be/Ak-9tRDeORA?si=q_11C3Tl1bbydWGG`.

[5] Minecraft Wiki Contributors. *Minecraft Wiki: Debug Screen.* Accessed: 2025-08-04. 2025. URL: `https://minecraft.wiki/w/Debug_screen`.

[6] Kendall E. Atkinson. *An introduction to numerical analysis.* 1978. ISBN: 0471029858.

[7] Bradley Efron and Robert Tibshirani. *An introduction to the bootstrap.* 1998. ISBN: 0412042312.

[8] David Kowalk. *End Portal Triangulator*. 2025. URL: `https://github.com/davidkowalk/end_portal_triangulator`.